



universität
wien

DISSERTATION / DOCTORAL THESIS

Titel der Dissertation / Title of the Doctoral Thesis

„Hubness in the Protein Sequence Universe“

verfasst von / submitted by

Roman Feldbauer BSc MSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Doctor of Philosophy (PhD)

Wien, 2020 / Vienna 2020

Studienkennzahl lt. Studienblatt /
degree programme code as it appears on the student
record sheet:

UA 794 685 437

Dissertationsgebiet lt. Studienblatt /
field of study as it appears on the student record sheet:

Biology

Betreut von / Supervisor:

Univ. Prof. Dr. Thomas Rattei

Acknowledgements

I thank my supervisors Arthur Flexer and Thomas Rattei for their continuous support in all aspects of my academic training over the last five years. Both of you taught me a great many things, and analyzing the intersection and union of your advice gives me yet more insight.

Many thanks also to all previous and current members of OFAI and CUBE. Not only for providing scientifically stimulating working environments, inspiring discussions, and most valuable input—you also made both labs fun to work in. It is my great pleasure to have had you as my colleagues.

While a dissertation may demonstrate the doctoral student’s ability to carry out independent scientific work as the university guidelines state, no scientific work is pursued in isolation. Therefore, I’d like to express my gratitude to all my coauthors for joining forces. It was a great experience to work with you on most diverse topics.

Thanks also to my family and friends for always supporting me on my path, and providing the framework that makes all of it possible.

Finally, my deepest gratitude is to Moni. Thank you so much for these great photos... and the wonderful life!

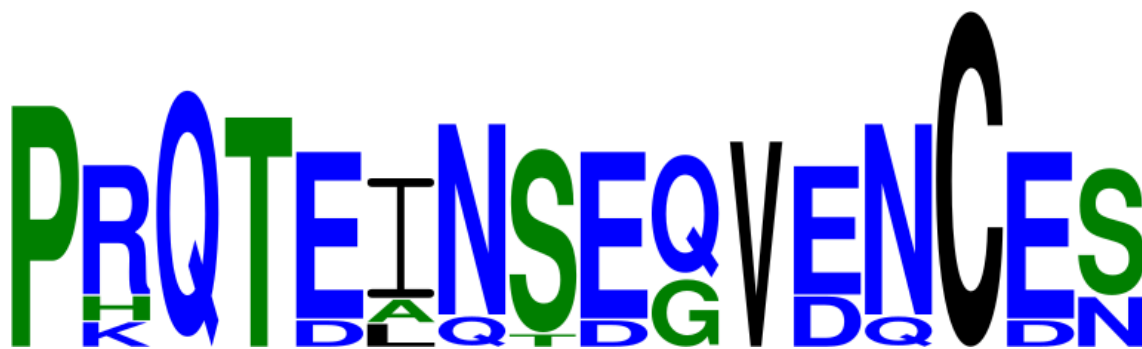


Figure 0.1.: Consensus sequence of manually engineered polypeptide augmented by point mutations with median positive substitution score (BLOSUM62).

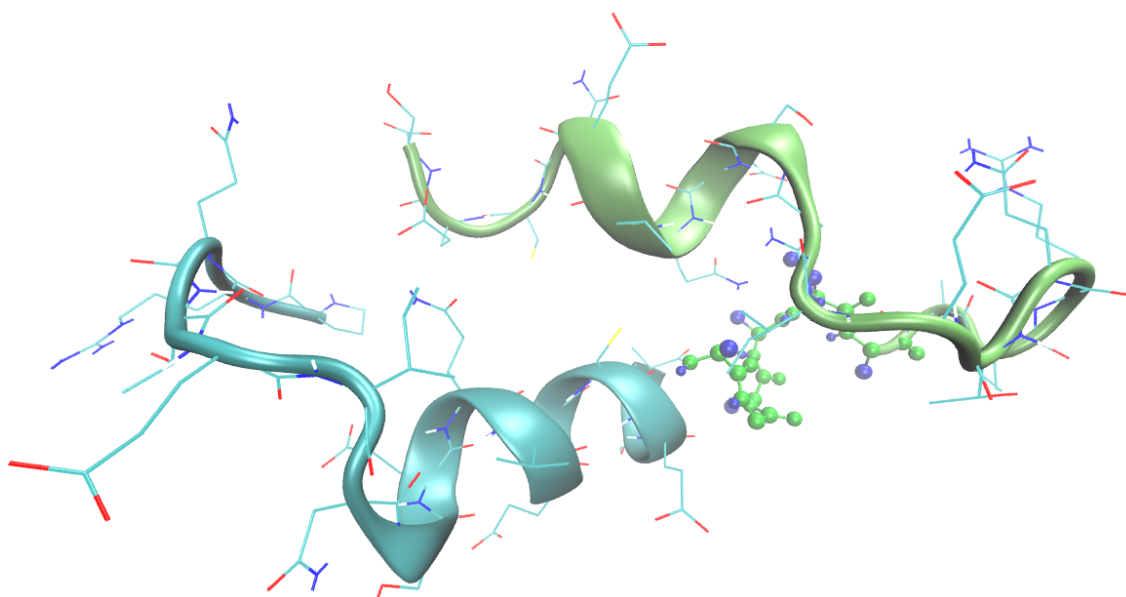


Figure 0.2.: Tertiary structure of consensus sequence dimer linked with Asf1-HistoneH3 spacer AAGAATAA [1]. Structure predicted by RaptorX [2], with distance violation 2.2204 Å, visualized with VMD [3].

Abstract

This work investigates the influence of “hubness” on the protein sequence universe. Proteins are important biomolecules that play major roles in virtually all cellular functions. A vast majority of bioinformatic experiments on proteins are performed on the level of amino acid sequences, and rely on some measure of sequence similarity. The entirety of all extinct, extant, and possible protein sequences is referred to as the protein sequence universe.

Hubness is a phenomenon of high-dimensional spaces, detrimental to a variety of machine learning tasks. Originally described in music similarity databases, hubness has since been recognized as a general “curse of dimensionality”, relevant to various application domains, including time series analysis, and natural language processing. Hubness is, consequently, hypothesized to be detrimental to biological tasks, such as sequence classification or clustering.

Hubness reduction methods are comprehensively evaluated with respect to learning performance improvements. Approximate methods with linear space and time complexity are developed, facilitating hubness analysis and reduction in large databases. The free open source software package “scikit-hubness” for Python implements these methods to make hubness analysis and reduction available to machine learning researchers and practitioners.

In addition, deep networks are used to learn protein sequence vector representations, and investigated for orthologous group inference. Finally, hubness is analyzed in sequence similarity databases and learned vector spaces using the developed methodology. While there is little evidence of improvements caused by hubness reductions, deep learning enables fast and accurate protein orthologous group inference.

In the future improved protein representations could eventually allow for accurate sublinear time sequence similarity search, which would alleviate the computational bottleneck of search in ever-growing sequence databases.

Zusammenfassung

In dieser Arbeit wird der Einfluss von „Hubness“ auf das Proteinsequenzuniversum untersucht. Proteine sind essentielle Biomoleküle, die wichtige Rollen in allen zellulären Prozessen übernehmen. Ein Großteil aller bioinformatischen Experimente zu Proteinen operieren auf Ebene der Aminosäuresequenzen, und benötigen ein Maß für Sequenzähnlichkeit. Die Gesamtheit aller Proteine wird als Proteinsequenzuniversum bezeichnet.

„Hubness“ ist ein Phänomen hochdimensionaler Räume, das eine Vielzahl von Methoden des maschinellen Lernens negativ beeinflussen kann. Das Phänomen wurde ursprünglich im Kontext von Musikähnlichkeitsdatenbanken beschrieben. Rezentere Forschung erkennt Hubness hingegen als eine generelle Eigenschaft intrinsisch hochdimensionaler Räume, die Daten in verschiedensten Anwendungsdomänen beeinflusst, darunter Zeitreihenanalyse oder die Verarbeitung natürlicher Sprache in der Computerlinguistik. Vor diesem Hintergrund wird die Hypothese aufgestellt, dass auch biologische Fragestellungen negativ durch Hubness beeinflusst werden, beispielsweise die Klassifizierung oder Clusteranalyse von Proteinsequenzen.

Eine umfassende Evaluierung von Hubness-Reduktionsmethoden zeigt auf, welche dieser Methoden besonders für die Verbesserung von maschinellem Lernen geeignet sind. Die Entwicklung approximativer Methoden mit linearer Zeit- und Platzkomplexität ermöglicht die Anwendung von Hubnessanalyse und -reduktion in großen Datenbanken. Diese und weitere Methoden werden im freien Softwarepaket „scikit-hubness“ für Python implementiert. Hubnessanalyse und -reduktion werden dadurch leichter zugängliche Bestandteile von maschinellem Lernen in Forschung und Anwendung.

Darüberhinaus wird *deep learning* genutzt, um Proteinsequenzvektorrepräsentationen und Modelle zur Zuordnung von Sequenzen zu orthologen Gruppen zu lernen. Abschließend wird mittels der entwickelten Methodologie Hubness in Sequenzähnlichkeitsdatenbanken und den gelernten Vektorräumen untersucht. Reduktion von Hubness bewirkt nur geringe Effekte in Klassifizierungs- und Clusterexperimenten. Hingegen ermöglicht *deep learning* schnelle und akkurate Zuordnung zu orthologen Gruppen.

Zukünftige Arbeit könnte weiter verbesserte Proteinsequenzrepräsentationen hervorbringen, mit denen akkurate Sequenzähnlichkeitssuche in sublinearer Zeit ermöglicht würde. Dadurch könnte das zentrale Problem des hohen Ressourcenbedarfs für Suche in immer weiter wachsenden Sequenzdatenbanken gelöst werden.

Contents

1. Introduction and background	1
1.1. Background: biology	1
1.1.1. Proteins: sequence and structure	1
1.1.2. The protein universe	2
1.1.3. Protein similarity search	3
1.1.4. Protein classification and clustering	4
1.2. Background: machine learning	5
1.2.1. Neighbors-based methods	5
1.2.2. Curse of dimensionality	6
1.2.3. Hubness	6
1.2.4. Interdisciplinary considerations	9
1.2.5. Deep learning	11
1.3. Scientific hypotheses	13
1.4. Outline	14
2. Evaluation of hubness reduction methods	15
2.1. A comprehensive empirical comparison of hubness reduction in high-dimensional spaces	16
2.1.1. Introduction	17
2.1.2. Related work	18
2.1.3. Hubness reduction methods	20
2.1.4. Evaluation	26
2.1.5. Results	31
2.1.6. Discussion	40
2.1.7. Conclusion	42
3. Development of approximate hubness reduction	47
3.1. Fast approximate hubness reduction for large high-dimensional data .	48
3.1.1. Introduction	48
3.1.2. Related work	49
3.1.3. Approximate hubness reduction methods	49
3.1.4. Evaluation	52
3.1.5. Application: Transport mode detection	54
3.1.6. Discussion	55
3.1.7. Conclusion and outlook	56
4. Implementation of hubness reduction	59
4.1. scikit-hubness: Hubness Reduction and Approximate Neighbor Search	60
4.1.1. Summary	60
4.1.2. Outlook	61

5. Protein sequence embedding	63
5.1. Introduction and background	63
5.2. Materials and methods	64
5.2.1. Protein alignment similarity scores	65
5.2.2. Proof of concept	66
5.2.3. Considerations on deep network architectures	67
5.2.4. Pipeline	68
5.2.5. Representation learning with convolutional networks	69
5.2.6. Approximate neighbor search	69
5.2.7. Data sets	70
5.3. Results and discussion	70
5.3.1. Proof of concept	71
5.3.2. Deep learning	71
5.4. Conclusion and outlook	73
6. Protein orthologous group prediction	75
6.1. DeepNOG: Fast and accurate protein orthologous group prediction	76
6.1.1. Introduction	76
6.1.2. Materials and methods	77
6.1.3. Implementation	79
6.1.4. Results	80
6.1.5. Discussion and outlook	82
6.2. Supplementary material	84
7. Hubness in protein similarity databases	89
7.1. Introduction and background	89
7.2. Materials and methods	91
7.2.1. Hubness estimation	91
7.2.2. Hubness reduction	91
7.2.3. Classification	91
7.2.4. Clustering	92
7.2.5. Data set EC: Enzymes	92
7.2.6. Data set PDB: Protein Data Bank	92
7.2.7. Data set SCOPe: Protein structure	92
7.2.8. Data set eggNOG: Orthologous groups	93
7.3. Results and discussion	93
7.3.1. Hubness estimation	93
7.3.2. Enzymes	94
7.3.3. SCOPe	94
7.3.4. EggNOG 1	96
7.3.5. EggNOG 5	96
7.3.6. PDB clustering	99
7.4. Conclusion and outlook	101
8. Conclusion and outlook	103

A. Contribution to scientific software	107
A.1. Published work	107
A.1.1. SciPy 1.0: fundamental algorithms for scientific computing in Python	108
A.2. Unpublished work	123
A.2.1. scikit-learn: machine learning in Python	123
A.2.2. thunderSVM: fast support vector machines on CPUs and GPUs	123
A.2.3. phenotrex: microbial phenotypic trait prediction	123
B. Additional projects	125
B.1. Protein unfolding	125
B.2. Microbial longitudinal division	126
C. Academic teaching	129
Bibliography	131
Academic journals and conference proceedings	131
Book sources	150
Online resources	151

1. Introduction and background

Cellular organisms are built from biomolecules, commonly divided into macromolecules including lipids, polysaccharides, nucleic acids, or proteins, and a wide range of small molecules [4, p. 3]. Proteins consist of one or several variable-length amino acid chains that fold into particular three-dimensional structures. The resulting conformation (or tertiary structure) determines the protein’s function. Proteins provide a wide range of functionality in cellular and noncellular organisms, including metabolic catalysis, cell signaling, intracellular transport, and also sustain structure [4, p. 61]. Consequently, proteins are central objects of interest in molecular biology and studied widely in computational biology with particular focus on primary structures, that is, protein sequences.

Machine learning refers to a vast variety of computer algorithms that adapt their behavior according to *experience*, that is, observed data [5, p. 4]. Data-driven learning systems often achieve better predictive performance than expert systems, and define the state-of-the-art in many domains, including computer vision, natural language processing, or information retrieval. Computational biology makes heavy use of machine learning, for instance in gene prediction [6], [7], protein structure prediction [8], [9], in mining gene co-expression patterns [10], or for modeling complex interactions in biological or ecological networks [10], [11].

The following sections outline the background in both biology and machine learning required for the research presented in this thesis. The work involves the conception, improvement, development, and implementation of machine learning algorithms addressing specific issues of high-dimensional spaces, and their application in protein sequence analysis in particular.

1.1. Background: biology

Proteins are essential components of all cellular organisms, spanning the complete tree of life, including all domains *Eukarya*, *Bacteria*, and *Archaea*, as well as noncellular structures displaying some properties of life, such as viruses. Understanding the structure, function, and evolution of proteins is, thus, crucial to the understanding of life.

1.1.1. Proteins: sequence and structure

Proteins are built from one or several subunits. Each subunit consists of a polymer of amino acid residues, also known as polypeptide. The standard set of proteinogenic amino acids comprises 20 different residues, subject to post-translational modifications. Two additional residues are used by a limited number of organisms. The polypeptide sequence is determined by the nucleotide sequence of the coding gene and the *genetic code*, which maps nucleotide triplets to single amino acids. Protein biosynthesis follows the “central dogma of molecular biology” [12]: Genetic DNA

is transcribed to messenger RNA by an RNA-polymerase and further translated to polypeptides at the ribosome subject to various additional processes, such as post-transcriptional and post-translational modifications, or translocation.

The native three-dimensional conformation of a protein is encoded in its sequence [13, Section 3.6], and the nascent polypeptide chain already starts the process of protein folding [14]. The primary structure (sequence) folds into local secondary structures, such as α -helices, or β -sheets, maintained by hydrogen bonds. Proteins are further folded into their tertiary structure that is determined by global interactions and additional physico-chemical interactions, including disulfide bonds or salt bridges. Higher-order structures refer to the arrangement of subunits in protein complexes, or protein interactions.

DNA sequencing has become cheap with the advent of high-throughput “next generation” sequencing technology [15]. Large quantities of protein sequences are produced by *in silico* gene prediction and translation from DNA sequences. However, resolving protein structure remains comparatively resource-intensive and difficult, using *wet lab* techniques such as X-ray crystallography, nuclear magnetic resonance spectroscopy, and, increasingly, cryo-electron microscopy [16]. When homologous templates are available, comparative modeling approaches allow to computationally determine structures to reasonable quality. *Ab initio* tertiary structure prediction, however, remains an actively researched area [9]. As a result, resolved structures are available for less than one percent of known protein sequences.¹

Protein sequences determine the structures, which in turn determine functions [13, Chapter 3]. Not only do sequences, thus, determine function in transitivity, but they are also available abundantly. It is, therefore, not surprising that data-driven methodology in bioinformatics more often relies on protein sequences than structures. Protein sequences constitute the central object of study in this thesis.

1.1.2. The protein universe

The entirety of all proteins is referred to as *protein universe* [17]. Lacking a strict definition, the term may encompass all extant and/or extinct or theoretically possible proteins [18]–[20]. Information concerning the protein universe is obtained at high rates, owing primarily to the massive stream of protein sequences generated from genomic and metagenomic DNA sequencing. While there is a moderate increase of single-domain architecture families, the number of multi-domain architecture families grows rapidly, where structural information is lacking for the majority of sequence profiles, and a considerable fraction of sequences do not match any profile [19]. The protein universe is described as a continuous-discrete space, with at least parts allowing for “interpolation” of structures [21]–[23], hinting at a large unrealized part of the protein universe. Furthermore, computational protein design uncovers large proportions of the possible repeat protein space to not be covered by existing proteins. That is, natural selection has so far only sampled a fraction of the protein universe, and sampling of new regions could be performed by engineering novel repeats [20].

¹Crude conservative estimate: $163949[\text{structures}] \div 39962345[\text{sequences}] = 0.41\%$, numbers derived from clustered sequences (UniRef50, <https://www.uniprot.org/uniref/?query=&file=identity:0.5>), and all structures available in the PDB (<https://www.ebi.ac.uk/pdbe/>, “Latest archive statistics”), both retrieved 2020/05/15.

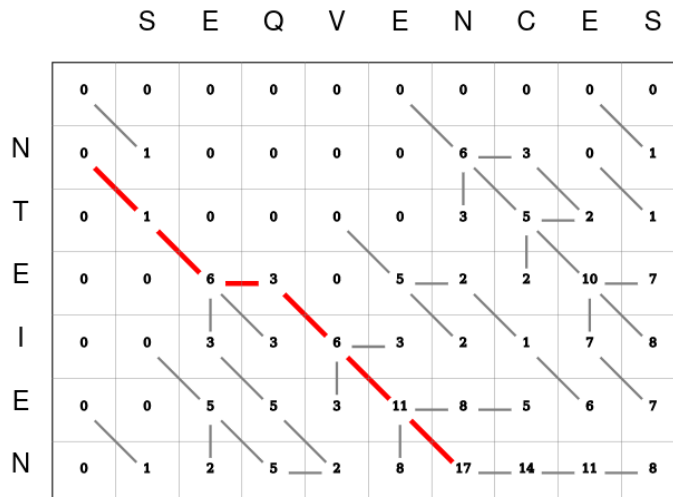


Figure 1.1.: Smith-Waterman alignment of two sequences. Red path indicates optimal local alignment with score 17 (BLOSUM62, linear gap cost 3).

Only a small fraction of the protein universe has been experimentally assessed, with the large rest remaining *hypothetical* proteins. Generating knowledge requires integrating the available information about the protein universe, and extensive computational characterization. Inference of molecular functions and biological roles exploits common ancestry of protein sequences. Evolutionary relationship is modeled in phylogenetic trees and databases of orthologous groups², or may be deduced from similarity scores obtained from sequence alignments, and evaluated for statistical significance considering some background model [26].

1.1.3. Protein similarity search

Sequence alignment is the corner stone of bioinformatics. Several alignment techniques reside among the most frequently used scientific methods [27]. Significantly similar sequence patterns strongly indicate common ancestry, which in turn allows for insights into evolutionary, structural, and functional relationships [28].

Optimal alignments are computed with the Needleman-Wunsch algorithm [29], and the Smith-Waterman algorithm [30] for global and local alignments, respectively. Both algorithms use dynamic programming to solve the optimization problem, and require substitution tables defining match and mismatch scores for all letter pairs, as well as the gap penalty. Figure 1.1 outlines the procedure. Multiple substitution schemes are available [31], [32]. The algorithm requires $\mathcal{O}(m_1 m_2)$ time and $\mathcal{O}(\min(m_1, m_2))$ space, where m_1, m_2 are the lengths of the aligned sequences, when using optimizations [33]–[35]. Implementations keep being updated to current hardware generations for improved performance [36]–[40]. The algorithm can be extended for multiple sequence alignments, where the NP-complete optimal solution [41] is prohibitively expensive for even moderate numbers of sequences.

Alignment heuristics are commonly applied to reduce the computational cost [42]–

²Types of homology: Orthologous sequences evolve from common ancestors by speciation events. In contrast, paralogs evolve from common ancestors by gene duplication [24]. The *ortholog conjecture* postulates that orthologs retain functional similarity, while paralogs evolve to different functions [25].

[46], which has become a bottleneck in large-scale analyses. Alignment-free methods are a computationally efficient alternative [47]–[49], typically trading runtime against sensitivity.

Protein similarity search may use scores obtained from alignment-based or alignment-free methods. A query sequence is compared against all sequences in a database of interest, for example, the **nr** database of non-redundant protein sequences³, and best hits according to this score, or a derived measure, such as e-values, are reported. This procedure may be regarded as *nearest neighbor* search using a similarity measure based on any of the methods outlined above (see also Section 1.2.1 below). Similarity search may also use structural alignments [50], [51]. This approach is limited by the low availability of protein structures, and, therefore, not further discussed here.

Protein sequence similarity search is time-consuming, and proportionally increasingly expensive with growing database sizes. The Similarity Matrix of Proteins (SIMAP) database [52] provides precomputed sequence alignments. In its latest iteration, SIMAP2 [53] provides Smith-Waterman alignment scores for all pairs of nearly 30 M protein sequences from several thousand fully assembled genomes. SIMAP2 is a sparse matrix with all scores below a permissive cutoff value set to zero⁴, and density below 0.005. SIMAP provides a large sample of the protein sequence similarity space, and serves as a central resource for the work presented in this thesis.

1.1.4. Protein classification and clustering

Assuming the similarity space spanned by protein sequences and a similarity measure as discussed above carry *semantic* meaning, knowledge transfer may be conducted by exploiting the geometry of this space. Molecular function and biological processes may then be readily inferred for a query sequence from the best hit among the indexed sequences in a database, or by more elaborate methods. Inference or classification is impaired, however, by non-uniformly densely populated areas in the protein space, varying evolutionary rates in different families, and complex evolutionary scenarios. Figuratively, (i) the protein universe comprises dense “galaxies” as well as sparse intergalactic space [17], with (ii) an unknown fraction of space remaining uncharted to date [19], and (iii) different parts of the universe expand with different velocities [54]. All of these aspects reduce the quality of point estimates in functional inference.

Numerous approaches propose to improve upon straightforward best hit methods. Multiple general clustering approaches were applied to partition the protein universe, e.g., [55]–[59]. Additional clustering strategies were conceived to construct clusters of orthologous groups. Notable examples include COG [60], KEGG orthology [61], OrthoMCL [62], OMA [63], and eggNOG [64]. Alternative approaches to orthology inference rely on phylogenetic trees [65].

Additional challenges arise from the large number of objects in the protein universe, its potentially high-dimensional realizations, highly variable protein (sequence) sizes, or spurious similarities. One particular challenging aspect of the protein sequence similarity space is the *hubness* phenomenon (see Section 1.2.3 below), that

³**nr** database: <ftp://ftp.ncbi.nlm.nih.gov/blast/db/>, accessed 2020/05/28

⁴SIMAP2 whitepaper with alignment procedures: <http://cube.univie.ac.at/sites/cube/files/simap-2.0-alignment.pdf>, accessed 2020/05/28

may impair classification and clustering of protein sequences. This thesis investigates the effects of hubness on the protein sequence universe.

1.2. Background: machine learning

Multiple types of learning exist, and one important type is arguably *learning from experience*. Machine learning may then be described as computer systems that adapt their programs based on previously observed (training) data [66, page 4]. Successful learning requires *generalization*: Only when previously unseen (test) data is handled appropriately has the system truly learned, and provides functionality beyond simple database search.

Machine learning algorithms are commonly categorized by their use of training *labels* (targets, ground truth). In supervised learning training data come with associated target vectors. Classification and regression are two canonical examples of supervised learning scenarios. In unsupervised learning, patterns or groups of objects are discovered without using ground truth labels. Clustering similar objects, determining data distribution with density estimation, or visualizing high-dimensional data in lower dimensional embedding spaces are unsupervised tasks [5, pp. 2–3]. Self-supervised learning does not require target labels, but formulates a proxy learning task solved without external supervision [67] in order to obtain semantically meaningful representations of objects. Proxy tasks include predicting masked words in natural language processing [68], or assessing similarity subject to data augmentation (color distortion, rotation, random crop, etc.) in computer vision [69]. Other types of learning, such as semi-supervised learning, or reinforcement learning are not considered or only briefly touched upon in this thesis.

1.2.1. Neighbors-based methods

Nearest neighbor search is the optimization problem of finding the closest object, or the k closest objects in a spatial database of cardinality N w.r.t. some not necessarily metric (dis)similarity measure [70, pp. 459–460]. Optimal solutions are found by brute force linear search comparing a query against all indexed objects in $\mathcal{O}(N)$ time, or space partitioning methods, such as k -d trees [71], ball trees [72], or R* trees [73], which can reduce average query time to $\mathcal{O}(\log N)$. Approximate nearest neighbor search can provide superior computational efficiency, when some retrieval error ϵ is allowed. Notable examples include locality-sensitive hashing (LSH, [74]), and hierarchical navigable small-world graphs (HNSW, [75]).

Neighbor search is used in the nonparametric k -nearest neighbor method (k NN) for classification, regression, or density estimation [5, pp. 124–127]. Classification can be performed as a majority vote by the k NN:

$$p(C_K|x) = \frac{p(x|C_K)p(C_K)}{p(x)} = \frac{k_K}{k} \quad (1.1)$$

gives the posterior probability of membership in class C_K for object x considering the hypersphere around x containing the k -nearest neighbors (the k -neighborhood), where k_K objects belong to class C_K [5, eq. 2.256]. The nearest neighbor classifier ($k = 1$), therefore, simply assigns the label(s) of the closest indexed object to the

query. The error rate of 1-NN is known to not exceed 2ϵ when $N \rightarrow \infty$, where ϵ is the minimal achievable error obtained from an optimal classifier [5, p. 127].

A major limitation of k NN is its *lazy* nature, which requires storing all training objects in the index. This results in linearly increasing computational cost and storage requirements, and poses additional problems, for example in privacy-preserving data mining [76].

The k -nearest neighbor classifier is a versatile method widely applied in machine learning and pattern recognition. While often specialized methods outperform k NN, it serves as a strong baseline in many areas, including time series classification, where many algorithms fail to improve upon 1-NN with dynamic time warping distances (see Section 1.2.2) [77]–[79]. In biological sequence analysis, (approximate) nearest neighbor search/classification with domain-specific similarity measures is applied ubiquitously, for instance, in functional annotation [80, pp. 706–708] or taxonomic inference [4, p. 380]. The method is colloquially better known as *blasting*.

1.2.2. Curse of dimensionality

Objects are commonly represented by numerical vectors in d -dimensional space, where d is the number of features (properties, independent variables). Learning in high-dimensional spaces poses multiple specific problems that are commonly referred to as the “curse of dimensionality” [81]. One central insight concerns data density: Volume increases exponentially with increasing dimensionality, or conversely, the volume of unit hyperspheres shrinks when considering increasing numbers of features. As a consequence, machine learning algorithms require more data in high dimensions [66, p. 18]; exponentially more data on average in the worst case, which is especially problematic for lazy learners, such as the above described k NN.

Distance concentration

Another well-known aspect of the dimensionality curse is *distance concentration*, or concentration of measure, which describes vanishing relative contrast with increasing dimensionality. That is, distances in all pairs of objects in high-dimensional spaces converge to a constant value [82]. The lack of contrast between near and distant neighbors is detrimental to high-dimensional data mining, search, retrieval, and learning. Various measures are known to concentrate, in particular ℓ^p norms, such as the widely used Euclidean or Manhattan distances [83]. On the other hand, there is indication of concentration also for more complex dissimilarity measures [84], such as those derived from *dynamic time warping* (DTW, [85]). DTW is conceptually identical to alignment algorithms used in biological sequence analysis, which are also based on dynamic programming. The algorithms differ in measuring similarity, where DTW employs standard distance measures between vectors (for example, the l^2 norm), and biological sequence alignment algorithms use substitution tables of nucleotides or amino acids instead.

1.2.3. Hubness

Hubness is an additional phenomenon of the curse of dimensionality. Albeit less well known, it is related to the concentration of distances. While distances are indistinguishable in the limit, in high but finite dimensionality neighborhood relationships

retain some meaning, but are distorted. Hub objects are on average closer to the data center, and, thus, occur more frequently among the nearest neighbors of other objects than expected. Antihubs are far from other objects, and are hardly ever or never found among any nearest neighbor lists. Learning in high dimensions is impaired, when distance spaces are semantically distorted by hubness.

For a motivating example, consider a music recommendation system employing a hubness-prone similarity measure and neighbor search that selects the next songs in a playlist based on the current song. Hub songs are recommended frequently, while antihub songs are never recommended. This negatively affects user experience, when hub recommendations do not fit semantically. In addition, artists that (unknowingly) produce antihub songs may suffer financial losses.

The hubness phenomenon

The hubness phenomenon was indeed first described in the context of audio similarity measures. Aucouturier and Pachet coined the term for their observation of scale-free false positive distributions in certain audio models [86], and hypothesized the phenomenon to generalize to a large class of pattern recognition algorithms. The authors suggest observing the k -occurrence $O^k(x)$ to describe hubness, which describes how often an object $\mathbf{x}$ resides in the k -neighborhood of other objects in the data set w.r.t. some distance measure (not necessarily metric):

$$O^k(x) = \sum_{y \in X} \mathbb{1}_{N^k(y)}(x), \quad (1.2)$$

where x and y are objects from a data set X , and the indicator function $\mathbb{1}_{N^k(y)}(x) = 1$ if $x \in N^k(y)$, otherwise 0, with $N^k(y) \subseteq X \setminus y$ (the k -nearest neighbors of y). The measure $O^k(x)$ is otherwise known as the *reverse neighbor count*, or the *indegree* given a directed k -nearest neighbor graph.

Hubness was quickly recognized as a general property of intrinsically high-dimensional vector spaces in [87]. The study provides extensive theoretical and empirical insights on the origin of the hubness phenomenon as well as its implications on supervised, semi-supervised, and unsupervised learning in synthetic and real-world data using various distance measures. Hubs are recognized as objects close to the center of unimodal data, or close to the mean of individual components, when multimodal data are described as mixtures of unimodal distributions. Common dimensionality reduction methods do not improve learning. The authors propose to characterize the k -occurrence distribution by the third standardized moment:

$$S^k = \frac{\mathbb{E} \left[(O^k - \mu_{O^k})^3 \right]}{\sigma_{O^k}}, \quad (1.3)$$

where μ and σ indicate mean and standard deviation, respectively. High hubness regimes are characterized by distributions strongly skewed to the right. The “ k -skewness” S^k is used throughout this thesis as a measure for hubness.

The origin of hubness in intrinsic high-dimensionality was scrutinized in [88], where hubness is suggested to arise from density gradients or bounded sampling. With increasing dimensionality the ratio of boundary size and its encapsulated volume increases exponentially, explaining the increasing emergence of hubs.

Additional theoretical contributions provide an extreme-value-theoretic foundation of a notion of intrinsic dimension that allows to characterize inlierless, outlieriness, discriminability and hubness [89], [90]. The k -occurrence distribution is precisely characterized in [91], where a closed form for k -occurrence in finite high-dimensional vector spaces and related probability distributions are introduced.

Hubness reduction

After hubness was demonstrated to be a general phenomenon of high-dimensional data and detrimental to various tasks, methods for its mitigation were sought. Several hubness reduction methods are introduced below, most of which provide *secondary distances*. That is, they operate on (arbitrary) dissimilarity spaces instead of vector spaces, and return another, transformed dissimilarity space.

Mutual proximity (MP) is introduced and shown to effectively reduce hubness in [92]. In hubness-prone spaces, many neighborhood relations are asymmetric, that is, $x \in N^k(y)$, while $y \notin N^k(x)$. MP transforms distances to joint probabilities of two objects being nearest neighbor to each other. The transformation enforces symmetric neighborhoods: MP distances are small only if both individual nearest neighbor probabilities are small. MP is shown to improve learning in many benchmark data sets, and demonstrated to improve recommendation quality and song reachability in the FM4 Soundpark⁵ music database. The study also repurposes “local scaling” (LS) and the “non-iterative contextual dissimilarity measure” (NICDM), two methods initially developed for improving spectral clustering [93], and image search [94], respectively. Both achieve results in hubness reduction comparable to mutual proximity.

The shared near neighbors (SNN) similarity measure also considers mutual neighborhood information by counting the number of objects present in both k -neighborhoods of two queries of interest [95]. SNN has recently been investigated for its capabilities in alleviating the curse of dimensionality [96] and hubness [97] with mixed results. The simhub measure combines SNN with supervised and unsupervised hubness information [98], [99].

Laplacian-based kernels can reduce hubness in some data sets [100] by applying inner product similarity scores that evaluate all queries to equal similarity to the centroid. Along the same line, centering data (CENT) removes spatial centrality of objects that might otherwise emerge as hubs [101]. All objects in a centered data set have exactly zero dot-product similarity to the centroid. Centering has the particular advantages of applicability to vector data (as compared to distance spaces only), and particular low cost (subtraction of the data mean). The approach is, however, limited to specific similarity measures and tasks. It is, for example, not applicable in combination with Euclidean spaces, in which centering does not change distances, or neighborhood relations.

Localized centering (LCENT) is proposed to reduce hubness in large sample data [102]. It generalizes the idea of centering to multimodal data, aiming to remove spatial centrality towards local centroids instead of the global centroid. Both centering variants are shown to improve learning in natural language processing.

A similar approach for (squared) Euclidean spaces is derived in [103], and argued to reduce hubness by flattening the density gradient. Both a global (DisSim^{Global},

⁵FM4 Soundpark: <https://fm4.orf.at/soundpark/>, accessed 2020/05/28

DSG) and a local (DisSim^{Local}, DSL) variant are presented, that eliminate spatial centrality toward global and local centroids, respectively.

Fractional distances (ℓ^p quasinorms with $p < 1$) have long been investigated for possible advantages over full metric distances with respect to the curse of dimensionality in general [82], and hubness specifically [104]. (Very recently, however, such quasinorms were reported to not overcome the curse of dimensionality [105].) A data-dependent dissimilarity measure is introduced in [106] and argued to feature advantageous properties with respect to hubness.

Hubness-affected methods and tasks

The k -nearest neighbor classifier is the most thoroughly investigated algorithm with respect to hubness. While there exist many general hubness reduction methods as outlined above, there have been efforts to integrate knowledge about the phenomenon directly into learning algorithms. Examples include hubness-aware Bayesian k NN [107], hubness-based fuzzy measures to improve k NN [108], and hubness-aware voting frameworks for data sets with highly imbalanced classes [109], or label noise [110].

Time-series classification with dynamic time warping is shown to suffer from the hubness phenomenon in [84], while hubness-aware approaches are proposed for time-series classification, instance selection, and feature construction in [111].

The topic of hubness has attracted less attention in regression, though there are few studies reporting for example, improved kernel regression due to hubness reduction [112], or drug–target interaction prediction with hubness-aware regression [113].

In unsupervised learning, the influence of hubness on clustering has been investigated in several works. Analogously to supervised learning, hubness reduction is found to improve k -medoid clustering [114]. On the other hand, there are approaches that do not consider hubness a necessarily harmful property of high-dimensional spaces, but propose k -means-like hubness-based clustering algorithms utilizing hub objects as prototype clusters [115], [116].

Unsupervised distance-based outlier detection is demonstrated to suffer from the presence of antihubs, and, therefore, to profit from hubness reduction [117].

Most research on hubness considers the case of exact neighbor search. However, the phenomenon may also influence approximate neighbor search methods. First insights into this connection are obtained for the NN-descent algorithm [118] in [119], [120].

1.2.4. Interdisciplinary considerations

Hubness is a general phenomenon of high-dimensional spaces. As such, it is not limited to specific application domains. Instead, implications of hubness have been investigated in various domains, each having specific requirements and pitfalls shedding light on different aspects of the hubness phenomenon. Considering hubness in various domains, therefore, allows to gain additional insights into the topic.

Music information retrieval

Hubness has first been observed and described in the context of music information retrieval. Consequently, a vast body of research output is available in this domain.

Early domain-specific approaches to overcome the hubness phenomenon in music similarity retrieval are described in [121], where combining features results in reduced hubness. A meta-analysis of the MIREX competition in “Audio Music Similarity and Retrieval” reveals a strong negative correlation between hubness and perceptual quality among seventeen different algorithms [122]. Mutual proximity is shown to improve the situation in the same study, and is subsequently demonstrated to improve visualization of music similarity spaces [123], and reachability in music recommendation graphs [124].

Integration of hubness analysis is suggested for multimedial retrieval in various domains, such as music, text, or image when distance measurements in high-dimensional spaces are performed [125]. Along the same line, rating prediction accuracy in neighborhood-based collaborative filtering is improved by mutual proximity [126].

Natural language processing

Natural language processing (NLP) is another domain that has seen a significant research effort regarding the hubness phenomenon. For example, cross-lingual document retrieval may profit from hubness-aware algorithms [127], and several hubness reduction methods specific to the text domain have already been introduced in Section 1.2.3, p. 8.

Zero-shot learning refers to learning vector-based embeddings of text or other objects for nearest neighbor search. Hubness analysis may improve zero-shot learning in NLP, computer vision, and cross-field mappings [128]–[130]. A “reconstructed similarity” measure for embeddings learned with generative adversarial networks improves the quality of bilingual dictionaries for word translation [131].

Other areas

As already indicated in the sections above, hubness has also been observed in other domains, such as computer vision. Considering the hubness phenomenon has first been shown to improve object recognition accuracy in [132]. Embedding spaces for zero-shot learning in computer vision are investigated with respect to the hubness phenomenon in [133].

Considering security, the presence of hubs and antihubs may confound speaker verification [134], and hubness reduction is proposed to thwart shilling attacks⁶ in recommender systems in [135].

Hubness is linked to the intrinsic dimensionality of data. This association is exploited in [136], [137] to estimate related measures used in hyperspectral unmixing applications in remote sensing.

Hubness-aware learning was proposed in the biomedical context for gene expression data classification [138], [139] or disease rating scale estimation for neurodegenerative disorders [140].

⁶Shilling attack: “In a recommendation system where everyone can give the ratings, people may give lots of positive ratings for their own items and negative ratings for their competitors. It is often necessary for the collaborative filtering systems to introduce precautions to discourage such kind of manipulations.” [wikipedia, https://en.wikipedia.org/wiki/Collaborative_filtering#Shilling_attacks, last retrieved 2020/05/12]

The term “hubness” is also used in other contexts with slightly different definitions. For example, “hubness” is also used to describe a general notion of node centrality, or inbetweenness in graphs in [141], [142]. Throughout this work, “hubness” will only refer to the specific phenomenon of popular neighbors in intrinsically high-dimensional spaces.

1.2.5. Deep learning

Besides neighborhood-based methods, deep learning is the second class of machine learning algorithms extensively used in this work. The term is often used interchangeably with artificial neural networks. Mathematical descriptions of biological neurons date back to the work of McCulloch and Pitts in the 1940s [143]. Building upon these models, the perceptron [144] marks the first artificial neuron and foundation of artificial neural networks. Initial limitations, such as the inability to solve the XOR problem [145], were overcome by introducing multiple layers of neurons. Multi-layer feed-forward neural networks are typically trained by error backpropagation [146].⁷ Deep networks with at least one hidden layer are known to be universal approximators to continuous functions on $\mathbb{R}^d$ (under mild assumptions) [147]–[151]. While the universal approximation theorem proves the capability to represent many functions, it does not provide guidelines on how to train the parameters of a corresponding model. Deep learning has found widespread use in research and industry since the 2010s. Multiple technological advancements contributed to this development, including the use of massively parallel compute hardware, including GPUs [152] and specialized processors, the availability of large-scale data sets [153], various architectural improvements to deal with problems arising in deep networks with many layers [154]–[157], and software frameworks [158]–[160] providing convenience function, such as automatic differentiation.

Generalized linear models comprise M linear combinations of non-linear basis functions ϕ applied to data x such that

$$y(x, w) = f \left(\sum_{j=1}^M w_j \phi_j(x) \right), \quad (1.4)$$

where w denotes a set of coefficients [5, eq 5.1]. Deep networks extend this framework by having ϕ depend on tunable parameters. Models are constructed by a series of transformations, where in the first layer activations a_j are calculated as

$$a_j = \sum_{i=1}^d w_{ji}^{(1)} x_i + w_{j0}^{(1)}, \quad (1.5)$$

the sum of M linear combinations applying weights w_{ji} and biases w_{j0} to the data x . Nonlinear (almost everywhere) differentiable activation functions $h(\cdot)$ are applied consequently as

$$z_j = h(a_j), \quad (1.6)$$

where sigmoid or ‘tanh’ functions have traditionally been used for h , and more recently, the ReLU function and numerous variants of it are applied more commonly,

⁷Since backpropagation is no plausible mode of learning in biological neural networks, many researchers, including the author, prefer the term “deep network”, without reference to neurons.

where $h_{ReLU}(x) = \max(0, x)$ [161]. The obtained outputs are called hidden units, and can be further combined:

$$a_k = \sum_{j=1}^M w_{kj}^{(2)} z_j + w_{k0}^{(2)}. \quad (1.7)$$

If this is the last layer in a two-layer network then $k \in [1, K]$ and K is the number of output units. A final activation function is applied to the activations a_k depending on the task, for example, the identity function in regression, or the softmax with logistic sigmoid functions [5, p. 228]:

$$y_k = \sigma(a_k) = \frac{1}{1 + \exp(-a_k)}. \quad (1.8)$$

Alternatively, additional hidden layers can be incorporated applying the same structure of linear combinations and subsequent nonlinear transformations as outlined in the equations above.

If all nodes in the $(l-1)^{\text{th}}$ layer are connected to all nodes in the l^{th} layer of the network by nonzero weights, the l^{th} layer is referred to as *fully connected*, *dense*, or *linear* layer.

It is not always desirable, however, to densely connect nodes. *Convolutional* layers feature several beneficial properties [162, pp. 324–330]:

- Sparse interactions require fewer operations, fewer parameters, thus less memory, and improve statistical efficiency. For example, in computer vision it is often sufficient to consider local relations between individual pixels rather than all global relations.
- Parameter sharing allows to reuse weights corresponding to one kernel (see below) at all positions in the data rather than learning separate weights for all of them. For example, short motifs are often highly informative in biological sequence analysis. A kernel related to one particular motif can be applied to complete sequences with identical weights, as the motif is independent of any long-range interactions.
- Equivariance of translation refers to the property of identical but shifted outputs when features are shifted in the input. For example, video cameras should report detected faces in the corner of a photo as prominently as in the center; or ribosomal binding sites should be detected mostly independent of its position on a chromosome.

These properties are obtained by using small kernels that are applied in a sliding-window approach to the input. Kernels are often referred to as filters, and their outputs as feature maps. In computer vision, convolutional networks have been demonstrated to extract features of increasing complexity. While filters in early layers are sensitive to simple patterns, such as straight lines of specific orientations, filters in later layers may respond to more complex geometric patterns, and filters in late layers would detect complex features of interest, for example, faces. Convolutional layers are typically combined with downsampling in *pooling* layers that provide summary statistics of local outputs [162, p. 330], contributing to translation invariance, and increasing the *receptive field*.

Deep learning requires additional building blocks, such as loss functions to guide the training (for example, cross-entropy in classification), optimization strategies to tune the network weights (usually variants of stochastic gradient descent), regularization strategies to avoid overfitting (for instance, dropout), or schedulers that adapt the central hyperparameter during training (the *learning rate*). For brevity, the reader is referred to the canonical text book of Goodfellow, Bengio, and Courville [162] for details.

Deep learning has been attracting wide-spread attention in computational biology and medicine since the mid-2010s [163]–[167]. Notable applications include expert-level disease classification for skin cancer [168], retinal diseases [169], and arrhythmia [170] as well as the current state-of-the-art in protein *ab initio* structure prediction [171]. This thesis presents approaches to supervised and self-supervised deep learning for protein sequences.

1.3. Scientific hypotheses

Preliminary results suggest protein sequence similarity spaces are prone to excessive hubness. In this work, these results are evaluated in depth, and the influence of hubness on supervised and unsupervised learning in the protein universe is investigated, testing hypotheses outlined below. Hubness is detrimental to various learning tasks in high-dimensional spaces, including classification, clustering, visualization, and others. Learning improvements are achieved by hubness reduction in multiple areas, such as music information retrieval, or natural language processing. SIMAP2, a large database of protein sequence similarities, is suspected to exhibit high hubness. On the basis of these conditions, the central hypothesis of this thesis is proposed:

Hypothesis H₁: Hubness reduction improves (A) supervised and (B) unsupervised learning in the protein sequence universe.

Prior to testing Hypothesis H₁, multiple research questions need to be addressed, and method development as well as software development are required, involving the following three hypotheses:

Hypothesis H₂: Different hubness reduction methods improve learning performance equally well.

Hypothesis H₃: Approximate hubness reduction methods with linear complexity in time and space are conceivable, and improve learning to an extent close to exact hubness reduction.

Hypothesis H₄: Protein sequences can be embedded in vector spaces such that Euclidean distances preserve similarity relations from sequence alignments.

Losely related to the hypotheses outlined above, and motivated from observations in experiments performed during investigations for testing these hypotheses, an additional hypothesis is formulated:

Hypothesis H₅: Deep learning enables fast and reliable protein orthologous group inference.

Additional, unrelated hypotheses are proposed for side projects briefly described in Appendix B:

Hypothesis H₆: Unsupervised outlier detection methods identify erroneous experiments in protein unfolding assays.

Hypothesis H₇: Longitudinally dividing bacteria are recognized in microscopic images by deep convolutional network-based computer vision.

1.4. Outline

This dissertation is structured as follows. Chapter 2 presents a large-scale empirical assessment of multiple hubness reduction methods, rejecting Hypothesis H₂, and providing guidelines on selecting hubness reduction methods. Chapter 3 describes the development and evaluation of approximate variants of the most successful hubness reduction methods identified before. Efficient and effective approximate hubness reduction in various vector data sets confirms Hypothesis H₃. Chapter 4 reports on the implementation of exact and approximate hubness reduction algorithms in the free open source software package `scikit-hubness`. The package is employed for hubness-related experiments in large data sets in this work, and enables researchers and machine learning practitioners to investigate hubness in their data easily. Chapter 5 introduces ideas to learning protein sequence vector embeddings that allow for hubness reduction and fast similarity search in large protein databases, thereby testing Hypothesis H₄. Chapter 6 demonstrates efficient and reliable protein orthologous group inference by deep learning, confirming Hypothesis H₅. The Python package `deepnog` allows users to assign their protein sequences to orthologous groups. Chapter 7 assesses multiple protein classification and clustering tasks, and the influence of exact and approximate hubness reduction on learning performance in these tasks, building upon the results presented in Chapters 2–5, testing the central Hypothesis H₁, and concluding the main body of this work.

Additional material is presented in the Appendices. Contributions to scientific software packages are described in Appendix A, including general scientific computing software, such as SciPy, and highly specialized tools like `phenotrex`. Selected ongoing research side projects are presented in Appendix B with preliminary results on Hypotheses H₆ and H₇. The dissertation is concluded with some remarks on the author’s efforts in academic teaching in Appendix C.

2. Evaluation of hubness reduction methods

This doctoral thesis deals with the *hubness* phenomenon in large protein similarity networks. Particular interest is in potential benefits from hubness reduction. At project start, several hubness reduction methods had already been published. However, no comprehensive evaluation of all available methods had been conducted. Consequently, the first work package in this project was a stringent evaluation. All available methods are implemented in the “Hub-Toolbox” [172], and evaluated in a large variety of tasks. The results of this comparison are analyzed employing rigorous statistics, rejecting Hypothesis H₂. Finally, methods with most encouraging results with respect to hubness reduction and classification performance are selected for the next work package (development of approximate hubness reduction in linear space and time, see Chapter 3).

Preliminary version A preliminary version of this work was presented at the International Conference on Artificial Neural Networks (ICANN), 2016, Barcelona, Spain, and published in the conference proceedings:

Feldbauer, R., Flexer, A. *Centering Versus Scaling for Hubness Reduction*. In: Villa A., Masulli P., Pons Rivero A. (eds) Artificial Neural Networks and Machine Learning – ICANN 2016. ICANN 2016. Lecture Notes in Computer Science, vol 9886. Springer, Cham. Print ISBN 978-3-319-44777-3. DOI: https://doi.org/10.1007/978-3-319-44778-0_21

Full version A substantially expanded version was subsequently published as:

Feldbauer, R., Flexer, A. *A comprehensive empirical comparison of hubness reduction in high-dimensional spaces*. Knowl Inf Syst 59, 137–166 (2019). DOI: <https://doi.org/10.1007/s10115-018-1205-y>

For brevity, only the latter journal paper is included in this dissertation.

Contributions AF and RF designed the study. RF implemented the software, conducted the experiments, and analyzed the results. RF wrote the manuscript with guidance from AF. AF supervised and supported all steps.

A comprehensive empirical comparison of hubness reduction in high-dimensional spaces

Roman Feldbauer¹  · Arthur Flexer¹ 

Received: 16 August 2017 / Revised: 7 February 2018 / Accepted: 6 May 2018 /
Published online: 18 May 2018
© The Author(s) 2018

Abstract Hubness is an aspect of the *curse of dimensionality* related to the *distance concentration* effect. *Hubs* occur in high-dimensional data spaces as objects that are particularly often among the nearest neighbors of other objects. Conversely, other data objects become *antihubs*, which are rarely or never nearest neighbors to other objects. Many machine learning algorithms rely on nearest neighbor search and some form of measuring distances, which are both impaired by high hubness. Degraded performance due to hubness has been reported for various tasks such as classification, clustering, regression, visualization, recommendation, retrieval and outlier detection. Several hubness reduction methods based on different paradigms have previously been developed. Local and global scaling as well as shared neighbors approaches aim at repairing asymmetric neighborhood relations. Global and localized centering try to eliminate spatial centrality, while the related global and local dissimilarity measures are based on density gradient flattening. Additional methods and alternative dissimilarity measures that were argued to mitigate detrimental effects of distance concentration also influence the related hubness phenomenon. In this paper, we present a large-scale empirical evaluation of all available unsupervised hubness reduction methods and dissimilarity measures. We investigate several aspects of hubness reduction as well as its influence on data semantics which we measure via nearest neighbor classification. Scaling and density gradient flattening methods improve evaluation measures such as hubness and classification accuracy consistently for data sets from a wide range of domains, while centering approaches achieve the same only under specific settings.

Keywords Hubness · Curse of dimensionality · Secondary distances · Classification · Nearest neighbors

✉ Roman Feldbauer
roman.feldbauer@ofai.at

Arthur Flexer
arthur.flexer@ofai.at

¹ Austrian Research Institute for Artificial Intelligence, Freyung 6/6/7, 1010 Vienna, Austria

1 Introduction

Learning in high-dimensional spaces is often challenging due to various phenomena that are commonly referred to as curse of dimensionality [4]. One well-known aspect of the curse is concentration of distances (or measure). With dimensionality approaching infinity, all distances between pairs of objects become indistinguishable [23], undermining the very concept of neighborhood-based approaches.

Hubness is a related phenomenon of the dimensionality curse: in high-dimensional spaces, some objects are closer to the global centroid (unimodal data) or local centroid (multimodal data) [44]. These objects often emerge as *hubs* with high k -occurrence, that is, they are among the k -nearest neighbors of many objects. Simultaneously, other objects are extruded from nearest neighbor lists, which makes some of these objects appear in no or few of these lists (antihubs). Due to these effects, nearest neighbor relations between any two objects become asymmetric more often in high dimensions than in low dimensions. That is, there are more unidirectional relations in high dimensions (object x is among the nearest neighbors of y , but not vice versa). Nearest neighbor relations in high hubness regimes are prone to semantic incorrectness: Hubs propagate their encoded information too widely in corresponding distance spaces, while information carried by antihubs is essentially lost [57]. Consequently, these distance spaces do not reflect class information well, that is, semantic meaning of the data. Since intraclass distances should generally be smaller than interclass distances, nearest neighbor classification accuracy can be used as a proxy to measure semantic correctness [21].

Hubness has been identified as a detrimental factor in similarity-based machine learning, impairing several classification [44], clustering [47, 60], regression [7], graph analysis [22], visualization [17], and outlier detection [18, 19, 45] methods. Reports on affected tasks include multimedia retrieval [51], recommendation [48], collaborative filtering [25, 34], speaker verification [50], speech recognition [62], and image data classification [58].

Various strategies to mitigate detrimental effects of hubness have previously been investigated. In the course of these studies, several techniques for hubness reduction have been developed. An overview is given in Sect. 2.1. With the high number of hubness reduction methods now at hand, a comprehensive overview of their methodology and an empirical comparison of their performance is still lacking. This study aims at providing this overview and comparison. We present a comprehensive empirical evaluation of unsupervised hubness reduction methods, which showed promising results in previous studies, examining their ability to reduce hubness and whether they respect semantics of the data spaces at the same time.

This article is structured as follows: Sect. 2 reviews properties of hubness and how they may be used for hubness reduction. It also provides an overview of previous empirical comparisons. Section 3 describes the hubness reduction methods. Section 4 details the evaluation framework, data sets, evaluation measures and statistical analyses. The results of this evaluation are presented in Sect. 5 and discussed in Sect. 6. Section 7 concludes the paper and gives an outlook for future research.

This article is a substantially expanded version of a conference contribution [16]. The main extensions are the evaluation of more methods (twelve instead of four), on more data sets (50 instead of 28), an enhanced hyperparameter tuning scheme, and a more comprehensive analysis of results, including rigorous statistics.

2 Related work

Hubness was first noted as a problem in automatic music recommendation [3], more specifically that certain songs were being recommended conspicuously often in nearest neighbor-based playlists. The phenomenon of hubness has then been characterized extensively in both theoretical and empirical aspects by Radovanović et al. [44]. Their publication already provided some starting points for the development of methods mitigating negative effects of hubness. Subsequently, such hubness reduction methods were designed by a number of other authors [2, 21, 26, 27, 49, 56, 59]. We summarize these concepts and methods into four categories, as outlined in the paragraphs of Sect. 2.1.

2.1 Hubness and its reduction

Hubness is known to arise in high-dimensional data spaces. This was shown to be caused by inherent properties of data distributions, not by other effects such as finite sample sizes [44]. Data sets are often embedded in spaces of higher dimensionality than is needed to capture all their information. The minimum number of features necessary to encode this information is called intrinsic dimension (ID). More formally, ID refers to a lower-dimensional submanifold of the embedding space containing all data objects without information loss [8]. Several methods for intrinsic dimension estimation have been proposed (see, for example, Ref. [8] for a recent review). Empirical results suggest that hubness depends on a data set's intrinsic dimension rather than the embedding dimension [44]. A later study challenges this view and argues that hubness arises due to density gradients in data sets [39], that is, spatial variations in the density of empirical data distributions. Density gradients may originate from data generating processes. Data sets consisting of points sampled from a continuous probability density function $f(\cdot)$ exhibit density gradients. Consider, for example, the bell curve-shaped PDF of a normal distribution. Additionally, density gradients emerge necessarily, when sampling regions are bounded (that is, $\exists x : f(x) = 0$), which even holds for uniform distributions otherwise not showing density gradients. Consequently, data sets sampled from uniform distributions with bounds still show density gradients [39]. With increasing dimensionality, the ratio of the size of a boundary and its encapsulated volume increases exponentially. From the viewpoint of density gradient, this explains the emergence of hubs in high-dimensional bounded data [39]. Common **dimensionality reduction** (DR) methods have been investigated in regard to hubness. Empirical results indicate principal component analysis (PCA), independent component analysis (ICA), and stochastic neighborhood embedding (SNE) to not significantly change hubness, unless the number of features falls below the intrinsic dimension of a data set [44]. In the latter case, information loss regarding pairwise distances and nearest neighbor relations may occur. Since this is an undesired effect for any neighborhood-based analysis, these DR methods are not suitable for hubness reduction. On the other hand, DR methods changing underlying pairwise distances, such as isomap or diffusion maps, do reduce hubness when retaining a number of features greater than the intrinsic dimension [44]. This finding directly motivates the adaption and development of secondary distance measures specialized on hubness reduction, as discussed under the next categories.

Objects in proximity of the sample mean of some data distribution are prone to become hubs in high-dimensional spaces, which is known as the spatial centrality of hubs [44]. For unimodal data, hubs are often close to the global data centroid. Real-world data sets are often better described as a mixture of distributions, for which hubs tend to be close to the mean of individual distributions [44]. Since the exact mixture of distributions in real-world data is often unknown, k -means clusters [44] and local neighborhoods [27] have previously been used to

describe spatial centrality in multimodal data. On the other hand, antihubs are typically far from centers and can be considered distance-based outliers [44]. Spatial centrality can thus be described by correlation between k -occurrence and distance to the centroid. Reducing **spatial centrality** is another approach to reduce hubness. *Centering* (subtraction of the centroid) was proposed to eliminate hubness from text data using inner product similarities [56]. Based on this, *localized centering* was developed for hubness that may arise due to large data set size rather than high dimensionality [27]. The same authors present $DisSim^{Global}$ and $DisSim^{Local}$ as variants of the above, applicable to Euclidean distance spaces, and argue that hubness reduction is achieved by flattening the data density gradient [26].

Nearest neighbor relations between two objects x and y are considered symmetric, if x is among the nearest neighbors of y and vice versa. Hubness directly affects rates of symmetry with more asymmetric relations arising under high hubness conditions [49], because hubs are by definition nearest neighbors to very many data points but only one data point can be the nearest neighbor to a hub. In addition, asymmetric nearest neighbor relations offend against the pairwise stability of clusters [49], leading to wrong information propagation. For this reason, the third category of hubness reduction methods aims at **repairing asymmetric relations**. Several methods have been proposed that symmetrize these relations by transformation to secondary distance spaces, that is, they are computed from other primary distance spaces (e.g., Euclidean distances). Among these methods are *shared nearest neighbors* [32], *local scaling* [64], the *(non-iterative) contextual dissimilarity measure* [33], *mutual proximity* [49], and *simhub* [59]. Only the latter two methods were developed explicitly for hubness reduction.

Finally, the related concentration effect may be mitigated by using alternative distance measures, for example, fractional norms [23]. Analogously, **alternative distance measures** might be less prone to hubness than commonly used measures like Euclidean or cosine distances. Consequently, ℓ^p norms were investigated in regard to their influence on hubness [21]. The data-dependent m_p -dissimilarity measure was recently presented as an alternative to geometric distances [2].

2.2 Previous comparisons of hubness reduction

Several empirical comparisons of hubness reduction methods have been conducted previously, typically in the context of presenting new methods.

Five dimensionality reduction methods were tested for their capability to reduce hubness on three real-world data sets [44]. PCA, ICA, and SNE fail to reduce hubness, unless dimensionality is reduced below ID. Isomap and diffusion maps show some hubness reduction capability. No tests to examine whether data semantics are respected by dimensionality reduction were performed.

The local and global scaling methods mutual proximity (MP) and non-iterative contextual dissimilarity measure (NICDM) were investigated with respect to both hubness reduction and improved data semantics on thirty real-world data sets from various domains [49]. Both scaling methods showed improved performance measures on high-dimensional data sets, and no degradation on low-dimensional data sets. This is true for both hubness reduction and nearest neighbor classification. Approximate MP variants were found to perform nearly as well as full MP.

Shared nearest neighbors (SNN) was compared to local scaling and mutual proximity on six real-world data sets [20]. SNN was able to reduce hubness, though not as strongly as the other methods. Classification accuracy was improved by SNN only in three data sets, and

the method thus deemed inferior to LS and MP, both of which improved accuracy in all six cases.

Simhub is a hybrid method composed of supervised (simhub^{PUR}) and unsupervised (simhub^{IN}) parts. Its evaluation was primarily performed for full simhub [59]. The individual component simhub^{IN} was compared to SNN on one image data set, for which it surpassed its competitor in terms of two classification measures.

Centering was compared to MP on three data sets from the text domain [56]. Both hubness and classification measures were improved for both methods to a similar extent and on par with a state-of-the-art technique. A follow-up study [27] compared localized centering (LCENT), centering, mutual proximity, local scaling, and a commute-time kernel (CTL) on four text data sets. The strongest hubness reduction was achieved by MP. LCENT and LS performed slightly better than MP in terms of classification accuracy. CTL appears to be non-effective in hubness reduction.

DisSim^{Local} was shown to outperform MP on four real-world data sets [26]. Both methods improve hubness and accuracy measures compared to DisSim^{Global} and to the Euclidean baseline.

Finding an optimal ℓ^p norm was shown to improve classification on seven data sets [21]. On four data sets, LS and MP were able to further increase accuracy. SNN yielded non-competitive results over all seven data sets in that comparison.

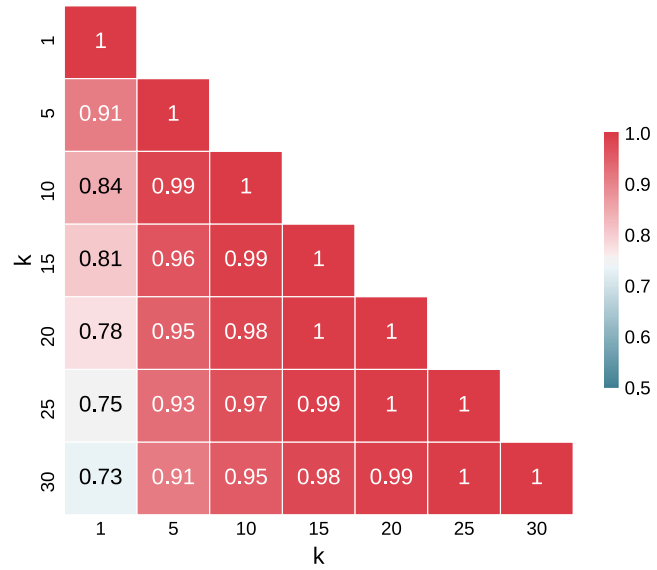
The m_p -dissimilarity reduces hubness in synthetic data [2]. It was reported to do so as well in real-world data sets, but results were not provided.

A comprehensive empirical comparison overcoming several shortcomings of the above-mentioned ones is still lacking. Such a comparison must (i) evaluate all available hubness reduction methods (ii) on a large number of data sets (iii) from various domains and (iv) present appropriate statistical treatment of the results. We strive to address all of these issues in this study. Please note that we did not include commute-time kernels and dimensionality reduction in our study, since these methods performed very poorly in previous studies. We also restricted our comparison with unsupervised methods, which do not use class label information for hubness reduction. Supervised methods like simhub [59] should rather be compared to related supervised approaches like, e.g., metric learning [35], which is beyond the scope of this paper.

3 Hubness reduction methods

This section reviews all unsupervised hubness reduction methods used in this paper. Some methods operate on vector data directly, others on distances between pairs of objects. Usually, Euclidean or cosine distances are used as input for the latter methods. Some methods also operate on non-metric dissimilarities or similarities. We use Euclidean distances as primary distances, unless kNN-classification with cosine distances yields significantly better results (McNemar test, not shown for brevity). Let $B \subseteq \mathbb{R}^m$ be a non-empty data set with n data objects in m -dimensional space, that is, $b_i = (b_{i,1}, \dots, b_{i,m}) \in B$ for $i \in \{1, \dots, n\}$. Let $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ be short-hands for three m -dimensional numeric vectors b_x , b_y , and b_z , respectively. Let $d : B \times B \rightarrow \mathbb{R}$ be a measure of dissimilarity. The dissimilarity between two objects $\mathbf{x}$ and $\mathbf{y}$ is then denoted as $d_{\mathbf{x},\mathbf{y}}$. Most hubness reduction methods have tunable hyperparameters. We try to follow the notation of the original publications, and thus reuse some symbols in multiple methods. We do so only, if their meaning is closely related. For example, k always

Fig. 1 Spearman correlation of k -occurrence using all data sets from Table 3. For each of the 50 data sets k -occurrences with $k = 1, 5, 10, 15, 20, 25, 30$ are computed and correlations between k -occurrences with different k are shown



refers to neighborhood size, though individual methods may use nearest neighbor information differently. Descriptions of all parameters follow in the next sections.

3.1 Measuring hubness

Before we introduce hubness reduction methods, we briefly introduce measures commonly used for describing the degree of hubness in a data set.

3.1.1 k -occurrence

The k -occurrence $O^k(x)$ of an object x is defined as the number of times x resides among the k nearest neighbors of all other objects in the data set. In the notion of network analysis, $O^k(x)$ is the indegree of x in a directed kNN graph. It is also known as reverse neighbor count.

3.1.2 Hubness

Hubness is typically measured as the skewness of the k -occurrence distribution [44]:

$$S^k = \frac{\mathbb{E}[(O^k - \mu_{O^k})^3]}{\sigma_{O^k}^3}, \quad (1)$$

where μ_{O^k} and σ_{O^k} denote the mean and standard deviation of the k -occurrence distribution, respectively. Typical values of k used in the literature include 1, 5, 10, and 20. Previous research indicates the choice of k to be non-critical. For the real-world data sets used in this paper, we observe very high correlation of k -occurrence among various k values (Fig. 1), except for $k = 1$, which is less correlated. We therefore deem any values of $5 \leq k \ll n$ suitable for analysis of hubness reduction and use $k = 10$ for all hubness measurements in this paper.

3.2 Methods based on repairing asymmetric relations

The following methods aim at repairing asymmetric neighbor relations. All these methods compute *secondary distances* by transforming the original primary distance (for example, Euclidean or cosine) in a data set.

3.2.1 Local scaling and the non-iterative contextual dissimilarity measure

Local scaling (LS) was proposed to improve spectral clustering performance on data of multiple scales [64]. Pairwise secondary distances are calculated as:

$$\text{LS}(d_{x,y}) = 1 - \exp\left(-\frac{d_{x,y}^2}{\sigma_x \sigma_y}\right). \quad (2)$$

The scaling parameter $\sigma_x(\sigma_y)$ is set to the distance between object $\mathbf{x}(\mathbf{y})$ to its k -th nearest neighbor. LS induces increased symmetry in nearest neighbor relations by incorporating local distance information and was proposed for hubness reduction for that reason [49].

The non-iterative contextual dissimilarity measure (NICDM, [33]) is closely related to local scaling: The scaling factor of an object $\mathbf{x}$ is set to the mean distance to its k nearest neighbors (compared to using only the k -th neighbor in LS). We use NICDM transformations adapted for hubness reduction [49]:

$$\text{NICDM}(d_{x,y}) = \frac{d_{x,y}}{\sqrt{\mu_x \mu_y}}, \quad (3)$$

where μ_x denotes the mean distance from object $\mathbf{x}$ to its k -nearest neighbors (analogous for μ_y and object $\mathbf{y}$). Parameter k in both LS and NICDM should reflect the embedding space around each object and can be tuned in order to minimize hubness.

3.2.2 Global scaling: mutual proximity

While LS and NICDM use local distance statistics to enforce symmetric neighborhoods, mutual proximity (MP, [49]) incorporates information of all pairwise distances in the data set to achieve the same. Let X be a random variable of distances between $\mathbf{x}$ and all other objects in the data set (analogously for Y and $\mathbf{y}$), and P the joint probability density function, then

$$\text{MP}(d_{x,y}) = P(X > d_{x,y} \cap Y > d_{y,x}). \quad (4)$$

Secondary distances are calculated as the complement of the joint probability of two objects being nearest neighbors to each other (i.e., $1 - \text{MP}$). To allow for this probabilistic view, MP models the distances $d_{x,i \in \{1, \dots, n\} \setminus x}$ between an object $\mathbf{x}$ and all other objects with some distribution. When using the empirical distance distribution, mutual proximity between two objects $\mathbf{x}$ and $\mathbf{y}$ is calculated by counting objects whose distances to both $\mathbf{x}$ and $\mathbf{y}$ are greater than $d(x, y)$:

$$\text{MP}(d_{x,y}) = \frac{|\{j : d_{x,j} > d_{x,y}\} \cap \{j : d_{y,j} > d_{y,x}\}|}{n - 2}. \quad (5)$$

Compared to the formula in the Ref. [49], we added a subtrahend to the denominator to account for identity distances. This influences the normalization to the $[0, 1]$ range but does not change neighborhood order, hubness, or nearest neighbor classification.

In the framework of MP, distances can also be modeled with any (continuous) distribution. This is especially useful, when the user has prior knowledge of the given data domain. Additionally, if X and Y are assumed to be independent, Formula 4 simplifies to

$$\text{MP}^{\text{I}}(d_{x,y}) = P(X > d_{x,y}) \cdot P(Y > d_{y,x}). \quad (6)$$

These approximations simplify calculations and decrease the computational complexity of MP. The Gaussian-based mutual proximity variant ($\text{MP}^{\text{GaussI}}$) models the distances of each object x to all other objects with a normal distribution ($X \sim \mathcal{N}(\mu, \sigma^2)$). Parameters μ_x and σ_x^2 can be estimated with the sample mean $\hat{\mu}_x$ and variance $\hat{\sigma}_x^2$:

$$\mathcal{N}_x \rightsquigarrow \hat{\mu}_x = \frac{1}{n-1} \sum_{i=1, i \neq x}^n d_{x,i}, \quad \hat{\sigma}_x^2 = \frac{1}{n-1} \sum_{i=1, i \neq x}^n (d_{x,i} - \hat{\mu}_x)^2 \quad (7)$$

Compared to Ref. [49], we exclude self-distances $d_{x,x}$ from parameter estimation. This should presumably improve the approximation, since self-distances are not informative. Secondary distances based on $\text{MP}^{\text{GaussI}}$ are calculated as

$$\text{MP}^{\text{GaussI}}(d_{x,y}) = \text{SF}(d_{x,y}, \hat{\mu}_x, \hat{\sigma}_x^2) \cdot \text{SF}(d_{y,x}, \hat{\mu}_y, \hat{\sigma}_y^2), \quad (8)$$

where $\text{SF}(d, \mu, \sigma^2) = 1 - \text{CDF}(d, \mu, \sigma^2)$, that is, the survival function (complement to the cumulative density function) at value d given the indicated distribution.

3.2.3 Shared nearest neighbors and simhub

A shared neighborhood is the intersection of the nearest neighbor sets of two objects [32]. Secondary distances based on shared nearest neighbors (SNN) increase pairwise stability and relation symmetry, which is considered beneficial for hubness reduction [20]. SNN similarities are calculated as:

$$\text{SNN}(x, y) = \frac{|\text{kNN}(x) \cap \text{kNN}(y)|}{k}, \quad (9)$$

where $\text{kNN}(\cdot)$ is the set of the k -nearest neighbors of some object.

Simhub [59] is a shared neighbors approach that weights shared neighbors z by informativeness (increasing weights of rare neighbors) and purity (penalizes neighborhoods with inconsistent class labels). Both weights may be used simultaneously (simhub) or separately ($\text{simhub}^{\text{IN}}$ and $\text{simhub}^{\text{PUR}}$ for informativeness and purity, respectively). Simhub is a supervised method when using purity weights. We thus restrict our evaluation to the unsupervised $\text{simhub}^{\text{IN}}$:

$$\begin{aligned} \text{simhub}^{\text{IN}}(x, y) &= \frac{\sum_{z \in (\text{kNN}(x) \cap \text{kNN}(y))} I_n(z)}{k \cdot \max I_n}, \\ I_n(z) &= \log \frac{n}{O^k(z) + 1}, \quad \max I_n = \log n \end{aligned} \quad (10)$$

where $I_n(z)$ is the occurrence informativeness of a shared neighbor z in a data set of size n . The neighborhood radius k can be tuned in both SNN-based methods to minimize hubness. Computing $1 - \text{SNN}$, or $1 - \text{simhub}$ turns the similarities into distances.

3.3 Methods based on spatial centrality reduction and density gradient flattening

Centering approaches aim at reducing spatial centrality, and use modified inner product similarities to span distance spaces. Global and local DisSim try to flatten density gradients, and construct dissimilarities from squared Euclidean distances.

3.3.1 Centering and localized centering

Centering is a widely used preprocessing step that shifts vectors ($\mathbf{x}$) so that the space origin coincides with the global centroid ($\bar{c}$). Centering dissimilarities can be calculated as

$$\text{CENT}(\mathbf{x}, \mathbf{y}) = -\langle \mathbf{x} - \bar{c}, \mathbf{y} - \bar{c} \rangle, \quad (11)$$

where $\langle \cdot, \cdot \rangle$ is the inner product of two vectors. The method was proposed for hubness reduction in the context of natural language processing [56]. Centering moves the centroid to the origin. Inner product dissimilarities between any object and the origin (zero vector) are uniformly zero. Centering effectively eliminates spatial centrality in inner product spaces, which should reduce hub emergence. Following this idea, localized centering was developed [27]: Instead of shifting the whole vector space, LCENT is a dissimilarity measure based on *global affinity* (mean similarity between an object $\mathbf{x}$ and all other objects) and *local affinity* (mean similarity between $\mathbf{x}$ and its k nearest neighbors):

$$\text{LCENT}(\mathbf{x}, \mathbf{y}) = -\langle \mathbf{x}, \mathbf{y} \rangle + \langle \mathbf{x}, c_k(\mathbf{x}) \rangle^\gamma, \quad (12)$$

where $c_k(\mathbf{x})$ denotes the local centroid among the k nearest neighbors of $\mathbf{x}$, γ is a parameter controlling the penalty introduced by the second term, and the leading negative sign indicates dissimilarities. LCENT dissimilarities are not guaranteed to be positive. Parameters γ and k can be tuned to minimize hubness.

3.3.2 Global and local dissimilarity measures

The above-described centering approaches have no effect on Euclidean distances. As an alternative, two dissimilarity measures were introduced [26]: They reduce hubness by flattening the density gradient and thus eliminate spatial centrality in commonly used Euclidean spaces. The global variant DisSim^{Global} (DSG) removes *sample-wise centrality* of two objects $\mathbf{x}$ and $\mathbf{y}$:

$$\text{DSG}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2 - \|\mathbf{x} - c\|_2^2 - \|\mathbf{y} - c\|_2^2, \quad (13)$$

where c is the global centroid and $\|\cdot\|_2^2$ indicates the squared Euclidean norm.

The local variant DisSim^{Local} (DSL) is free from the assumption that all instances in the data set come from the same distribution: Instead of subtracting the global centroid, local centroids are estimated as $c_k(\mathbf{x}) = \frac{1}{k} \sum_{\mathbf{x}' \in \text{kNN}(\mathbf{x})} \mathbf{x}'$, where $\text{kNN}(\mathbf{x})$ is the set of k -nearest neighbors of $\mathbf{x}$, and substitution in Formula 13 yields:

$$\text{DSL}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2^2 - \|\mathbf{x} - c_k(\mathbf{x})\|_2^2 - \|\mathbf{y} - c_k(\mathbf{y})\|_2^2. \quad (14)$$

Parameter k can be tuned to minimize hubness.

3.4 Hubness-resistant dissimilarity measures

The methods described in this section try to avoid hubness by using alternative distance measures between data objects.

3.4.1 Choosing ℓ^p norms and the m_p -dissimilarity measure

Euclidean distances correspond to a special case of the family of ℓ^p norms (also known as Minkowski norms) with $p = 2$. The effect of using norms with $p \neq 2$ in the context of *hubness* has been investigated previously [21]. An ℓ^p norm of a vector $(\mathbf{x} - \mathbf{y})$ can be interpreted as a dissimilarity between $\mathbf{x}$ and $\mathbf{y}$ and is calculated as follows:

$$d^p(\mathbf{x}, \mathbf{y}) = \left(\sum_{i=1}^m |x_i - y_i|^p \right)^{1/p} \quad (15)$$

For $0 < p < 1$ the resulting Minkowski norms (also called fractional norms) do not guarantee the triangle inequality. Consequently, they do not constitute full distance metrics. The parameter p can be tuned to minimize hubness. In this work, we evaluate ℓ^p norms with $p = 0.25, 0.5, \dots, 5$ (as in Ref. [21]) and ten values randomly selected from $]0, 5[$.

A data-dependent dissimilarity measure was recently derived from ℓ^p norms [2]. The m_p -dissimilarity takes into account data distributions by estimating the probability mass $|R_i(\mathbf{x}, \mathbf{y})|$ in a region R around $\mathbf{x}$ and $\mathbf{y}$ in each dimension i :

$$m_p(\mathbf{x}, \mathbf{y}) = \left(\frac{1}{m} \sum_{i=1}^m \left(\frac{|R_i(\mathbf{x}, \mathbf{y})|}{n} \right)^p \right)^{1/p}, \quad |R_i(\mathbf{x}, \mathbf{y})| = \sum_{q=l}^u |h_{iq}| \quad (16)$$

That is, all objects are binned in each dimension. Let h_{il} and h_{iu} be the bins that contain $\min(x_i, y_i)$ and $\max(x_i, y_i)$, respectively. The probability data mass $|R_i|$ is then estimated by counting the objects in all bins from h_{il} to h_{iu} . R_i replaces the geometric distance used in ℓ^p norms. Dissimilarities are thus increased in dense regions and decreased in sparse regions.

3.5 Time and space complexity

Hubness reduction is expensive due to calculation of distances between all pairs of objects. Table 1 lists time complexity of all methods. All methods applied on data vectors require $\mathcal{O}(n^2m)$ time¹. Since their prefactors differ considerably, timings for two synthetic data sets of increasing size and dimensionality are also provided. Methods applied on data distances require $\mathcal{O}(n^2)$ or $\mathcal{O}(n^3)$ time in addition to $\mathcal{O}(n^2m)$ time for preprocessing primary distances. All methods require $\mathcal{O}(n^2)$ space for returning the distance matrix. For primary distances in data sets with $m > n$ this is dominated by the memory requirement of the input vectors $\mathcal{O}(nm)$. Intermediate steps typically require $\mathcal{O}(n)$ space, except for m_p -dissimilarity, which requires $\mathcal{O}(b^2m)$ for distances between all pairs of bins in each dimension, where b is the number of bins, and $b \ll n$.

3.6 OFAI Hub-Toolbox

The availability of machine learning algorithms not only as formulas, but also as working code in reference implementations allows easy reproducibility and applicability of methods. Consequently, all methods described in this publication are available as part of a free open source software package for the Python programming environment. The Hub-Toolbox is easily installable from the PyPI package repository² and licensed under GNU GPLv3. Please

¹ Note that centering requires only $\mathcal{O}(nm)$ time, if no subsequent distance calculations are performed.

² <https://pypi.python.org/pypi/hub-toolbox/>.

Table 1 Time complexity for computing distances between all pairs of objects and timings for two synthetic data sets (B_{1000} with $n = m = 1000$ and B_{10000} with $n = m = 10000$). Timings were performed with the Hub-Toolbox for Python (see Sect. 3.6) on a single core of an Intel Core i5-6500 CPU 3.20GHz with 15.6 GiB main memory

	Input data	Time complexity	Time (s) B_{1000}	Time (s) B_{10000}	Parameters
Eucl	Vectors	$\mathcal{O}(n^2m)$	0.1	26	–
cos	Vectors	$\mathcal{O}(n^2m)$	0.3	48	–
MP	Distances	$\mathcal{O}(n^3)$	1.6	1382	–
MP ^{GaussI}	Distances	$\mathcal{O}(n^2)$	0.2	8	–
LS	Distances	$\mathcal{O}(n^2)$	< 0.1	4	$k = 10$
NICDM	Distances	$\mathcal{O}(n^2)$	< 0.1	3	$k = 10$
SNN	Distances	$\mathcal{O}(n^3)$	0.7	754	$k = 10$
simhub ^{IN}	Distances	$\mathcal{O}(n^3)$	2.5	2510	$s = 10$
CENT	Vectors	$\mathcal{O}(n^2m)$	0.1	33	–
LCENT	Vectors	$\mathcal{O}(n^2m)$	0.5	71	$\kappa = 10, \gamma = 1.5$
DSG	Vectors	$\mathcal{O}(n^2m)$	0.2	46	–
DSL	Vectors	$\mathcal{O}(n^2m)$	0.3	51	$k = 10$
ℓ^p norm	Vectors	$\mathcal{O}(n^2m)$	29.8	30018	$p = 1.5$
m_p -dissim	Vectors	$\mathcal{O}(n^2m)$	75.8	63143	$p = 1.5,$ $n_{\text{bins}} = 200$

visit the GitHub page³ for source code, development versions, issue tracking, and contribution possibilities. A MATLAB version of the Hub-Toolbox providing core functionality is also available on GitHub.⁴

4 Evaluation

The evaluation strategy focuses on two indicators: (i) hubness is measured as the skewness of k -occurrence distribution (see Sect. 3.1); (ii) k -nearest neighbor classification accuracy C^{kNN} is used to measure the degree of correct data semantics in primary and secondary distance spaces. The neighborhood size parameter k and the weighting mode for k -nearest neighbor classification are selected in a nested cross-validation scheme (see Fig. 2). Weighting can be distance-based, that is, neighbors are weighted by their inverse distance during prediction, giving more influence to closer neighbors. Otherwise, nearest neighbors have uniform weights, and prediction is a majority vote among them. Ties are resolved by the nearest neighbor.

Baseline accuracy (column C^{kNN} in Table 3) is obtained in a cross-validation procedure as described in Sect. 4.1. Parameter k is selected by maximizing C^{kNN} . This is performed using both Euclidean and cosine distances. For each data set, C^{kNN} is reported for the distance measure that yielded higher accuracy as indicated in column ‘d’ in Table 3.

³ <https://github.com/OFAI/hub-toolbox-python3>.

⁴ <https://github.com/OFAI/hub-toolbox-matlab>.

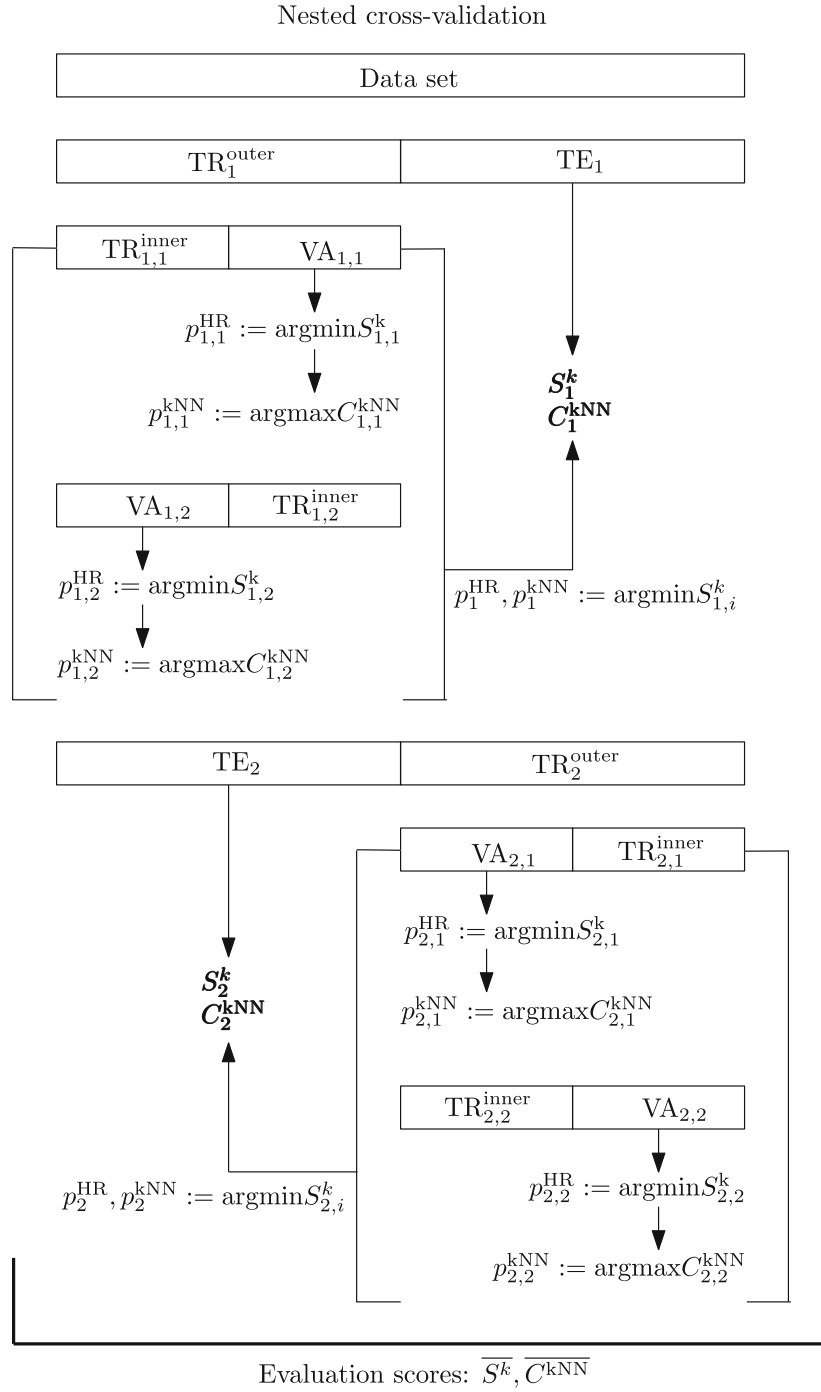


Fig. 2 Nested cross-validation scheme. Twofold inner and twofold outer loops are depicted for better presentability. The actual evaluation is performed as tenfold/tenfold nested CV. Abbreviations: CV...cross-validation, TR...training set, VA...validation set, TE...test set, $\overline{S^k}$...mean hubness, $\overline{C^{kNN}}$...mean accuracy. Text superscripts indicate outer/inner CV, subscript indices display number of outer fold and inner fold

4.1 Evaluation scheme

In this section, we evaluate the hubness reduction methods described in Sect. 3 with regard to improved hubness and data semantics. We follow a standard procedure for comparing classifiers, preprocessing, or postprocessing steps over multiple data sets [14]. The procedure requires 'reliable' scores, that is, they must come from an evaluation scheme with sufficiently

many experiments on each data set, which should ideally be performed on the same random samples for each evaluated method.

We evaluate twelve hubness reduction methods on fifty data sets in a nested cross-validation (CV) scheme [9] as depicted schematically in Fig. 2 (for twofold CV for reasons of better presentability). Each data set is first split into outer training set (TR^{outer}) and test set (TE) in a tenfold outer cross-validation. TR^{outer} is then split into an inner training set (TR^{inner}) and a validation set (VA) in a tenfold inner cross-validation. On both levels, the data is split randomly and stratified based on the class labels. The splits are identical for all methods to meet the requirements of the comparison procedure. In the inner CV we first find optimal hyperparameter values for hubness reduction, then an optimal k and weighting mode for kNN classification. We perform randomized hyperparameter search [5], that is, we draw hyperparameter values uniformly and randomly from a predefined range, until a certain budget of samples is consumed.

Specifically, randomized hyperparameter search is performed to minimize hubness ($\min S^k$) on the validation set VA in each inner loop i of each outer loop o for each hubness reduction method. Table 2 lists the parameter ranges used in the optimization steps. A budget of 30 hyperparameter values is used (or 30 pairs of values in case two hyperparameters are being optimized), and the best value (pair) is denoted as $p_{o,i}^{\text{HR}}$. This step is omitted for MP, CENT, and DSG, for which there are no hyperparameters to tune. Secondary distances between objects in VA and TR^{inner} are then calculated using hubness reduction with optimal hyperparameters $p_{o,i}^{\text{HR}}$. Subsequently, classification accuracy is maximized ($\max C^{\text{kNN}}$) on VA in the inner loop using a kNN classifier. The best pair of hyperparameter values (k and weighting mode) from a budget of ten is denoted as $p_{o,i}^{\text{kNN}}$. For each outer fold o , the best hyperparameters p_o^{HR} and p_o^{kNN} are selected from the inner fold showing lowest hubness among all inner folds $i = 1 \dots 10$.

The hubness reduction methods are scored on TE using these optimized hyperparameters, that is, p_o^{HR} for a given hubness reduction method, and p_o^{kNN} for kNN classification. We

Table 2 Hyperparameters and selection ranges for hubness reduction methods (first twelve rows) and kNN classification (last row). The methods have one or two parameters, or are parameter-free. Ranges of numerical parameters are subsets of $\mathbb{N}^+$, or $\mathbb{R}^+$ as indicated by decimal points

	Parameter 1	Range	Parameter 2	Range
MP	—	—	—	—
MP ^{GaussI}	—	—	—	—
LS	k	[1, 50]	—	—
NICDM	k	[1, 50]	—	—
SNN	k	[1, 50]	—	—
simhub ^{IN}	s	[1, n]	—	—
CENT	—	—	—	—
LCENT	κ	[5, 100]	γ	]0., 5.]
DSG	—	—	—	—
DSL	κ	[1, n]	—	—
ℓ^p norm	p	]0., 5.]	—	—
m_p -dissim	p	]0., 5.]	n_{bins}	[10, $\min(\frac{n}{2}, 200)$]
kNN	k	[1, 10], 15, 20, 25, 30, 40, 50, 100	Weights	{Uniform, distance}

calculate mean hubness $\overline{S^k}$ and mean kNN classification accuracy $\overline{C^{\text{kNN}}}$ over ten outer folds $o = 1 \dots 10$. Given the evaluation strategy, these scores are assumed to be robust and unbiased performance estimates. The hubness reduction methods are then compared using the nonparametric Friedman test on both measures, independently. If significant differences are found among the methods, the post hoc Nemenyi test is used to determine the best performing method.

4.2 Data sets

We evaluate the previously described methods on 50 different public machine learning data sets. The selection of data sets was motivated primarily by the observed degree of hubness: Data sets with high hubness were selected in order to exploit the full potential of hubness reduction methods. Several low hubness data sets were also added to the collection. We did not expect substantial performance boosts in these cases. However, we use these data sets to investigate possible adverse effects of hubness reduction when there is hardly anything to reduce. Please note that some low-dimensional data sets were added for the very same reason. Table 3 contains details about the data sets, 28 of which have already been used in a previous study. Additional 22 data sets were obtained from four public machine learning repositories.

- Already used in a previous study [49]: *arcene*, *amlall*, *gisette*, *mfeat-factors*, *mfeat-karhunen*, *mfeat-pixels*, *heart*, *sonar*, *dexter*, *mini-newsgroups*, *dorothea*, *lungcancer*, *reuters-transcribed*, *ovarian 61902*, *australian*, *diabetes*, *german numbers*, *liver-disorders*, *breast-cancer*, *duke (train)*, *colon-cancer*, *fourclass*, *ionosphere*, *splice*, *c1ka-twitter*, *c224a-web*, *corel1000*, *movie-reviews*.
Please note that *ballroom* and *ismir2004* were omitted, because they use symmetrized Kullback–Leibler divergence, which is non-trivial to combine with some of the methods evaluated here.
- UCI Machine Learning Repository [38]: *Parkinson Speech Dataset with Multiple Types of Sound Recordings* [46], *Amazon Commerce reviews* [55], *p53 Mutants* [13], *CNAE-9* [12], *Student Alcohol Consumption* [42], *Arrhythmia* [24], *Farm Ads* [41], *Mice Protein Expression* [28], *Opportunity Activity Recognition* [11], *Chronic Kidney Disease* [53]
- LibSVM [10]: *dna* [30], *protein* [63], *sector* [40], *rcv1.multiclass* [37]
- OpenML [61]: *Semeion Handwritten Digit* [52], *AP Breast Ovary* and *OVA Uterus* [54], *wap.wc*, *hepatitisC*, *Lymphoma* and *Spectrometer*
- MLdata [29]: *DMOZ* [15]

All data sets were downloaded from the sources indicated above and split into feature and label vectors according to their individual descriptions. Euclidean and cosine distances were calculated with the SciPy package for Python. Missing values were imputed with the median strategy over all instances in the following six data sets: Mice Protein Expression, Opportunity Activity Recognition, p53 Mutants, Chronic Kidney Disease, Arrhythmia, Lymphoma. For exceptionally large data sets ($n > 10000$), a stratified random sample of 2000 instances was drawn, preserving the percentage of instances for each class. This applies to the following four data sets marked with asterisks in column n of Table 3: Opportunity Activity Recognition, rcv1.multiclass, protein, sector. The data set p53 Mutants has highly skewed class distribution ($< 1\%$ negatives). To reduce this skewness, all negative instances were kept, but positive instances were sampled at random until they made up 90% of the new subset. The preexisting train-test-splits of Parkinson Multiple Sound Recordings were ignored and merged into a single data set. Categorical features of the Student Alcohol Con-

Table 3 Overview of 50 data sets from public machine learning repositories (*Source*) ordered by ascending hubness ($S^{k=10}$) of the indicated distance space (d)

#	Source	Name	Cls.	n	m	m_{mle}	d	C^{kNN}	$S^{k=10}$
1	UCI	Opportunity activity recognition	5	*2000	238	2	ℓ^2	0.8995	-0.1156
2	LibSVM	Fourclass (sc)	2	862	2	2	ℓ^2	1.0000	0.1528
3	LibSVM	Liver-disorders (sc)	2	345	6	4	ℓ^2	0.6260	0.1795
4	OpenML	Spectrometer	2	531	101	5	ℓ^2	0.9661	0.1903
5	UCI	Mice protein expression	8	1080	77	2	cos	0.9787	0.1961
6	LibSVM	Australian	2	690	14	2	ℓ^2	0.6928	0.2011
7	UCI	Chronic kidney disease	2	400	24	2	cos	0.7400	0.2745
8	UCI	Parkinson speech	2	1208	26	3	ℓ^2	0.6755	0.3932
9	UCI	Arcene	2	100	10000	8	ℓ^2	0.7500	0.4428
10	LibSVM	Breast-cancer (sc)	2	683	10	1	ℓ^2	0.9693	0.5686
11	LibSVM	Heart (sc)	2	270	13	3	ℓ^2	0.8222	0.5734
12	LibSVM	Colon-cancer	2	62	2000	10	ℓ^2	0.7742	0.5950
13	LibSVM	Diabetes (sc)	2	768	8	5	cos	0.7656	0.5950
14	UCI	mfeat-karhunen	10	2000	64	7	ℓ^2	0.9755	0.6898
15	KR	Ovarian-61902	2	253	15154	8	ℓ^2	0.9447	0.7603
16	OpenML	semeion	10	1593	256	12	ℓ^2	0.9165	0.8012
17	UCI	mfeat-factors	10	2000	216	6	ℓ^2	0.9570	0.8193
18	LibSVM	Duke (train)	2	38	7129	11	ℓ^2	0.7105	0.8275
19	CP	c224a web	14	224	1244	21	cos	0.9286	0.8345
20	LibSVM	German numbers (sc)	2	1000	24	5	ℓ^2	0.7320	0.8835
21	UCI	mfeat-pixels	10	2000	240	9	ℓ^2	0.9785	0.9642
22	KR	amlall	2	72	7129	11	ℓ^2	0.9167	1.1655
23	LibSVM	Sonar (sc)	2	208	60	5	ℓ^2	0.8462	1.1798
24	UCI	Student alcohol consumption	5	1044	56	4	ℓ^2	0.4157	1.2105
25	LibSVM	Splice (sc)	2	1000	60	7	cos	0.7720	1.2288
26	KR	Lungcancer	2	181	12533	10	ℓ^2	1.0000	1.2483
27	UCI	p53 mutants	2	1430	5408	6	cos	0.9294	1.2574
28	Corel	corel1000	10	1000	192	7	ℓ^2	0.6820	1.4179
29	UCI	Arrhythmia	13	452	279	10	ℓ^2	0.5951	1.4994
30	LibSVM	rcv1.multiclass	42	*2000	47236	10	cos	0.7550	1.5359
31	UCI	Reuters-transcribed	10	201	2730	23	ℓ^2	0.5672	1.6147
32	LibSVM	Ionosphere (sc)	2	351	34	5	ℓ^2	0.8917	1.6763
33	OpenML	OVA uterus	2	1545	10936	13	ℓ^2	0.9346	1.7759

Table 3 continued

#	Source	Name	Cls.	n	m	m_{mle}	d	C^{kNN}	$S^{k=10}$
34	OpenML	Lymphoma	11	96	4026	9	ℓ^2	0.8646	1.8785
35	UCI	Farm ads	2	4143	54877	1	cos	0.8938	1.9327
36	UCI	Gisette	2	6000	5000	51	cos	0.9783	1.9667
37	OpenML	AP breast ovary	2	542	10936	14	ℓ^2	0.9004	1.9812
38	OpenML	Hepatitis C	3	283	54621	25	ℓ^2	0.8975	2.1221
39	UCI	Dorothea	2	800	100000	253	cos	0.9375	2.3578
40	UCI	CNAE-9	9	1080	856	5	cos	0.8713	2.5492
41	UCI	Dexter	2	300	20000	34	ℓ^2	0.8667	3.3307
42	MLDATA	DMOZ	5	1329	10630	5	cos	0.4981	3.6351
43	UCI	Amazon commerce reviews	50	1500	10000	11	cos	0.3573	4.1013
44	LibSVM	Protein	3	*2000	357	34	cos	0.5845	4.2757
45	PaBo	Movie-reviews	2	2000	10382	44	ℓ^2	0.7935	4.3452
46	UCI	Mini-newsgroups	20	2000	8811	16	ℓ^2	0.8330	4.3705
47	LibSVM	Sector	104	*2000	55197	11	cos	0.7015	5.5795
48	OpenML	Wap.wc	20	1560	8460	12	cos	0.5045	9.3380
49	CP	c1ka-twitter	17	969	49820	44	cos	0.3633	10.7119
50	LibSVM	dna	3	2000	180	5	ℓ^2	0.8790	15.5188

The data sets are characterized by their *Name*, number of classes (*Cls.*), instances (n), features (m), and their estimated intrinsic dimension (m_{mle}) [36]. Column C^{kNN} reports baseline nearest neighbor classification performance (see Sect. 4). Asterisks indicate random samples as discussed in Sect. 4.2

sumption data set were transformed with a One-Hot-Encoder, i.e., to multiple binarized features. The Chronic Kidney Disease ARFF file contains several formatting errors (like, e.g., tabs after comma or at EOL, double commas) that hinder import into the evaluation framework. These errors were corrected manually. The SVMlight-styled file of the protein data set was missing leading zeros for floating point values in [0.0, 1.0). These were added manually.

5 Results

We found significant overall differences between the evaluated methods, both in terms of hubness reduction (Friedman $p < .000$) and nearest neighbor classification performance (Friedman $p < .000$). To analyze differences in more detail, we use critical difference plots ([14], see Fig. 3a, b) and post hoc Nemenyi tests. A critical difference (CD) plot shows the average ranks achieved by the competing methods across all data sets. When comparing all methods against each other, groups of methods not showing significant differences are connected with a black bar. In addition, this critical difference (CD bar length) is shown above the graph. It depends on the number of compared models (methods), the number of measurements (data sets), and the confidence level.

5.1 Performance ranking

The CD plot for hubness reduction comparing twelve methods plus two baseline distances (Euclidean, cosine) is shown in Fig. 3a. Well-performing methods have a low average rank, that is they reduce hubness more strongly than other methods on many data sets. Any two methods with a rank difference of at least 2.81 (CD bar length) perform significantly different according to the Nemenyi test ($K = 14$ models, $N = 50$ measurements, $\alpha = .05$). That is, methods connected by a black bar do not differ significantly. To give one concrete example, CENT yielded significantly lower hubness than Eucl, but it does not compared to cos. Therefore, CENT is connected to cos with a CD bar, but not to Eucl. On a more general note, the post hoc Nemenyi tests show that nearly all methods reduce hubness compared to baseline distances Eucl and cos, except for using ℓ^p norms, DSG, and LCENT. Method LCENT even shows increased hubness on average. Strongest hubness reduction was achieved by DSL, followed without significant rank differences by MP, simhub^{IN}, LS, NICDM, and SNN.

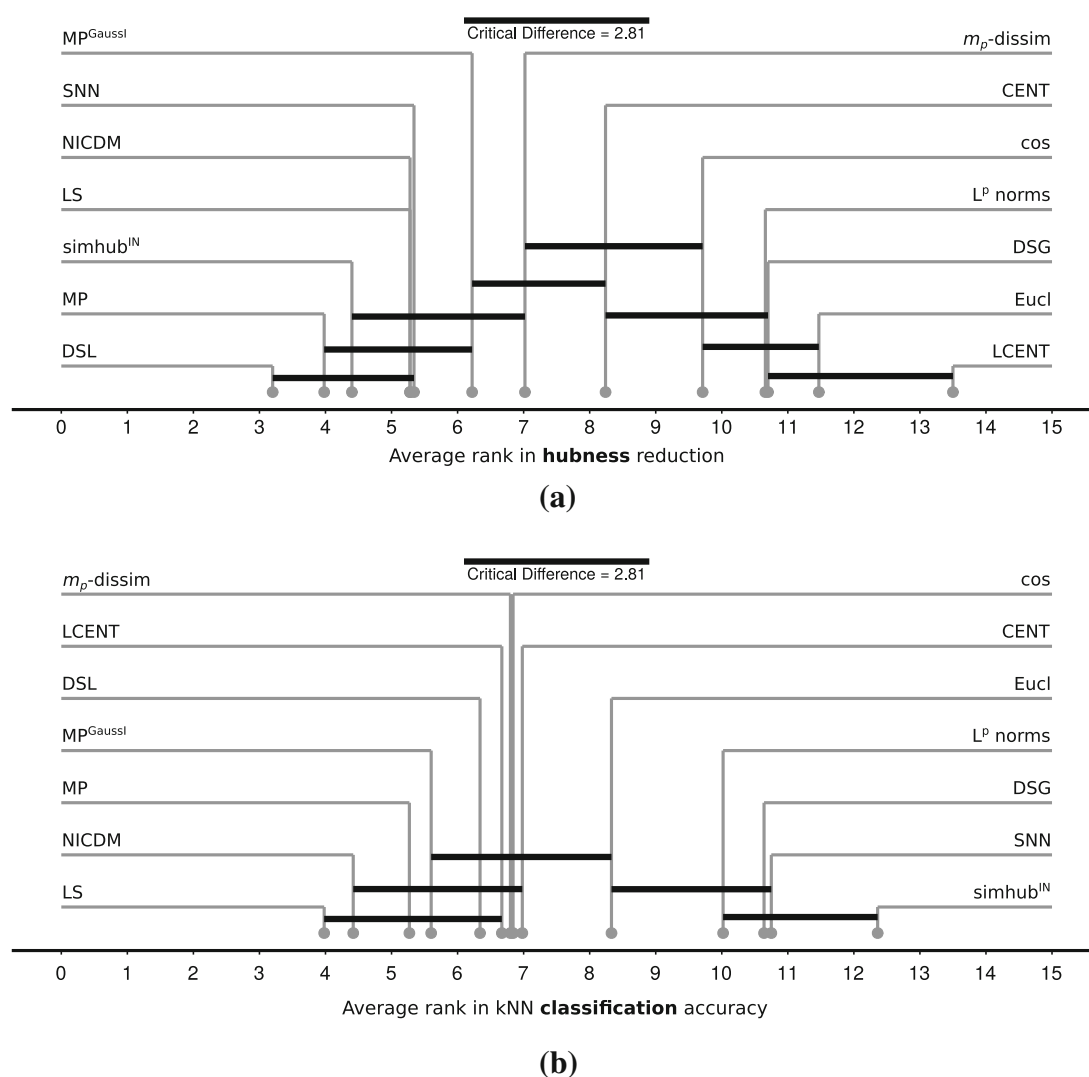


Fig. 3 Critical difference plots of average ranks in **a** hubness reduction and **b** nearest neighbor classification accuracy C^{kNN} for twelve hubness reduction methods plus two baseline distances (Eucl, cos). Low ranks indicate good performance. See Sect. 5 for details

The CD plot for nearest neighbor classification performance again comparing twelve methods plus two baseline distances (Euclidean, cosine) is shown in Fig. 3b. Regarding classification performance, the Nemenyi test reveals two coarse groups of methods: eight of them perform better or at least as well as the baselines, while four methods perform worse. Among the low rank (that is, ‘good’) methods, LS, NICDM, and MP yield significantly better results than the Euclidean baseline. Compared to both baselines, this is solely achieved by LS. The CD plot reveals, however, that the best six methods (aforementioned plus $\text{MP}^{\text{GaussI}}$, DSL, LCENT) are ranked within the critical distance, that is, no significant difference in their performance given the evaluation setup was found. Both centering variants as well as m_p -dissimilarity yield results very similar to the baselines. The high rank (that is, ‘bad’) methods are significantly worse than using cosine distances, and only $\text{simhub}^{\text{IN}}$ is also worse than using Euclidean distances.

Overall, local (LS, NICDM) and global scaling (MP) methods and DSL perform well both in regard to hubness reduction and classification performance, and appear to be the most promising hubness reduction methods. Shared neighbors methods (SNN, $\text{simhub}^{\text{IN}}$) yield distances with reduced hubness, but also impaired classification performance, indicating that their secondary distance space does not respect the semantic meaning of the primary distance space. Localized centering (LCENT) on the other hand showed reasonable classification performance, though it increased hubness on average, instead of reducing it. DSG and ℓ^p norms have hardly any influence on both evaluation measures.

5.2 Details of hubness reduction and classification performance

Figure 4 depicts the evaluation results in greater detail for six methods representing the different families of hubness reduction methods described in Sect. 3. Data sets in this plots are again ordered by ascending hubness ($S^{k=10}$) as measured before any hubness reduction, with low hubness data sets in the upper rows and high hubness data sets in the lower rows. Measures are absolute differences after and before hubness reduction.

We selected three methods achieving low ranks in both hubness reduction and classification performance (Fig. 4a). LS, MP, and DSL represent local scaling, global scaling, and density gradient flattening, respectively. Looking at the hubness reduction results, depicted as absolute changes in hubness in the left part of Fig. 4a, we can see that all three methods show only very small improvements for low hubness data sets, but do show substantial hubness reduction for higher hubness data sets, starting around data set 26 (lungcancer), which exhibited hubness of 1.2483 before reduction. Previous studies [49] on hubness reduction have shown a similar picture with reduction methods being effective above 10-occurrence skewness of 1.4. All three methods show highly comparable hubness reduction in general, with MP (dark gray bars) achieving negligibly stronger hubness reduction than LS (light gray bars) in many data sets, explaining the nonsignificant rank differences in Fig. 3a. $\text{DisSim}^{\text{Local}}$ (DSL, black bars) performs equally well as LS and MP on many data sets. Hubness reduction is nearly always on par with both scaling methods. Though DSL actually increases hubness for data sets 35 and 39 (farm ads, dorothea), coinciding with degraded accuracy, it achieves lowest hubness values for many others, resulting in the best rank in terms of hubness reduction in Fig. 3a.

Looking at absolute changes in classification performance in the right part of Fig. 4a, most significant changes are observed in high hubness data sets, starting around data set 25 (splice). Exceptions are two data sets of low to medium hubness (chronic kidney disease, no. 7, duke, no. 18), where DSL considerably degrades classification performance. On the other hand, LS increases classification performance for many of the high hubness data sets, with

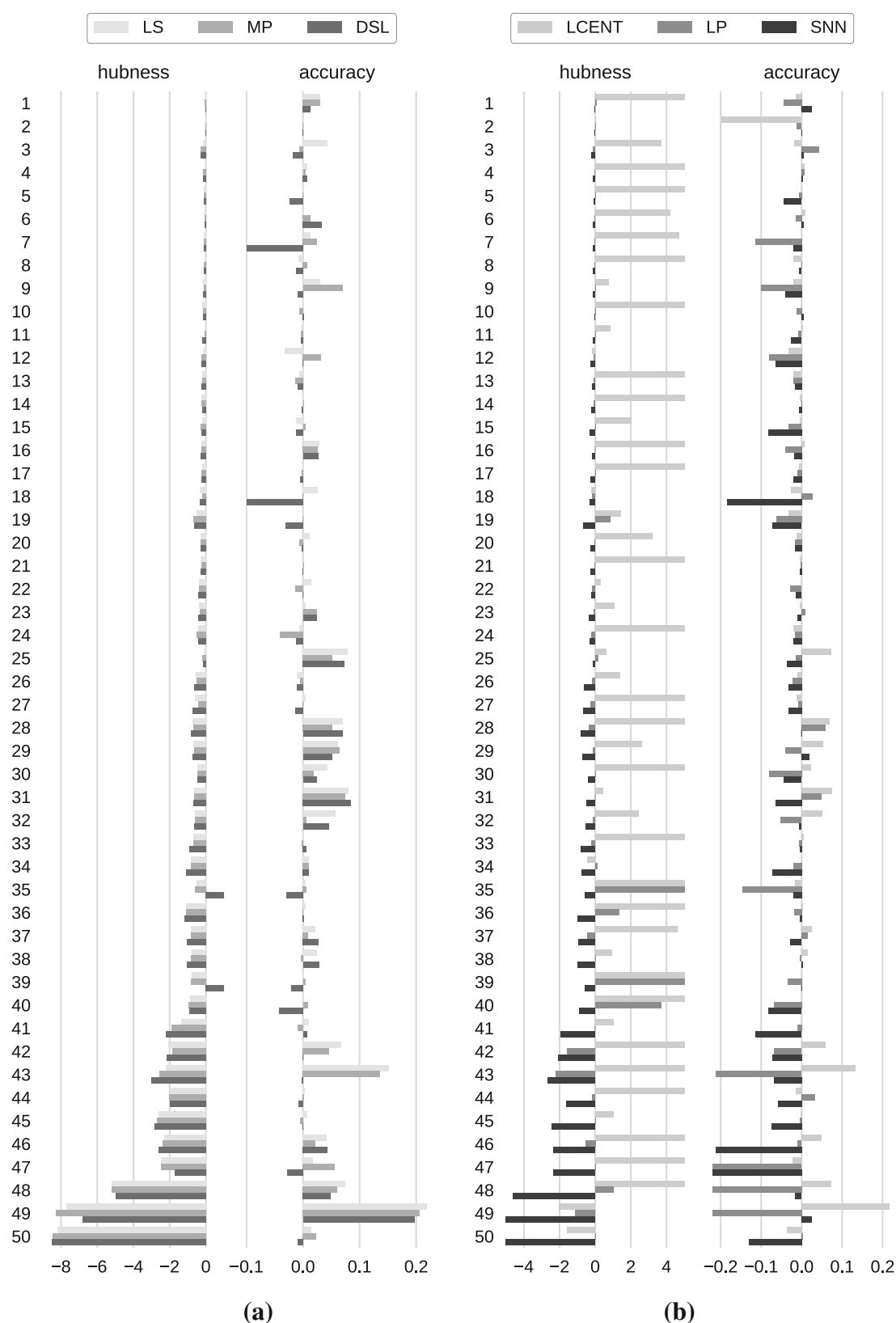


Fig. 4 Absolute change of hubness and kNN classification accuracy per data set for each hubness reduction method compared to baseline (cf. Sect. 4). Improvements are indicated by negative values of hubness change, and by positive values of accuracy change. Accuracy differences can range from -1 to 1, where, e.g., 0.1 refers to a performance increase of ten percent points over the baseline. Data sets are ordered by ascending hubness ($S^{k=10}$) as measured before any hubness reduction. **a** Successful methods. **b** Methods with mixed results

MP showing highly similar results. While LS often shows slightly improved classification accuracy compared to MP, these differences are nevertheless nonsignificant as already shown in Fig. 3b. DSL shows competitive classification performance compared to scaling methods for some high hubness data sets but is more often detrimental (e.g., farm ads, no. 35, dorothea, no. 39, CNAE-9, no. 40, sector, no. 47). While the degradation in classification for DSL coincides with an increase in hubness for data sets 35 and 39, there is no such connection for all other cases.

Overall, both scaling methods MP and LS consistently reduce hubness and often improve classification performance. They may safely be applied on data sets from low to high hubness, without any noticeable risk of performance degradation. DSL is sometimes competitive with LS and MP, but also shows a higher risk of decreased performance measures, hinting at density gradient flattening being less generally applicable than scaling.

Figure 4b shows three methods with mixed evaluation results. LCENT, ℓ^p norms, and SNN represent centering techniques, choosing alternative distance measures, and shared neighbors approaches. Looking at the hubness reduction results, depicted as absolute changes in hubness in the left part of Fig. 4b, LCENT (light gray bars) shows increased hubness for almost all data sets, except the highest hubness data sets 49 (c1ka-twitter) and 50 (dna). Shared nearest neighbors (SNN, black bars) on the other hand is able to reduce hubness in all medium to high hubness data sets. Application of ℓ^p norms (LP, dark gray bars) has mixed effects on high hubness data, with reduction of hubness for data sets 42 (DMOZ), 43 (Amazon commerce reviews) and 49 (c1ka-twitter), and considerable increase in hubness for data sets 35 (farm ads), 36 (gisette), 39 (dorothea) and 40 (CNAE-9).

Looking at absolute changes in classification performance in the right part of Fig. 4b, it can be seen that almost only LCENT (light gray bars) is able to improve accuracy on selected data sets, typically for those from the text domain, for example data sets 43 (Amazon commerce reviews) and 49 (twitter). Looking at the respective hubness performance, hubness is reduced in data set 49, but, strikingly, it is actually strongly increased in data set 43. LCENT can be a suitable choice for data sets from the text domain, with scaling methods LS and MP (see right side in Fig. 4a) still performing better. Choosing alternative ℓ^p norms (LP, dark gray bars) or shared nearest neighbors (SNN, black bars) shows virtually only degraded classification performance, especially for high hubness data. Hubness reduction with these methods does not seem to be sensible.

5.3 Neighborhood symmetry

Hubness negatively affects neighborhood symmetry as outlined in Sect. 2.1. Hubness reduction should therefore increase the proportion of symmetric k -nearest neighbor relations. Figure 5 depicts changes in neighborhood symmetry after hubness reduction compared to baseline in twelve subplots for the twelve hubness reduction methods. To explain what is shown in these twelve subplots, we first turn to the results for mutual proximity (MP), in the left subplot of the top row in Fig. 5. We show the percentage of symmetric nearest neighbor relations ($k = 10$) on the y-axis for all 50 data sets on the x-axis. To be more precise, we show the change in percentage of symmetric nearest neighbor relations when MP is applied relative to a baseline when no hubness reduction is done. These changes are shown as solid violet lines in case the percentage of symmetric nearest neighbor relations increases due to hubness reduction, and as a dashed orange line in case it decreases. Please note that the dot at one end of every line signifies the result achieved by MP, whereas the other end of a line signifies the baseline result. These ends show a clear trend of decreasing values from left to right: Since the data sets on the x-axis are ordered by their hubness values (refer to Table 3),

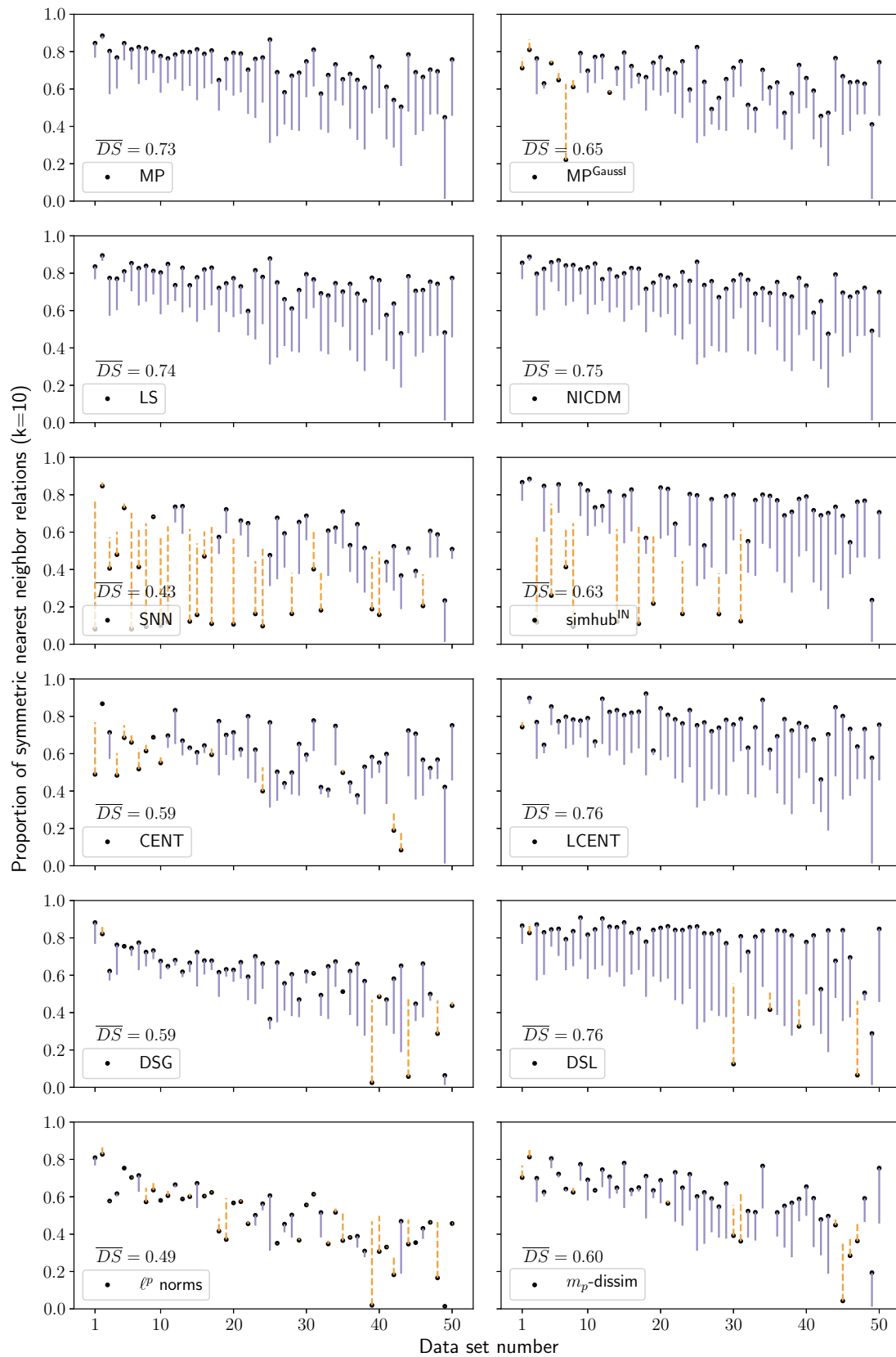


Fig. 5 Symmetric neighborhoods. Percentage of symmetric nearest neighbor relations ($k = 10$). Solid lines (violet) indicate improvement over the baseline, dashed lines (orange) indicate degradation. Filled circles represent symmetry using the indicated method, while the opposite end of the line represents the baseline which is identical for all subplots. $\overline{DS}$ denotes the average degree of symmetry. Baseline $\overline{DS} = 0.50$ (color figure online)

this indicates that the baseline measures create fewer symmetric nearest neighbor relations in data sets with higher hubness, as expected from theory. These baseline values are identical in all subplots. The subplot for MP clearly shows that the percentage of symmetric nearest neighbor relations is improved for all data sets. This improvement is stronger for high hubness data sets. On average, the proportion of symmetric relations $\overline{DS}$ increased from baseline 0.50 to 0.73 when using MP.

As a matter of fact, all local and global scaling methods consistently increase the number of symmetric neighborhood relations in nearly all cases (MP: $\overline{DS}=0.73$, MP^{GaussI} : $\overline{DS}=0.65$, LS: $\overline{DS}=0.74$, NICDM: $\overline{DS}=0.75$). They show comparable results among each other with the effect being weaker in case of MP^{GaussI} for some data sets. Modeling distance distributions with independent Gaussians may not approximate the real distributions in these cases. LCENT and DSL also improve neighborhood symmetry to a similar extent as scaling methods (both $\overline{DS}=0.76$). DSL achieves the highest symmetries for several data sets, but reduces symmetry in five cases, for example, data sets 35 (farm ads) and 39 (dorothea), which in addition also show increased hubness after application of DSL (cf. Fig. 4a). The global variants of centering (CENT) and DisSim (DSG) are on average much less successful in increasing symmetry (both $\overline{DS} = 0.59$). This is not surprising, as they were argued to reduce hubness in unimodal distributions, but real-world data sets are usually better described as mixtures of distributions. It is noteworthy that most cases of improved symmetry due to centering correspond to data sets from the text domain. Usage of alternative ℓ^p norms changes symmetry compared to baseline only for a few data sets (LP: $\overline{DS}=0.49$). Indeed, in some cases parameter selection yielded $p = 2$, reducing the metric to the Euclidean baseline. Small positive effects were found for the m_p -dissimilarity ($\overline{DS}=0.60$). SNN severely reduces symmetry for many data sets, predominantly among those with low to medium hubness, resulting in a reduced $\overline{DS}$ of 0.43. The SNN-variant simhub^{IN} also reduces neighborhood symmetry in several cases. However, it does increase symmetry in data sets with high hubness strongly, and on average to 0.63.

The results in this section are based on neighborhood sizes $k = 10$. Similar effects on neighborhood symmetry were observed with neighborhood ranges adaptive to data set sizes using $k = \frac{n}{10}$ (not shown).

5.4 Similarity to centroids

Objects highly similar to their (local) centroids may emerge as hubs (cf. Sect. 2.1). Reducing these similarities should therefore improve hubness. Centroids can be trivially computed from vector data or may be derived from distances based on metrics like the Euclidean norm. It is, however, not generally possible to calculate them from arbitrary dissimilarity matrices. We therefore use (local) medoids as proxies for their corresponding centroids, and measure correlation between k -occurrence and distance to the medoid. Reduced correlation should hint at reduced emergence of hubs.

Figure 6 depicts changes in correlation after hubness reduction compared to baseline analogously to Fig. 5 displaying neighborhood symmetry changes. To describe the plot in more detail, let us consider the left subplot in the fourth row, showing the results for CENT. Dot markers indicate the correlation between 10-occurrence and distance to local medoids on the y-axis for each data set on the x-axis. As expected, there is a trend of stronger negative correlations in data sets of higher hubness before hubness reduction. The average absolute value of correlation is denoted as $|\overline{r}|$ and given for all hubness reduction methods in their corresponding subplot. The horizontal line indicates the targeted correlation value of $r = 0$. CENT uses the inner product as similarity measure. After centering, the data centroid is a

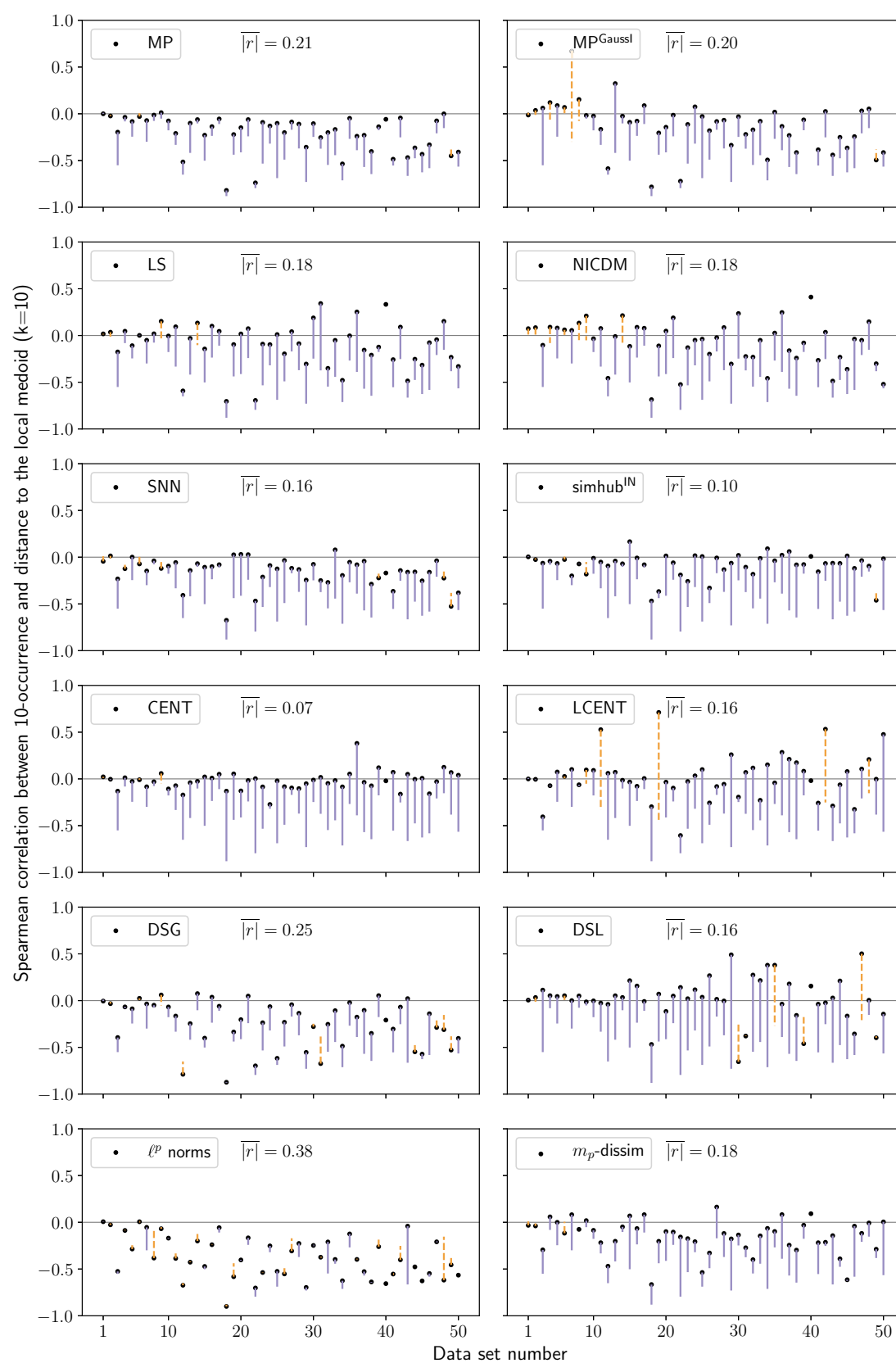


Fig. 6 Local medoids. Spearman correlation between the 10-occurrence of each object and the distance to their local medoids (k=10). Improvements and degradations are indicated analogously to Fig. 5. Changes are considered to be improvements, if the absolute value of correlation is reduced. $\overline{|r|}$ denotes the average of absolute values of correlation after hubness reduction. Baseline $\overline{|r|} = 0.38$

zero vector and, thus, all inner product similarities to the centroid are uniform zero [56]. Consequently, there is no correlation between these similarities and k -occurrence. Given the results of CENT with most correlations being very close to zero and the average correlation $|\overline{r}| = 0.07$, we assume that medoids are indeed suitable proxies for centroids.

Vertical lines reveal the change in correlation due to hubness reduction compared to baseline. That is, baseline correlations reside at the end of the lines opposite to the dot markers. Solid violet lines indicate improved correlations, which are closer to zero after hubness reduction. Dashed orange lines signalize degraded correlations with higher absolute values after hubness reduction. Consider for example the right subplot of the fourth row: In five cases, LCENT ‘overshoots’ and yields positive correlations of higher absolute value than the negative correlations before hubness reduction.

We find an average Spearman $|\overline{r}| = 0.38$ for the baseline not using any hubness reduction. All reduction methods are able to reduce this correlation. Weakest average correlations ($|\overline{r}| < 0.2$) are observed for LCENT, DSL, LS, NICDM and simhub^{IN}. All other methods, except for ℓ^p norms ($|\overline{r}| = 0.38$), create weak correlations as well. In case of SNN, this may be partly due to the fact, that its secondary distances can only take a low number of different discrete values (precisely the neighborhood size $k + 1$). Consequently, SNN often yields many equal distances, and random rankings among those distances reduce correlation.

Interestingly, the correlations are reduced not only when using spatial centrality-based methods, but also for those based on neighborhood symmetry, like for example MP or LS.

5.5 Association of evaluation criteria

We rank all methods and baseline measures by their success in hubness reduction (Fig. 3a), classification performance (Fig. 3b), degree of symmetric neighborhood relations (Fig. 5), and spatial centrality reduction (Fig. 6). Across all hubness reduction methods and baseline measures, symmetry ranks are highly correlated with accuracy ranks (Spearman $r = 0.72$, Fig. 7). That is, effectiveness in terms of increased symmetry corresponds well to the results in nearest neighbor classification. Ranks in reducing spatial centrality are more in line with

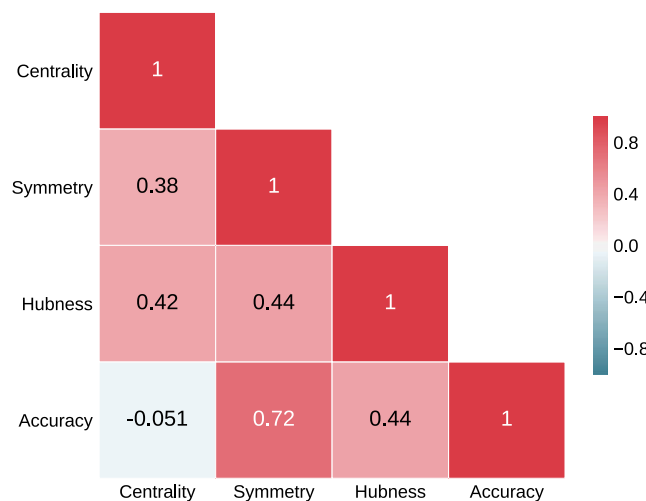


Fig. 7 Spearman correlation between evaluation criteria. Correlations are calculated from the ranks of twelve hubness reduction methods and two baseline measures given each criterion (see Sect. 5.5). ‘Centrality’ refers to the association between k -occurrence and distance to local medoids (Sect. 5.4), ‘Symmetry’ refers to the proportion of symmetric relations in nearest neighbor lists (Sect. 5.3), ‘Hubness’ is the skewness of the 10-occurrence distribution, and ‘Accuracy’ denotes the classification performance

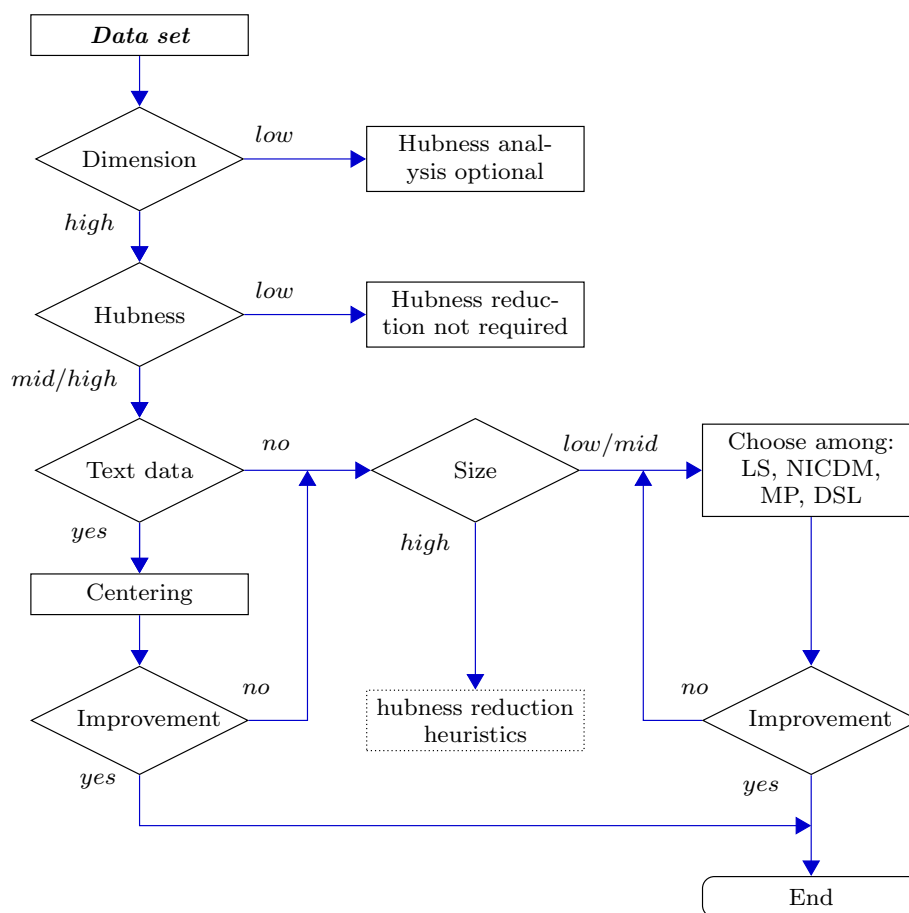


Fig. 8 Choosing appropriate hubness reduction methods

the ranks in hubness reduction (Spearman $r = 0.42$, Fig. 7), than with classification accuracy, for which we observe no rank correlation with spatial centrality ($r = -0.05$). The association between reduced spatial centrality and hubness reduction is, however, less clear than the association between strengthened neighborhood symmetry and increased classification performance. To give one concrete example, correlation between 10-occurrence and distance to local medoids is strongly decreased using CENT, but hubness is reduced only slightly, and in case of MP, vice versa (Fig. 6).

6 Discussion

We find that global and local scaling methods (MP, LS, NICDM) consistently improve performance for all evaluation measures over a wide range of data sets from various domains. This result is in line with the findings of a previous study [49]. Scaling methods achieve highest classification accuracy among the competing hubness reduction methods, and perform competitively to the best methods given the other evaluation measures. LS and NICDM are conceptually similar. While LS only considers the distance to one fixed neighbor, NICDM uses statistics over several neighbors. For this reason, we had expected higher stability of NICDM results over LS. Instead, both methods perform equally well. Carefully tuning their hyperparameters might have compensated for any instability. Density gradient flattening with DSL yields the best results in hubness reduction and neighborhood symmetry. Differ-

ences in classification performance between DSL and scaling methods are nonsignificant. We recommend using any of the four methods MP, LS, NICDM, and DSL for general hubness reduction. For large data sets, the cubic time complexity of MP is prohibitive, whereas LS, NICDM, and DSL scale only quadratically with the data set size. In the framework of mutual proximity, quadratic complexity can be achieved by approximating the empiric distance distribution with normal or Gamma distributions [49]. The approximation with normal distributions ($\text{MP}^{\text{GaussI}}$) yielded good results. Overall performance degradation compared to MP using empiric distributions is not significant, and might be caused by some data sets, for which independent Gaussians do not fit the true distance distributions well. $\text{MP}^{\text{GaussI}}$ may thus be used, when MP is too expensive in large data sets. Furthermore, distance distributions can be modeled using any continuous distribution. Given specific domain knowledge, other distributions may yield better MP approximations.

For very large data sets, algorithms with quadratic complexities are not applicable. Hubness reduction heuristics with subquadratic time and space requirements can be devised: LS, NICDM, and DSL require local neighborhood information. Their transformations could be accelerated using approximate nearest neighbor techniques. *Locality-sensitive hashing* [31] is commonly used for approximate search in high-dimensional spaces [1]. Since LSH requires vector data, different approaches are necessary for data sets only providing distances between objects. Sampling strategies could serve as an alternative for these cases, and may also be employed to reduce the complexity of mutual proximity. Heuristics based on these or other strategies would allow for hubness reduction in very large data sets. Evaluation of effectiveness and efficiency is yet to be performed, however.

Centering approaches show mixed results. They improve performance measures in several data sets from the text domain. Several other data sets are hardly influenced by centering, possibly due to its global nature: Hubs emerge close to the global mean only in case of unimodal data distributions [44]. Subtracting the mean does not eliminate spatial centrality in data sets with underlying multimodal distributions. The mechanism of hubness reduction behind centering does fail in such cases. LCENT similarities are calculated by subtracting *affinity* to local centroids. However, significant improvements over CENT were not observed in the evaluation. Centering seems to be applicable primarily to text data sets, but it does not outperform scaling methods or DSL in these cases. Due to its low cost, CENT can be applied to large data sets, and may thus be used for hubness reduction, when other methods are too expensive.

We do not recommend the other evaluated methods for hubness reduction. Using alternative ℓ^p norms or DSG does not yield improved performance measures. The formulation of DSG assumes all data to be generated from a unimodal probability distribution, an assumption presumably violated by many real-world data sets used in this study. This could explain the marked performance difference between the global and local DisSim variants, since DSL can handle mixtures of distributions by considering local neighborhoods. The shared neighbors approaches SNN and $\text{simhub}^{\text{IN}}$ reduce hubness, but fail to preserve, let alone improve data semantics. It has been argued previously that shared neighbor transformations cause information loss, because they only use rank information, and their codomain contains only $k + 1$ different values [20]. The data-dependent m_p -dissimilarity measure improves performance indicators compared to baseline, but does not yield results competitive with the best methods.

Given the results in Figs. 5, 6, 7, we observe that symmetric nearest neighbor relations have a stronger influence on classification performance than distances to medoids, which weakly corresponds to hubness. A possible explanation for these observations is that spatial centrality might not be the actual (or at least not the only) driving force in hub emergence. For

example, Low et al. describe hubness as an effect of density gradients due to non-uniform data distributions or boundary effects [39]. Spatial centrality-based methods may fail to reduce hubness and improve data semantics in such cases, unless they also flatten the density gradient. As opposed to this, asymmetric nearest neighbor relations are not a *source* of hubness, but a necessary *consequence* of skewed k -occurrences. Fixing detrimental effects of hubness one step later in the chain of causation might give methods based on neighborhood symmetry the advantage of independence of the primary cause of hubness. Verifying this hypothesis remains a task for future research.

Figure 8 serves as a reference for the interested reader. Based on some simple criteria, it guides through the hubness analysis workflow: Dimensionality and hubness measurements help decide, whether hubness reduction is indicated for given data. Data set size and application domain help decide which method to choose. Finally, alternatives are recommended if the selected hubness reduction method does not yield sufficient performance improvements.

On a side note, the difference between LCENT and LS has previously been described in terms of their theoretical motivation [27]: LCENT tries to reduce correlation between k -occurrence and local affinity, which is related to the distance to the local centroid/medoid. On the other hand, LS tries to make nearest neighbor relations more symmetric. Our empirical results do not support this distinction, since both methods achieve similar improvements both in terms of neighborhood symmetry and k -occurrence/medoid correlation.

7 Conclusion

In this paper, we presented a large-scale empirical evaluation of unsupervised hubness reduction methods. We analyzed hubness in terms of k -occurrence skewness, spatial centrality, and neighborhood symmetry before and after hubness reduction. Global and local scaling (MP, LS, NICDM) as well as density gradient flattening (DSL) improve these measures as well as data semantics (classification accuracy) over a wide range of data sets from various domains and may be considered state-of-the-art in hubness reduction. Global and localized centering are not as generally applicable, but can be successful for data from the text domain, with CENT being especially simple and inexpensive.

Future work will continue to investigate the impact of hubness on supervised and unsupervised learning methods beyond nearest neighbor classification. The development and evaluation of hubness reduction heuristics with subquadratic time and space complexity will allow to tackle large data sets.

Acknowledgements Open access funding provided by Austrian Science Fund (FWF). We would like to express our gratitude to Thomas Rattei for granting access to the Life Science Compute Cluster of the University of Vienna and Jan Schlüter for helpful comments on the manuscript. We also thank the authors of the *scikit-learn* [43] and *mlr* [6] machine learning libraries as well as all members of the scientific community sharing their data on public repositories. This research is supported by the Austrian Science Fund (FWF): P27082, P27703.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

References

1. Andoni A, Indyk P, Laarhoven T, Razenshteyn I, Schmidt L (2015) Practical and optimal LSH for angular distance. In: Cortes C, Lawrence ND, Lee DD, Sugiyama M, Garnett R (eds) *Advances in neural information processing systems*, vol 28. Curran Associates, Inc., Red Hook, pp 1225–1233
2. Aryal S, Ting KM, Washio T, Haffari G (2017) Data-dependent dissimilarity measure: an effective alternative to geometric distance measures. *Knowl Inf Syst* 53(2):479–506
3. Aucouturier JJ, Pachet F (2004) Improving timbre similarity: how high is the sky. *J Negat Results Speech Audio Sci* 1(1):1–13
4. Bellman RE (1961) *Adaptive control processes: a guided tour*. Princeton University Press, Princeton
5. Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization. *J Mach Learn Res* 13:281–305
6. Bischl B, Lang M, Kotthoff L, Schiffner J, Richter J, Studerus E, Casalicchio G, Jones ZM (2016) mlr: machine learning in R. *J Mach Learn Res* 17(170):1–5
7. Buza K, Nanopoulos A, Nagy G (2015) Nearest neighbor regression in the presence of bad hubs. *Knowl-Based Syst* 86:250–260
8. Camastra F, Staiano A (2016) Intrinsic dimension estimation: advances and open problems. *Inf Sci* 328:26–41
9. Cawley GC, Talbot NL (2010) On over-fitting in model selection and subsequent selection bias in performance evaluation. *J Mach Learn Res* 11:2079–2107
10. Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. *ACM Trans Intell Syst Technol (TIST)* 2(3):27:1–27:27
11. Chavarriaga R, Sagha H, Calatroni A, Digumarti ST, Trster G, Milln J del R, Roggen D (2013) The opportunity challenge: a benchmark database for on-body sensor-based activity recognition. *Pattern Recogn Lett* 34(15):2033–2042
12. Ciarelli PM, Salles EOT, Oliveira E (2010) An evolving system based on probabilistic neural network. In: *Proceedings of the eleventh Brazilian symposium on neural networks*, pp 182–187
13. Danziger SA, Baronio R, Ho L, Hall L, Salmon K, Hatfield GW, Kaiser P, Lathrop RH (2009) Predicting positive p53 cancer rescue regions using most informative positive (MIP) active learning. *PLoS Comput Biol* 5(9):1–12
14. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. *J Mach Learn Res* 7:1–30
15. Faddoul JB (2012) DMOZ web directory topics. URL <http://mldata.org/repository/data/viewslug/dmoz-web-directory-topics/>
16. Feldbauer R, Flexer A (2016) Centering versus scaling for hubness reduction. In: Villa AE, Masulli P, Rivero AJP (eds.) *25th International conference on artificial neural networks*, lecture notes in computer science, pp 175–183. Springer
17. Flexer A (2015) Improving visualization of high-dimensional music similarity spaces. In: *Proceedings of the 16th international society for music information retrieval (ISMIR) conference*, pp 547–553
18. Flexer A (2016) An empirical analysis of hubness in unsupervised distance-based outlier detection. In: *16th International conference on data mining workshops (ICDMW)*, pp 716–723. IEEE
19. Flexer A (2016) Hubness-aware outlier detection for music genre recognition. In: *Proceedings of the 19th international conference on digital audio effects*
20. Flexer A, Schnitzer D (2013) Can shared nearest neighbors reduce hubness in high-dimensional spaces? In: *IEEE 13th international conference on data mining workshops*, pp 460–467. IEEE
21. Flexer A, Schnitzer D (2015) Choosing ℓ^p norms in high-dimensional spaces based on hub analysis. *Neurocomputing* 169:281–287
22. Flexer A, Stevens J (2018) Mutual proximity graphs for improved reachability in music recommendation. *J New Music Res* 47(1):17–28
23. Francois D, Wertz V, Verleysen M (2007) The concentration of fractional distances. *IEEE Trans Knowl Data Eng* 19(7):873–886
24. Güvenir HA, Acar B, Demiröz G, Çekin A (1997) A supervised machine learning algorithm for arrhythmia analysis. In: *Proceedings of the computers in cardiology conference*, pp 433–436
25. Hara K, Suzuki I, Kobayashi K, Fukumizu K (2015) Reducing hubness: a cause of vulnerability in recommender systems. In: *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pp 815–818
26. Hara K, Suzuki I, Kobayashi K, Fukumizu K, Radovanović M (2016) Flattening the density gradient for eliminating spatial centrality to reduce hubness. In: *Proceedings of the 30th AAAI conference on artificial intelligence*, pp 1659–1665

27. Hara K, Suzuki I, Shimbo M, Kobayashi K, Fukumizu K, Radovanovic M (2015) Localized centering: reducing hubness in large-sample data. In: Proceedings of the 29th AAAI conference on artificial intelligence (AAAI), pp 2645–2651
28. Higuera C, Gardiner KJ, Cios KJ (2015) Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. *PLoS ONE* 10(6):e0129126
29. Hoyer PO, Henschel S, Sonnenburg S, Braun ML, Ong CS (2009) machine learning data set repository. URL <http://mldata.org/>
30. Hsu CW, Lin CJ (2002) A comparison of methods for multiclass support vector machines. *IEEE Trans Neural Netw* 13(2):415–425
31. Indyk P, Motwani R (1998) Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the thirtieth annual ACM symposium on theory of computing, pp 604–613. ACM
32. Jarvis RA, Patrick EA (1973) Clustering using a similarity measure based on shared near neighbors. *IEEE Trans Comput* 100(11):1025–1034
33. Jegou H, Harzallah H, Schmid C (2007) A contextual dissimilarity measure for accurate and efficient image search. In: IEEE conference on computer vision and pattern recognition, pp 1–8. IEEE
34. Knees P, Schnitzer D, Flexer A (2014) Improving neighborhood-based collaborative filtering by reducing hubness. In: Proceedings of international conference on multimedia retrieval, ICMR '14, pp 161–168
35. Kulis B (2013) Metric learning: a survey. *Found Trends Mach Learn* 5(4):287–364
36. Levina E, Bickel PJ (2005) Maximum likelihood estimation of intrinsic dimension. In: Saul LK, Weiss Y, Bottou L (eds) *Advances in neural information processing systems*, vol 17. MIT Press, Cambridge, pp 777–784
37. Lewis DD, Yang Y, Rose TG, Li F (2004) RCV1: a new benchmark collection for text categorization research. *J Mach Learn Res* 5:361–397
38. Lichman M (2013) UCI machine learning repository. URL <http://archive.ics.uci.edu/ml>
39. Low T, Borgelt C, Stober S, Nürnberger A (2013) The hubness phenomenon: Fact or artifact? In: Borgelt C, Gil MÁ, Sousa JM, Verleysen M (eds) *Towards advanced data analysis by combining soft computing and statistics*. Springer, Berlin, pp 267–278
40. McCallum A, Nigam K (1998) A comparison of event models for naive bayes text classification. In: Proceedings of the AAAI'98 workshop on learning for text categorization
41. Mesterharm C, Pazzani MJ (2011) Active learning using on-line algorithms. In: Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining, pp 850–858
42. Pagnotta F, Amran HM (2016) Using data mining to predict secondary school student alcohol consumption. <http://www3.dsi.uminho.pt/pcortez>
43. Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V et al (2011) Scikit-learn: machine learning in python. *J Mach Learn Res* 12:2825–2830
44. Radovanović M, Nanopoulos A, Ivanović M (2010) Hubs in space: popular nearest neighbors in high-dimensional data. *J Mach Learn Res* 11:2487–2531
45. Radovanović M, Nanopoulos A, Ivanović M (2015) Reverse nearest neighbors in unsupervised distance-based outlier detection. *IEEE Trans Knowl Data Eng* 27(5):1369–1382
46. Sakar BE, Isenkul ME, Sakar CO, Sertbas A, Gorgen F, Delil S, Apaydin H, Kursun O (2013) Collection and analysis of a Parkinson speech dataset with multiple types of sound recordings. *IEEE J Biomed Health Inf* 17(4):828–834
47. Schnitzer D, Flexer A (2015) The unbalancing effect of hubs on k-medoids clustering in high-dimensional spaces. In: International joint conference on neural networks (IJCNN), pp 1–8
48. Schnitzer D, Flexer A, Schedl M, Widmer G (2011) Using mutual proximity to improve content-based audio similarity. In: Proceedings of the 12th international society for music information retrieval conference, pp 79–84
49. Schnitzer D, Flexer A, Schedl M, Widmer G (2012) Local and global scaling reduce hubs in space. *J Mach Learn Res* 13(1):2871–2902
50. Schnitzer D, Flexer A, Schlüter J (2013) The relation of hubs to the doddington zoo in speaker verification. In: Proceedings of the 21st European signal processing conference (EUSIPCO), pp 1–5. IEEE
51. Schnitzer D, Flexer A, Tomašev N (2014) A case for hubness removal in high-dimensional multimedia retrieval. In: European conference on information retrieval, pp 687–692. Springer
52. Semeion: Semeion handwritten digit. Tech. rep., Semeion Research Center of Sciences of Communication, via Sersale 117, 00128 Rome, Italy Tattile Via Gaetano Donizetti, 1-3-5, 25030 Mairano (Brescia), Italy (2008)
53. Soundarapandian P (2015). URL http://archive.ics.uci.edu/ml/datasets/Chronic_Kidney_Disease
54. Stiglic G, Kokol P (2010) Stability of ranked gene lists in large microarray analysis studies. *J Biomed Biotechnol* 2010:616358

55. Sun J, Yang Z, Wang P, Liu S (2010) Variable length character n-gram approach for online writeprint identification. In: Proceedings of the international conference on multimedia information networking and security, pp 486–490
56. Suzuki I, Hara K, Shimbo M, Saelens M, Fukumizu K (2013) Centering similarity measures to reduce hubs. In: Proceedings of the 2013 conference on empirical methods in natural language processing, 13: 613–623
57. Tomašev N (2015) Taming the empirical hubness risk in many dimensions. In: Proceedings of the 15th SIAM international conference on data mining (SDM), pp 1–9. SIAM
58. Tomašev N, Brehar R, Mladenović D, Nedevski S (2011) The influence of hubness on nearest-neighbor methods in object recognition. In: IEEE international conference on intelligent computer communication and processing (ICCP), pp 367–374
59. Tomašev N, Mladenović D (2014) Hubness-aware shared neighbor distances for high-dimensional k-nearest neighbor classification. *Knowl Inf Syst* 39(1):89
60. Tomašev N, Radovanović M, Mladenović D, Ivanović M (2011) The role of hubness in clustering high-dimensional data. *Adv Knowl Discov Data Min* 6634:183–195
61. Vanschoren J, van Rijn JN, Bischl B, Torgo L (2013) Openml: networked science in machine learning. *SIGKDD Explor* 15(2):49–60
62. Vincent E, Gkiokas A, Schnitzer D, Flexer A (2014) An investigation of likelihood normalization for robust ASR. In: Proceedings of the annual conference of the international speech communication association
63. Wang JY (2002) Application of support vector machines in bioinformatics. Master's thesis, National Taiwan University
64. Zelnik-Manor L, Perona P (2005) Self-tuning spectral clustering. *Adva Neural Inf Process Syst* 17:1601–1608



Roman Feldbauer holds a B.Sc. degree in Microbiology and Genetics, and an M.Sc. degree in Computational Science (Springer BestMasters award 2015). He is a Ph.D. candidate at the University of Vienna and Junior Researcher at the Austrian Research Institute for Artificial Intelligence. Roman has published papers in the fields of neuroscience, computational biology, and artificial intelligence. Recently, his main research interest has been machine learning-based approaches for life science.



Arthur Flexer is a Senior Researcher and Project Manager at the Intelligent Music Processing and Machine Learning Group at the Austrian Research Institute for Artificial Intelligence, Vienna. His research interests include machine learning and intelligent music processing. Flexer holds a Ph.D. in psychology from the University of Vienna.

3. Development of approximate hubness reduction

The main goal of this dissertation is demonstrating the effects of hubness reduction in large protein similarity networks. Any algorithm designed for *large* data sets should not exceed linear or log-linear complexity with respect to data set size, ideally for both compute and memory requirements. Otherwise, the project time frame and/or budget will eventually be exceeded for sufficiently large data. Let us consider hubness reduction in the SIMAP2 protein similarity network as a motivating example. The state-of-the-art hubness reduction algorithms identified in Chapter 2 calculate *secondary* distances, that is, they build upon a measure of (dis-)similarity, and modify them according to a set of rules. Naive implementations, therefore, require at least quadratic time and space with respect to the number of protein sequences for calculating and storing the all-against-all sequence (dis-)similarity matrix. While the SIMAP2 database provides the precomputed Smith-Waterman similarity scores of approximately 30 million protein sequences¹, merely holding the matrix in secondary memory for fast random access has prohibitive RAM requirements².

Hubness reduction in SIMAP2, therefore, requires strategies to reduce algorithmic complexity, which are developed, implemented, and evaluated in the second work package, confirming Hypothesis H₃. The resulting *approximate hubness reduction* methods are investigated in a case study of transport mode detection using accelerometer data from mobile devices. After establishing the methods' functionality, they are implemented in a `scikit-learn`-compatible software package (see Chapter 4).

Published work Copyright © 2018 IEEE. Reprinted, with permission, from Feldbauer, R., Leodolter, M., Plant, C., and Flexer, A. *Fast Approximate Hubness Reduction for Large High-Dimensional Data*, 2018 IEEE International Conference on Big Knowledge (ICBK), Singapore, 2018, pp. 358-367.

Contributions RF and AF designed the study. RF conceived and implemented the algorithms, conducted the experiments, and analyzed the results. ML prepared the case study, and also analyzed results. RF wrote the manuscript draft. ML, CP, and AF provided feedback on the manuscript. AF and CP supervised the work.

¹SIMAP2 compute time: ≈ 10 million CPU core hours (Vienna Scientific Cluster 3, <http://typo3.vsc.ac.at/systems/vsc-3/>, accessed 2020/05/28)

²SIMAP2 memory example: ≈ 8 TiB (30 M sequences, FP16, 99.5% sparsity, excluding sparse matrix overhead)

Fast approximate hubness reduction for large high-dimensional data

Roman Feldbauer^{*¶}, Maximilian Leodolter[†], Claudia Plant^{‡§}, Arthur Flexer^{*}

^{*} Austrian Research Institute for Artificial Intelligence (OFAI), Vienna, Austria

[†] Austrian Institute of Technology, Vienna, Austria

[‡] Faculty of Computer Science, University of Vienna, Vienna, Austria

[§] ds:UniVie, University of Vienna, Vienna, Austria

[¶] Department of Microbiology and Ecosystem Science, University of Vienna, Vienna, Austria

roman.feldbauer@ofai.at, maximilian.leodolter@ait.ac.at, claudia.plant@univie.ac.at, arthur.flexer@ofai.at

Abstract—High-dimensional data mining is challenging due to the “curse of dimensionality”. Hubness reduction counters one particular aspect of the dimensionality curse, but suffers from quadratic algorithmic complexity. We present approximate hubness reduction methods with linear complexity in time and space, thus enabling hubness reduction for large data for the first time. Furthermore, we introduce a new hubness measure especially suited for large data, which is, in addition, readily interpretable. Experiments on synthetic and real-world data show that the approximations come at virtually no cost in accuracy in comparison with full hubness reduction. Finally, we demonstrate improved transport mode detection in massive mobility data collected with mobile devices as concrete research application. All methods are made publicly available in a free open source software package.

Index Terms—curse of dimensionality, high-dimensional data mining, hubness, linear complexity, interpretability, smartphones, transport mode detection

I. INTRODUCTION

Data mining in high-dimensional spaces is often challenging due to a number of phenomena commonly referred to as “curse of dimensionality” [1]. Concentration of distances is one of the more well-known aspects [2]: Distances between all pairs of data objects become increasingly similar with increasing dimensionality, thereby losing their discriminative power. Many popular distance measures are known to concentrate, including general ℓ^p and fractional norms, cosine similarities, dynamic time warping-based scores, and others [2]–[6].

Hubness is a related, albeit less known phenomenon [7]. In high but finite dimensional regimes, some objects have on average smaller distances to all others. These central objects have a high probability of becoming *hubs*, that is, objects that are unwontedly often among the nearest neighbors of many other objects in the data set. Simultaneously, *antihub* objects are extruded from neighborhood lists. As a consequence, neighborhood relations in high-dimensional spaces are often asymmetric: Any hub object x is nearest neighbor to many other objects, which, however cannot all be the nearest neighbor of x . Hubs are known to propagate their information, for example class labels, (too) widely in distance space, while information

carried by antihubs is essentially lost. As a consequence, many learning algorithms based on distances may be impaired by hubness. The literature contains many examples of detrimental effects of hubness to classification, clustering, visualization, outlier detection, and other data mining tasks [7]–[11].

Several hubness reduction methods have been proposed to mitigate detrimental effects of hubness. Three methods have been identified as state-of-the-art in terms of reducing hubness and improving learning performance in a recent review [12]. Mutual proximity (MP, [13]) and local scaling (LS, [14]) reduce hubness by repairing asymmetric neighborhood relations. DisSimLocal (DSL, [15]) flattens density gradients, which have been proposed as an alternative source of hubness [16]. All three methods transform *primary distances* to *secondary distances*. MP and LS may be applied to any dissimilarities, while DSL is restricted to (squared) Euclidean distances.

High computational complexity is a major restriction of secondary distance-based algorithms. Assuming hubness reduction methods require square distance matrices as input, the complexity of computing and storing distances between all pairs of objects can trivially be identified as $\Omega(n^2)$ in terms of both time and space, where n is the number of objects in the data set. Both quadratic time and space complexity are prohibitive for large data sets, essentially restricting hubness reduction to small data sets until now. In order to support hubness reduction in linear time and space complexity, we propose (i) a new hubness measure which enables reliable estimation of hubness even from small samples, and (ii) approximation strategies for established hubness reduction methods, both of which are required for successful hubness analysis and reduction of massive data. Extensive experiments demonstrate that both approaches support fast and powerful hubness reduction in large and high-dimensional data.

Hubness measures and reduction methods are briefly reviewed in Section II. Our main contributions are presented in Section III. The proposed methodology is evaluated in Section IV. Successful approximate hubness reduction is demonstrated for a real-world application in Section V, showing improved transport mode detection based on massive mobile phone sensor data. We discuss our findings in Section VI and conclude in Section VII, where we also point to future work.

This research is supported by the Austrian Science Fund (FWF): P27082 and P27703.

II. RELATED WORK

A. Hubness measures

Several hubness measures have been proposed so far, most of which are based on the notion of k -occurrence. Let $D \subseteq \mathbb{R}^m$ be a non-empty data set containing n objects in m -dimensional space. The k -occurrence $O^k(x)$ of an object $x \in D$ counts, how often x occurs among the k -nearest neighbors of all other objects $D \setminus x$ indexed in the database. In low dimensional spaces, k -occurrence distributions typically are consistent with binomial distributions [7]. For example, assuming uniform i.i.d. data, this is in line with the intuition that all objects should be among the nearest neighbors of other objects approximately equally often. With increasing dimensionality, k -occurrence distributions are, however, increasingly skewed to the right [7]. This skewness is often used as a measure for hubness:

$$S^k = \mathbb{E}[(O^k - \mu_{O^k})^3] / \sigma_{O^k}^3, \quad (1)$$

where μ and σ indicate mean and standard deviation, respectively. The mean of a k -occurrence distribution is exactly k .

Computing the k -occurrence requires $\Omega(n)$ time for calculating all distances and partitioning. The measure defined above incorporates the k -occurrences of all objects, and thus requires $\Omega(n^2)$ time. The complexity can be trivially reduced by sampling, and calculating estimates of the above. The sample size may be varied *ad libitum* to approximate the true value. However, this is problematic in high dimensions, which we demonstrate in Section III-A, where we also present a new hubness measure overcoming these issues.

B. Hubness reduction methods

A recent survey recommends three methods for hubness reduction [12]. Let us briefly recall the formulations of full hubness reduction methods (quadratic complexity), before introducing approximate variants (linear complexity).

Local scaling (LS, [14]) transforms primary distances $d_{x,y}$ between objects x and y with local neighborhood information:

$$\text{LS}^k(d_{x,y}) = 1 - \exp(-d_{x,y}^2 / (r_x^k r_y^k)). \quad (2)$$

The scaling factors r_x^k and r_y^k refer to the distance of objects x and y to their k -th nearest neighbors, respectively.

Mutual proximity (MP, [13]) considers distances between all pairs of objects in a probabilistic way. Primary distances $d_{x,i \in \{1, \dots, n\} \setminus x}$ between an object x and all other objects are modeled with some distribution. Let X be the resulting random variable (analogously for random variable Y and object y), and P the joint probability density function, then

$$\text{MP}(d_{x,y}) = P(X > d_{x,y} \cap Y > d_{y,x}), \quad (3)$$

that is, the joint probability of x and y being nearest neighbors to each other. The complement $1 - \text{MP}$ yields secondary distances. Here, we model primary distances with independent normal distributions $X \sim \mathcal{N}(\hat{\mu}_x, \hat{\sigma}_x^2)$, and $Y \sim \mathcal{N}(\hat{\mu}_y, \hat{\sigma}_y^2)$, so that secondary distances are calculated as

$$\text{MPG}(d_{x,y}) = SF(d_{x,y}, \hat{\mu}_x, \hat{\sigma}_x^2) \cdot SF(d_{y,x}, \hat{\mu}_y, \hat{\sigma}_y^2), \quad (4)$$

with the survival function $SF(d, \mu, \sigma^2) = 1 - CDF(d, \mu, \sigma^2)$.

Two dissimilarity measures were introduced in [15] to reduce hubness in (squared) Euclidean spaces. The local variant DSL rescales distances using local centroids $c^k(\cdot)$:

$$\text{DSL}^k(x, y) = \|x - y\|_2^2 - \|x - c^k(x)\|_2^2 - \|y - c^k(y)\|_2^2 \quad (5)$$

The local centroids are estimated as $c^k(x) = \frac{1}{k} \sum_{x' \in \text{kNN}(x)} x'$, where $\text{kNN}(x)$ is the set of k -nearest neighbors of x .

C. Complexity of hubness reduction

Two factors dominate the complexity of hubness reduction. First, calculation of the primary distance between a pair of objects is typically in $\mathcal{O}(m)$, that is, linear in dimensionality. This holds for Euclidean and general ℓ^p distances. For dynamic time warping-based distances, near-optimal alignments can also be obtained in $\mathcal{O}(m)$, while exact alignments require $\mathcal{O}(m^2)$ [17].

Without loss of generality, assume the database is split into index $I \subset D$ and queries $Q \subset D$, with $n_{\text{indexed}} := |I|$ and $n_{\text{query}} := |Q|$. The set sizes are further determined by a split ratio $p \in]0, 1[$, so that $n_{\text{indexed}} = p \cdot n$, $n_{\text{query}} = (1 - p)n$, and $I \cup Q = D$, similarly e.g. to an evaluation scenario, where data is split into training and test sets. Calculating distances between each query object and all indexed objects thus requires $\Omega(n_{\text{query}} n_{\text{indexed}} m) = \Omega(n^2 m)$ time.

The second dominating factor is exact neighbor search. Hubness reduction methods often require knowledge of local neighborhoods. For example, consider scaling factors r^k in (2), or local centroids c^k in (5). Probing an object's neighborhood requires partitioning its distance list at the k -th nearest neighbor. $\Theta(n_{\text{indexed}}) = \Theta(n)$ time is required to find k nearest neighbors, for example using the introselect algorithm. Neighbor search thus amounts to $\Theta(n^2)$ for the whole data set.¹

The quadratic complexity of hubness reduction methods is prohibitive for large data sets. To this end, we present approximate hubness reduction with linear complexity with respect to the data set size n in Sections III-B and III-C.²

III. APPROXIMATE HUBNESS REDUCTION METHODS

A prototypical hubness analysis pipeline was described in [12], and must answer two questions: (i) To what extent is the data set at hand affected by hubness? (ii) Can hubness reduction improve learning performance? The pipeline thus comprises two essential steps: hubness estimation, and hubness reduction. To perform hubness analysis in linear time and space, neither step must exceed linear complexity. To this end we present approximation strategies for hubness estimation and hubness reduction in linear time and space. In combination with a new hubness measure, this enables hubness analysis in large data sets.

¹It is unnecessary to sort complete distance lists, which would require $\Theta(n \log n)$ time for each object.

²We assume dimension m to be fixed for each data set. Lower dimensional embeddings could be used additionally to the methodology suggested below to reduce complexity also w.r.t. dimension m . They are, however, out of scope for this work. Therefore, constant $\mathcal{O}(m)$ will be omitted henceforth.

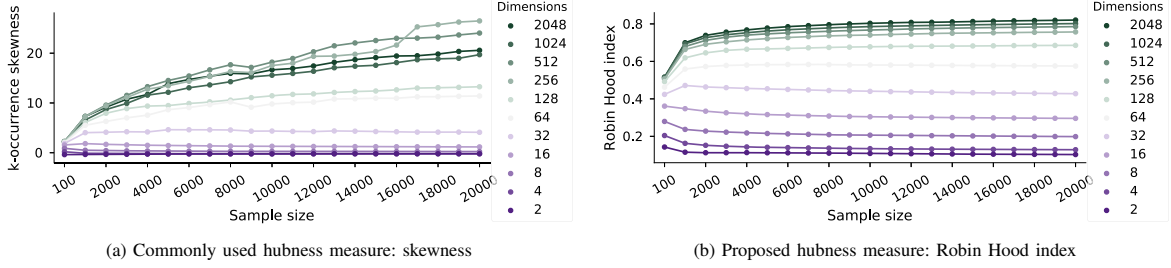


Fig. 1. Hubness measure dependency on sample size (x-axis), and dimensionality (lines). Each line in the plots constitutes one experiment of a MADELON-like data set (cf. Section IV-C) with 20 000 vectors in m -dimensional space, with $m \in \{2^1, 2^2, \dots, 2^{11}\}$. Hubness is measured in random subsamples with sizes between 100 and 19 000 (bootstrapped ten times, averaged). (a) Skewness of k -occurrence histogram, (b) Robin Hood index of k -occurrence histogram.

A. Approximate hubness estimation

The sampling strategy for estimating hubness in linear time as described in Section II-A is problematic in high dimensions. We demonstrate issues of this strategy in synthetic data sets. For a motivating example, we consider clustered Gaussian data (MADELON-like, as described in Section IV-C). Fig. 1a shows the effect of sampling on k -occurrence skewness (k -occurrence parameter $k = 10$ is used throughout the paper). Values increase with increasing dimensionality as expected. In low dimensions (up to 16 or 32), skewness in small samples approximates skewness in large samples well. In high dimensions (64 and higher), however, convergence to the true value is slow. This is undesirable for fast hubness estimation, which should ideally be accurate already at small sample sizes.

To counter this, we propose a new hubness measure borrowed from socio-economics. The Robin Hood index (also known as Ricci-Schutz/Piètra/Hoover index) is traditionally used to measure income inequality [18]. We use it for measuring k -occurrence inequality:

$$\mathcal{H}^k = \frac{1}{2} \frac{\sum_{x \in D} |O^k(x) - \mu_{O^k}|}{(\sum_{x \in D} O^k(x)) - k} = \frac{\sum_{x \in D} |O^k(x) - k|}{2k(n-1)} \quad (6)$$

Fig. 1b shows the behavior of the new hubness measure on the same synthetic data used before. Again, we observe a trend of increasing values with increasing dimensionality, as expected and as desired. There is, however, only little dependence on sample sizes. That is, values converge quickly to the ‘true’ values (as observed in the full data set). Similar behavior of both hubness measures was observed in i.i.d. uniform and normal data as well (not shown).

In addition, the Robin Hood index is more easily accessible for interpretation than the skewness measure. Values fall in the fixed interval $[0, 1]$, where $\mathcal{H}^k = 0$ indicates that all objects are equally often among the nearest neighbors of others ($\forall x \in D : O^k(x) = k$), whereas $\mathcal{H}^k = 1$ means that a single hub is nearest neighbor to all objects ($\exists x \in D : O^k(x) = n - 1$). In other words, the Robin Hood index answers the question: What share of ‘nearest neighbor slots’ must be redistributed to achieve k -occurrence equality among all objects?

B. Instance selection for approximate hubness reduction

Reducing the number of objects indexed in the database is a straight-forward approach to achieve linear time and space complexity. Selecting a fixed number $s \ll n_{\text{indexed}}$ representative objects yields $\mathcal{O}(n_{\text{query}}s)$ query time and memory usage. Varying s trades off time and space requirements versus accuracy. We consider the following instance selection algorithms:

- *Random sampling* is fast and easy to use. It may be used in unsupervised fashion, or combined with class labels for stratification. Random sampling is rather trivial, and will serve as baseline here.
- *k-means++* is k-means clustering with ‘careful seeding’ [19]. Seed cluster centers are selected by an initialization strategy so that clustering converges in fewer iterations. We use this strategy for unsupervised instance selection, expecting to obtain better representatives than by random sampling.
- Other popular instance selection methods, such as *DROP3*, *ICF*, and related methods, use supervised algorithms [20]. We restrict this work to unsupervised techniques, however.

Algorithm 1 describes approximate hubness reduction with generic instance selection and the example of local scaling. Assume `selectInstances()` yields instances according to any one of the above mentioned methods. Within this algorithmic framework hubness reduction methods are easily exchangeable. For details, see Section III-C.

Instance selection essentially drops many objects from the database. While this reduces storage footprint to $\mathcal{O}(sm)$, information loss occurs, which may deteriorate learning performance. To overcome this issue, we present a second approximation strategy for hubness reduction with linear complexity.

C. Approximate nearest neighbors for hubness reduction

The complexity of hubness reduction may also be reduced by relaxing the requirement of exact neighbor search (cf. Section II-C). Approximate nearest neighbor (ANN) search retrieves true nearest neighbors with high recall, but allows for some errors. This relaxation makes sublinear query time per

Algorithm 1 Approximate local scaling with instance selection

Input:
 k $\triangleright$ nearest neighbor parameter
 s $\triangleright$ Number of selected instances
 I, Q $\triangleright$ Indexed objects, query objects, respectively
 r_I, r_Q $\triangleright$ Empty vectors of sizes n_{indexed} , n_{query} , resp.

Output:
 distLS $\triangleright$ Sparse matrix of shape $n_{\text{query}} \times n_{\text{indexed}}$, $\mathcal{O}(n_{\text{query}}s)$

```

1: procedure FIT
2:    $\text{selectedSample} \leftarrow \text{SELECTINSTANCES}(I, s)$   $\triangleright \mathcal{O}(n_{\text{indexed}}m)$ 
3:   for all  $y \in I$  do
4:      $r_I(y) \leftarrow \text{K-TH-NN}(y, \text{selectedSample})$ 
5: procedure TRANSFORM
6:   for all  $x \in Q$  do  $\triangleright \mathcal{O}(n_{\text{query}}sm)$ 
7:      $r_Q(x) \leftarrow \text{K-TH-NN}(x, \text{selectedSample})$ 
8:     for all  $y \in \text{candidates}$  do  $\triangleright \mathcal{O}(sm)$ 
9:        $\text{dist} \leftarrow \text{PRIMARY DISTANCE}(x, y)$ 
10:       $\text{distLS}(x, y) \leftarrow \text{LS}(\text{dist})$ 
11: return  $\text{distLS}$ 

```

object possible. A plethora of ANN methods has been developed, most of which are based on tree algorithms [21], locality-sensitive hashing [22], product quantization [23], or proximity graphs [24]. ANN techniques have previously been suggested for hubness estimation [25], and hubness reduction [12]. To the best of our knowledge, this has not yet been evaluated or implemented in related software packages. We strive to fill this gap with the research presented here.

Combining ANN with hubness reduction yields a filter-and-refine methodology. For each object some $s \ll n_{\text{indexed}}$ approximate nearest neighbors are filtered from the database in sublinear time with respect to n_{indexed} . In the refinement step, hubness reduction is performed on these candidate neighbors, effectively reordering the approximate neighborhood. Assuming hubness reduction is then in $\mathcal{O}(s)$ per object rather than in $\mathcal{O}(n_{\text{indexed}})$, linear time complexity is obtained for the complete set of query objects.

Algorithm 2 exemplarily describes approximate local scaling with ANN techniques. Comments on the right-hand side ($\triangleright$) indicate time spent in each loop, including nested loops, and subroutines. The `fit()` procedure initially sets up the ANN index. Assume `fitANNIndex()` creates this index in linear time, and `k-th-ANN(y)` returns the distance between y and its k -th approximate nearest neighbor in sublinear time³. LS scaling factors for all indexed objects are stored in vector r_I .

Query objects are handled in the `transform()` procedure. In the filtering step, `rangeANN()` returns the s approximate nearest neighbors ('candidates') for each query object x . Primary distances are calculated between x and all candidates. Subsequently, these distances are transformed to secondary distances by LS in the refinement step. The order of candidate neighbors may have changed due to hubness reduction. Secondary distances are stored in a sparse matrix with shape $n_{\text{query}} \times n_{\text{indexed}}$ and $n_{\text{query}}s$ nonzero elements, which determines the memory requirements of $\mathcal{O}(n_{\text{query}}s)$.

³ This holds for LSH, HNSW, among other ANN techniques. Index creation scales logarithmically in case of HNSW. For details please refer to related publications and documentation.

Algorithm 2 Approximate local scaling with ANN

Input:
 k $\triangleright$ nearest neighbor parameter
 s $\triangleright$ Candidate neighborhood size
 I, Q $\triangleright$ Indexed objects, query objects, respectively
 r_I, r_Q $\triangleright$ Empty vectors of sizes n_{indexed} , n_{query} , resp.

Output:
 distLS $\triangleright$ Sparse matrix of shape $n_{\text{query}} \times n_{\text{indexed}}$, $\mathcal{O}(n_{\text{query}}s)$

```

1: procedure FIT
2:    $\text{annIndex} \leftarrow \text{FITANNINDEX}(I)$   $\triangleright \mathcal{O}(n_{\text{indexed}}m)$ 
3:   for all  $y \in I$  do
4:      $r_I(y) \leftarrow \text{K-TH-ANN}(y, \text{annIndex})$ 
5: procedure TRANSFORM
6:   for all  $x \in Q$  do  $\triangleright \mathcal{O}(n_{\text{query}}sm)$ 
7:      $\text{candidates} \leftarrow \text{RANGEANN}(s, x, \text{annIndex})$ 
8:      $r_Q(x) \leftarrow \text{K-TH-ANN}(x, \text{annIndex})$ 
9:     for all  $y \in \text{candidates}$  do  $\triangleright \mathcal{O}(sm)$ 
10:       $\text{dist} \leftarrow \text{PRIMARY DISTANCE}(x, y)$ 
11:       $\text{distLS}(x, y) \leftarrow \text{LS}(\text{dist})$ 
12: return  $\text{distLS}$ 

```

Approximate mutual proximity (Algorithm 3) bears close resemblance to the algorithm above. For each object, a normal distribution is estimated from its distances to t indexed objects ($t = 30$ was suggested in [13]). MPG incorporates these Gaussians as described in (4).

Algorithm 3 Approximate mutual proximity (Gaussian)

Input:
 s $\triangleright$ Candidate neighborhood size
 t $\triangleright$ Sample size for estimating μ, σ
 I, Q $\triangleright$ Indexed objects, query objects, resp.
 μ_I, μ_Q $\triangleright$ Empty vectors of sizes n_{indexed} , n_{query} , resp.
 σ_I, σ_Q $\triangleright$ Empty vectors of sizes n_{indexed} , n_{query} , resp.

Output:
 distMPG $\triangleright$ Sparse matrix of shape $n_{\text{query}} \times n_{\text{indexed}}$, $\mathcal{O}(n_{\text{query}}s)$

```

1: procedure FIT
2:    $\text{annIndex} \leftarrow \text{FITANNINDEX}(I)$ 
3:    $\text{samples} \leftarrow \text{RANDOM SAMPLE}(t, I)$ 
4:   for all  $y \in I$  do  $\triangleright \mathcal{O}(n_{\text{indexed}}m)$ 
5:      $\text{distances} \leftarrow \text{PRIMARY DISTANCE}(y, \text{samples})$ 
6:      $\mu_I(y) \leftarrow \text{MEAN}(\text{distances})$ 
7:      $\sigma_I(y) \leftarrow \text{STANDARDDEVIATION}(\text{distances})$ 
8: procedure TRANSFORM
9:   for all  $x \in Q$  do  $\triangleright \mathcal{O}(n_{\text{query}}sm)$ 
10:     $\text{distances} \leftarrow \text{PRIMARY DISTANCE}(x, \text{samples})$ 
11:     $\mu_Q(x) \leftarrow \text{MEAN}(\text{distances})$ 
12:     $\sigma_Q(x) \leftarrow \text{STANDARDDEVIATION}(\text{distances})$ 
13:     $\text{candidates} \leftarrow \text{RANGEANN}(s, x, \text{annIndex})$ 
14:    for all  $y \in \text{candidates}$  do  $\triangleright \mathcal{O}(sm)$ 
15:       $\text{dist} \leftarrow \text{PRIMARY DISTANCE}(x, y)$ 
16:       $\text{distMPG}(x, y) \leftarrow \text{MPG}(\text{dist})$ 
17: return  $\text{distMPG}$ 

```

DisSimLocal estimates local centroids for each object. A local centroid is the mean vector of an object's k nearest neighbors (see (5) and the paragraph below). Approximate DSL (Algorithm 4) estimates the centroids from approximate nearest neighbors.

All described methods are implemented in the publicly available OFAI Hub-Toolbox⁴ [26].

⁴Source code available at <https://github.com/OFAI/hub-toolbox-python3>. Python package available at <https://pypi.org/project/hub-toolbox/>. The Hub-Toolbox is free open source software licensed under GPLv3.

Algorithm 4 Approximate DisSimLocal

Input:
 k $\triangleright$ nearest neighbor parameter
 s $\triangleright$ Candidate neighborhood size
 I, Q $\triangleright$ Indexed objects, query objects, resp.
 c_I $\triangleright$ Empty matrix of size $n_{\text{indexed}} \times m$.
 c_Q $\triangleright$ Empty matrix of size $n_{\text{query}} \times m$.

Output:
 distDSL $\triangleright$ Sparse matrix of shape $n_{\text{query}} \times n_{\text{indexed}}$, $\mathcal{O}(n_{\text{query}}s)$

```

1: procedure FIT
2:    $\text{annIndex} \leftarrow \text{FITANNINDEX}(I)$ 
3:   for all  $y \in I$  do  $\triangleright \mathcal{O}(n_{\text{indexed}}m)$ 
4:      $\text{neighbors} \leftarrow \text{RANGEANN}(k, y, \text{annIndex})$ 
5:      $c_I(y) \leftarrow \text{LOCALCENTROID}(y, \text{neighbors})$ 
6: procedure TRANSFORM
7:   for all  $x \in Q$  do  $\triangleright \mathcal{O}(n_{\text{query}}sm)$ 
8:      $\text{neighbors} \leftarrow \text{RANGEANN}(k, x, \text{annIndex})$ 
9:      $c_Q(x) \leftarrow \text{LOCALCENTROID}(x, \text{neighbors})$ 
10:     $\text{candidates} \leftarrow \text{RANGEANN}(s, x, \text{annIndex})$ 
11:    for all  $y \in \text{candidates}$  do  $\triangleright \mathcal{O}(sm)$ 
12:       $\text{dist} \leftarrow \text{PRIMARYDISTANCE}(x, y)$ 
13:       $\text{distDSL}(x, y) \leftarrow \text{DSL}(x, y, c_I, c_Q)$ 
return  $\text{distDSL}$ 
    
```

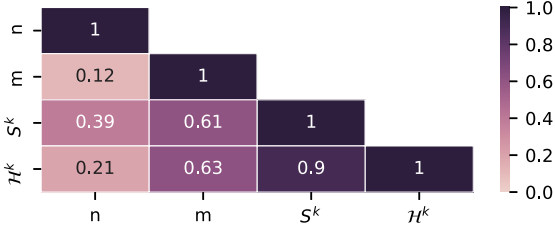


Fig. 2. Evaluation of Robin Hood (H^k) measure. Values indicate Spearman correlation between attributes (n data set size, m dimension, S^k skewness).

IV. EVALUATION

We evaluate the proposed hubness measure and approximate hubness reduction methods with multiple experiments on both synthetic and real-world data.

A. Robin Hood index

The findings on synthetic data presented in Fig. 1 suggest the Robin Hood index to be a suitable measure for hubness. We further evaluate the Robin Hood index on 50 real-world data sets, which have previously been used for analyzing aspects of hubness measures in [12]. Skewness $S^{k=10}$ and Robin Hood index $H^{k=10}$ are calculated for each data set without any sampling. The proposed hubness measure shows very strong rank correlation (Spearman $r = 0.90$) with the well-established measure (Fig. 2). We observe equal correlations of both measures with data dimensionality, but weaker dependency of the Robin Hood index on data set size, compared to the skewness measure. Both results are in line with the results from Fig. 1. We thus deem the Robin Hood index a suitable measure for hubness. Since it allows to estimate hubness from small sample sizes, hubness can now be estimated accurately in linear time.

B. Approximate hubness reduction - preface

The methodology presented in Section III-C is agnostic to the choice of ANN method. We select two methods from different ANN families as showcases for approximate hubness reduction:

- *Locality-sensitive hashing* (LSH) is an established and commonly used ANN method [22]. LSH is based on hash functions, which are constructed so that close objects have high probability of collisions. Objects with colliding hashes are mapped to the same approximate neighborhood.
- *Navigable small-world graphs* (NSW) are a more recent, graph-based approach to ANN. Indexed objects form the vertices of a graph, whose edges are built so that the network contains an approximate Delaunay subgraph [24]. Insertions and queries are performed by greedy search. The HNSW variant features a controllable hierarchy [27], and was particularly successful in a recent ANN benchmark [28].

We use the FALCONN [29] and nmslib [30] libraries for LSH- and HNSW-based experiments, respectively.

Previous work showed hubness reduction methods to be robust with respect to the neighborhood parameter k . We use fixed LS/DSL parameter $k = 5$ for all experiments in this work. MP is parameter-free.

We select a restrictive sample size $s = 100$ for all experiments. This is relaxed to $s = 1000$ only for the two largest data sets (UCI character fonts, Table IIg; and mobility data, Table III).

C. Results with synthetic data

We evaluate approximate hubness reduction in synthetic data sets of sizes $n \in \{1000, 2000, \dots, 20\,000\}$, constructed similarly to MADELON [31]. For each data set, n objects in ten clusters are generated at the vertices of an 800-dimensional hypercube, and 200 uninformative noise features are added, yielding a 1000-dimensional embedding space. Five class labels are assigned to the data, with each class comprising two clusters. We perform five-fold stratified cross-validation, and consider three evaluation metrics: (i) runtime (Fig. 3), (ii) hubness reduction, measured with the Robin Hood index (Fig. 4a), and (iii) accuracy in ten-nearest neighbor classification (Fig. 4b), where random classification yields 20% accuracy.

Full hubness reduction scales quadratically with the number of objects. This is visualized in the top row of Fig. 3 for model creation (fit procedure) in the left column, and (secondary) distance calculations (transform procedure) for query objects in the right column. The red line in Fig. 3ab represents baseline k -NN with Euclidean distances. Each of the following rows illustrates the time complexity of one approximation method. For example, Fig. 3c shows time requirements for instance selection with random sampling, which is, by far, the fastest method, because it does not consider properties of the indexed data. In contrast, k -means++ processes the indexed data to

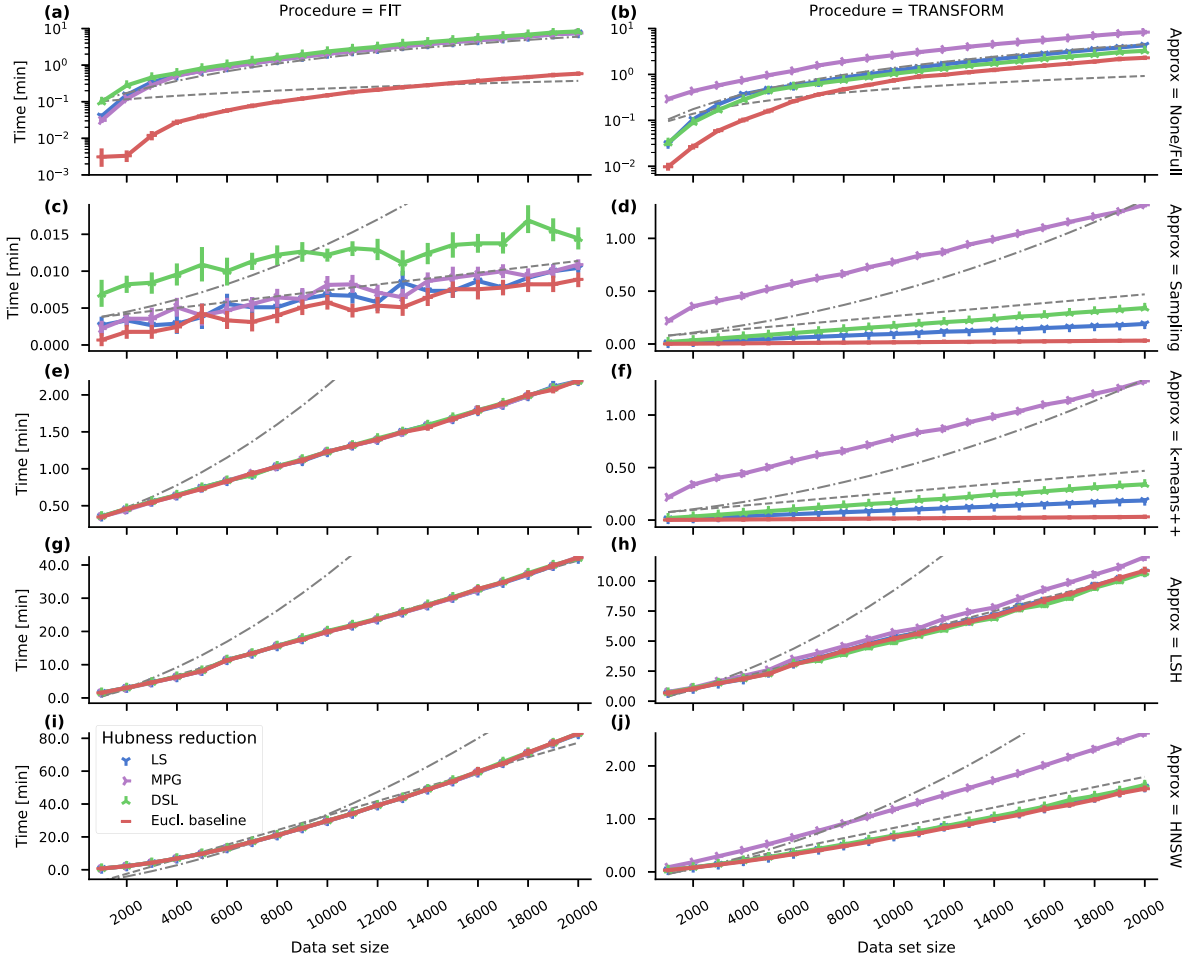


Fig. 3. Linear time complexity of approximate hubness reduction. Runtime for 80 % / 20 % train / test-split with synthetic data. Colors code for hubness reduction methods (blue: local scaling, violet: mutual proximity, green: DisSimLocal, red: Euclidean distances without hubness reduction). Each row corresponds to one approximation method, as indicated on the right-hand side. Left column: model creation time, right column: (secondary) distance compute time. For example, the blue line in subplot h shows time spent in procedure TRANSFORM of Algorithm 2. It is mostly overlayed by the red line, indicating that primary distance calculation dominates overall runtime. Values are averages of ten replicates. Error bars indicate standard deviation (generally low and clearly visible only in subplot c). Gray lines show linear (dashed) and quadratic (dash-dotted) functions for comparison. Note the log-scaled y-axes in a and b.

select suitable instances. It thus spends more time in the fit procedure, while the transform time is identical for both instance selection methods.

Runtime of approximate hubness reduction with ANN methods is dominated by index structure generation, as indicated by the Euclidean baseline in red, which overlays the other curves. Index generation is linear in case of LSH (Fig. 3g), and linearithmic in case of HNSW (Fig. 3i; consider the slightly increased growth compared to the fitted linear function displayed as dashed gray line).

Query time is only slightly longer with hubness reduction than without, especially in case of LS and DSL (Fig. 3hj).

In other words, hubness reduction comes essentially ‘for free’ compared to solely performing ANN search.

Fig. 4 shows hubness and classification performance in synthetic data with 20 000 objects (corresponding to the right-most values in Fig. 3). Fig. 4a shows that ANN-based methods reduce hubness in terms of the Robin Hood index to comparable levels as full hubness reduction. Instance selection, however, nearly maximizes the Robin Hood index. This is caused by measuring hubness with respect to the original database. Selecting hundred instances from twenty thousands objects essentially creates 19 900 antihubs and thus leads to extremely imbalanced k -occurrences.

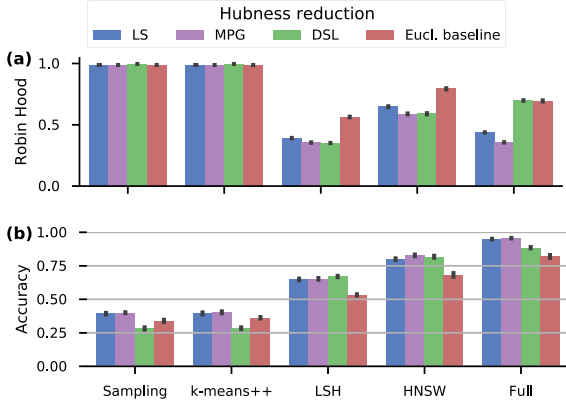


Fig. 4. Evaluation results of medium-sized synthetic data.

Fig. 4b shows that hubness reduction methods improve classification in all cases of approximation or no approximation compared to the Euclidean baselines depicted as red bars (except for DSL with instance selection). HNSW with hubness reduction is on par with full nearest neighbor classification using Euclidean distances, despite the very small sample size of 100, i.e. only 0.625 % of the training set size.

The feasibility of hubness reduction in large data sets is demonstrated on a synthetic data set of one million objects, for which full hubness reduction would require nearly 4 TiB main memory. Table I shows the throughput of approximate hubness reduction algorithms at runtime on a Intel Xeon Silver 4114 CPU @ 2.20 GHz in a single-threaded environment. Hubness and classification performance are in line with the results in Fig. 4 (not shown). Hubness reduction calculates secondary distances from Euclidean distances. Therefore, it requires additional compute time compared to the Euclidean baseline. The extra time expenditure is, however, moderate for LSH and HNSW, in which cases hubness reduction requires roughly twice the baseline time. Instance selection methods are orders of magnitude faster, but suffer from severely impaired accuracy.

D. Results with public machine learning data sets

Having established successful approximate hubness reduction in synthetic data, we evaluate our methods on seven real-world data sets affected by hubness from public machine learning repositories also [32]–[34].

For all data sets, the Robin Hood index is improved by full hubness reduction methods, as expected (Table II, left-hand side). ANN without hubness reduction yields the same level of hubness as the complete Euclidean distance matrices, with LSH reducing hubness slightly in some cases. Approximate hubness reduction yields highly comparable Robin Hood values compared to full hubness reduction in all cases, except for HNSW, which does not perform well for UCI farm-ads.

TABLE I
APPROXIMATE HUBNESS REDUCTION THROUGHPUT IN 1M DATA.

	Random sampling	k-means++	LSH	HNSW
LS	103 412	538	10	31
MPG	113 636	566	11	30
DSL	99 304	577	10	31
Eucl.	446 428	3240	25	59

(a) Throughput in procedure FIT (objects/s)

	Random sampling	k-means++	LSH	HNSW
LS	33 344	30 211	35	1857
MPG	8501	8730	39	1509
DSL	22 036	19 527	35	1419
Eucl.	255 754	264 550	86	2759

(b) Throughput in procedure TRANSFORM (objects/s)

For each data set, the best result in terms of classification accuracy is obtained with one of the hubness reduction methods (Table II, right-hand side). The results of approximate methods closely follow those of full methods, that is, both full and approximate hubness reduction improve accuracy over the Euclidean baseline. Indeed, LSH-based hubness reduction achieved slightly higher accuracy than full hubness reduction in five data sets. Note, that in case of UCI character fonts, full hubness reduction is not feasible with reasonable compute resources. We were, however, able to compute the full Euclidean baseline. Approximate hubness reduction improves over this baseline also in this large data set.

Both random sampling and k-means++ yield high Robin Hood values, comparably to the ones described in Section IV-C. Classification accuracy is on average 15 %-points below ANN-based approximate hubness reduction. Since instance selection is clearly inferior to ANN for hubness reduction here, we do not report the results in Table II for brevity.

V. APPLICATION: TRANSPORT MODE DETECTION

We test our methods on accelerometer time series collected by ninety participants equipped with smartphones and a special smartphone application that records sensor data during traveling in an urban environment. The users labeled the recorded trips with the chosen transport mode. Table III reports the amount of collected data per mode and the corresponding labels for classification.

For this work, we preprocess the data by splitting each recorded trip into segments of 30 seconds and aggregate the time series to a dimension of 600. Further, to make the recorded signal independent of the orientation of the device we use the 2-norm of the three-dimensional accelerometer vector as in [35]. Dynamic time warping (DTW) alignments are computed between time-series vectors with the IncDTW package for R [36]. DTW-based similarities are used to classify trip segments as either active mobility, transport on the road, or on the rail. The fourth class indicates the user having reached his or her destination (Table III). The complete DTW similarity matrix is used for experiments with full hubness

TABLE II
EVALUATION RESULTS OF REAL-WORLD DATA SETS.

	Hubness (Robin Hood)			Classification accuracy		
	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.282	0.282	0.282	0.932	0.932	0.933
MPG	0.282	0.283	0.294	0.933	0.935	0.930
DSL	0.303	0.323	0.354	0.938	0.906	0.927
Eucl	0.329	0.329	0.329	0.904	0.906	0.902

(a) OpenML Semeion (1593 samples, 256 features, 10 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.294	0.309	0.293	0.831	0.836	0.845
MPG	0.290	0.292	0.279	0.846	0.850	0.852
DSL	0.413	0.417	0.675	0.834	0.830	0.794
Eucl	0.388	0.433	0.414	0.773	0.779	0.780

(b) LibSVM DNA (2000 samples, 180 features, 3 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.308	0.307	0.297	0.746	0.744	0.745
MPG	0.323	0.331	0.357	0.715	0.715	0.693
DSL	0.315	0.556	0.332	0.735	0.500	0.740
Eucl	0.419	0.419	0.419	0.669	0.670	0.669

(c) Corel1000 (1000 samples, 192 features, 10 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.345	0.358	0.327	0.888	0.889	0.883
MPG	0.357	0.376	0.358	0.860	0.861	0.887
DSL	0.345	0.364	0.389	0.895	0.889	0.896
Eucl	0.485	0.511	0.508	0.875	0.856	0.864

(d) UCI CNAE-9 (1080 samples, 856 features, 9 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.329	0.353	0.338	0.783	0.776	0.775
MPG	0.320	0.334	0.320	0.777	0.768	0.769
DSL	0.338	0.369	0.515	0.742	0.761	0.782
Eucl	0.470	0.526	0.517	0.707	0.695	0.699

(e) LibSVM splice (1000 samples, 60 features, 2 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.430	0.681	0.383	0.801	0.683	0.832
MPG	0.505	0.670	0.462	0.814	0.747	0.848
DSL	0.447	0.671	0.595	0.852	0.769	0.838
Eucl	0.567	0.676	0.616	0.780	0.699	0.757

(f) UCI farm-ads (4143 samples, 54877 features, 2 classes)

	LSH	HNSW	Full	LSH	HNSW	Full
LS	0.303	0.308	n/a	0.675	0.675	n/a
MPG	0.304	0.335	n/a	0.678	0.665	n/a
DSL	0.322	0.335	n/a	0.687	0.680	n/a
Eucl	0.355	0.358	0.359	0.656	0.655	0.633

(g) UCI character-fonts (832670 samples, 410 features, 153 classes)

reduction. In case of LSH and HNSW, the candidate neighbors are first retrieved using the methods' intrinsic distance measures. These distances are subsequently replaced by DTW similarities, which are used for the baseline in Fig. 5 ('LSH', 'HNSW'). Nearest neighbor classification performance is estimated in leave-one-user-out cross-validation. Classification accuracy macro-averaged over the folds is reported as an estimate, how well the system performs for a new user. No attempt was made to optimize hyperparameters, since the objective here is to demonstrate improvements due to hubness reduction,

TABLE III
ACCELEROMETER DATA FOR TRANSPORT MODE DETECTION

Class	Mode	Time (h)	# 30 s trip segments
ACTIVE	bicycle	52	6273
	waiting	9	1108
	walking	245	29 443
ROAD	bus	49	5828
	car	108	12 937
	motorcycle	8	932
RAIL	metro	46	5481
	train	74	8848
	tram	33	3997
DESTINATION	destination	61	7263
		626	82 110

rather than maximizing performance of the complete classifier pipeline. Therefore, we use fixed $k = 10$ neighbors for classification to allow for direct comparisons between pipelines with and without hubness reduction. We postpone hyperparameter optimization to future work, which will focus on maximizing transport mode detection performance.

Hubness reduction is performed on DTW-based similarities with LS and MPG. We omit DSL due to its tailoring to Euclidean spaces. The database size of 82 110 allows us to compute a full DTW similarity matrix, and perform full hubness reduction for this evaluation scenario.⁵ We observe marked improvements in terms of Robin Hood index and classification accuracy due to hubness reduction with LS (Fig. 5, 'Full'). In contrast, full MPG is detrimental to both hubness and accuracy.

Hubness appears to be heavily increased by the approximate methods. For instance selection, this is expected, as described in Section IV-C. Rather surprisingly, ANN methods show similar behavior given this data set. Analysis of k -occurrences revealed the imbalance again to be caused by a very high number of antihubs.

The approximations show reduced baseline classification performance compared to the full scenario. Applying hubness reduction on top, however, strongly improves accuracy. For both HNSW and random sampling, applying LS more than doubles accuracy. Strikingly, random sampling with LS yields the best classification performance among the approximate methods, and is on par with the full scenario. HNSW and k-means++ show competitive results also. The approximations profit from MPG as well, showing accuracy slightly below LS.

VI. DISCUSSION

Hubness analysis of massive data requires both hubness estimation and hubness reduction with linear complexity in time and space w.r.t. the data set size. The new Robin Hood index enables accurate hubness estimation from small samples. Linear algorithmic complexity of hubness reduction methods is achieved by introducing approximations based on instance

⁵ 1750h DTW compute time and 80 GiB peak memory during hubness reduction clearly show the infeasibility of this approach in productive systems.

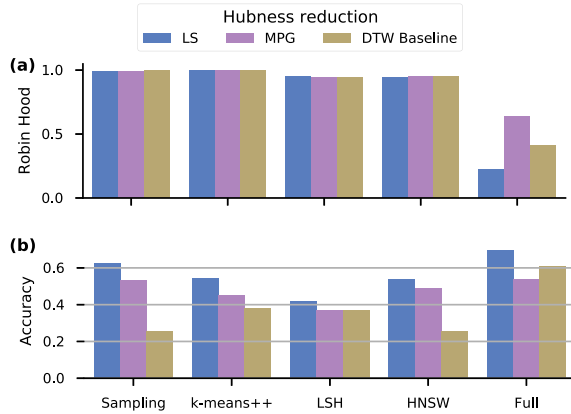


Fig. 5. Transport mode detection performance. Column ‘Full’ refers to using the complete DTW similarity matrix.

selection or ANN methods. We find that this comes at almost no expense in terms of classification accuracy in case of the ANN-based approach: Both LSH and HNSW showed overall comparably good performance. Instance selection performs worse than ANN in most cases. In contrast to our expectations, we observe no advantages of k-means++ instance selection over random sampling. The extra time required for k-means++ may be better used for larger random samples. Our experiments on synthetic data show only moderate additional time required for hubness reduction compared to approximate neighbor search alone. The ANN libraries used in this work were among the fastest in a recent benchmark [28]. We thus deem our methods to be fast enough for use in productive systems.

Experiments on data sets from machine learning repositories show, that successful approximate hubness reduction is not limited to synthetic data sets, which may exhibit special properties, such as Gaussian distributions etc. The approximate methods performed on par with the full methods in all evaluated data sets, hinting at their potential to completely replacing expensive full hubness reduction.

We demonstrate, how a real-world application is enhanced by approximate hubness reduction. Transport mode detection based on mobile phone sensor data is improved, despite supposedly unsuccessful reduction of hubness *per se*. Let us discuss the high Robin Hood indices of approximate hubness reduction in mobility data in detail: LSH and HNSW use metric distances for approximate neighbor search. These are generally not well suited for time-series data. Theoretical hubness research found, that hubs are usually close to the data centroid. For normed accelerometer data, the time-series centroid is a vector of approximately 9.81 m/s^2 at each point in time, i.e. zero movement plus earth gravity. Indeed we find, that mobility hub objects in Euclidean space are mostly trip segments of very little user movement. This includes waiting at red traffic lights or traffic jams, which may occur

in nearly any vehicle, or during walking. Segments with little movement may thus belong to essentially any of the presented transport modes. It is easy to imagine how this severely affects classification. Metric ANN seems to be influenced by this behavior as well, retrieving many low movement segments, but only few high movement segments as candidate neighbors. We hypothesize that hubness reduction is able to reorder the candidate neighborhood in a way that penalizes segments of very low movement. Unless the candidate neighborhood size is too restrictive, the high number of antihubs appears to be noncritical, and hubness reduction thus improves classification accuracy.

VII. CONCLUSION AND OUTLOOK

We presented approximate hubness reduction methods with linear complexity, which enable hubness reduction in large data sets. The approximations come at almost no cost in terms of accuracy compared to full hubness reduction as demonstrated for synthetic data, and even surpassed the full methods in several real-world data sets. We also showed how transport mode detection is improved by both full and approximate hubness reduction. In addition, the new Robin Hood index enables fast hubness estimation in massive data sets, and is more easily interpretable than a traditional measure. Having eliminated the bottleneck of high complexity, we hope hubness reduction will aid many researchers and practitioners with their specific tasks for large data sets. To this end, we make all our hubness reduction methods publicly available in a free open source software package.

In future work on general hubness, we will investigate additional approximate kNN-graph construction methods [37], [38] for hubness estimation, and additional ANN methods, such as tree-based methods or product quantization, for approximate hubness reduction. In terms of concrete applications, we will perform in-depth analysis of hubness in mobility data. We will test our hypotheses of how hubness reduction improves classification. In addition, we will examine, whether different ANN techniques can yield better candidate neighbors and avoid high numbers of antihubs. A focus will be on non-metric techniques, such as distance-based hashing [39] and others.

ACKNOWLEDGMENT

We thank Jan Schlüter for fruitful discussions, and Thomas Rattei for computational resources.

REFERENCES

- [1] R. E. Bellman, *Adaptive control processes: A guided tour*. Princeton university press, 1961.
- [2] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is ‘nearest neighbor’ meaningful?” In *Database Theory – ICDT’99*, C. Beeri and P. Buneman, Eds., Springer Berlin Heidelberg, 1999, pp. 217–235.
- [3] A. Zimek, E. Schubert, and H.-P. Kriegel, “A survey on unsupervised outlier detection in high dimensional numerical data,” *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 5, no. 5, pp. 363–387, 2012. DOI: 10.1002/sam.11161.
- [4] D. Francois, V. Wertz, and M. Verleysen, “The concentration of fractional distances,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 7, pp. 873–886, 2007.

- [5] A. Nanopoulos, M. Radovanovic, and M. Ivanovic, "How does high dimensionality affect collaborative filtering?" In *Proceedings of the 2009 ACM Conference on Recommender Systems*, New York, NY, USA, L. D. Bergman, A. Tuzhilin, R. D. Burke, A. Felfernig, and L. Schmidt-Thieme, Eds., ACM, 2009, pp. 293–296.
- [6] D. Yu, X. Yu, and A. Wu, "Making the nearest neighbor meaningful for time series classification," in *4th International Congress on Image and Signal Processing*, vol. 5, Oct. 2011, pp. 2481–2485.
- [7] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Hubs in space: Popular nearest neighbors in high-dimensional data," *Journal of Machine Learning Research*, vol. 11, pp. 2487–2531, 2010.
- [8] D. Schnitzer and A. Flexer, "The unbalancing effect of hubs on k-medoids clustering in high-dimensional spaces," in *International Joint Conference on Neural Networks (IJCNN)*, Jul. 2015, pp. 1–8.
- [9] A. Flexer, "Improving visualization of high-dimensional music similarity spaces," in *Proceedings of the 16th International Society for Music Information Retrieval (ISMIR) Conference*, 2015, pp. 547–553.
- [10] M. Radovanović, A. Nanopoulos, and M. Ivanović, "Reverse nearest neighbors in unsupervised distance-based outlier detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 5, pp. 1369–1382, 2015.
- [11] A. Flexer, "An empirical analysis of hubness in unsupervised distance-based outlier detection," in *16th International Conference on Data Mining Workshops (ICDMW)*, IEEE, 2016, pp. 716–723.
- [12] R. Feldbauer and A. Flexer, "A comprehensive empirical comparison of hubness reduction in high-dimensional spaces," *Knowledge and Information Systems*, May 2018. [Online]. Available: <https://doi.org/10.1007/s10115-018-1205-y>.
- [13] D. Schnitzer, A. Flexer, M. Schedl, and G. Widmer, "Local and global scaling reduce hubs in space," *Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2871–2902, 2012.
- [14] L. Zelnik-Manor and P. Perona, "Self-tuning spectral clustering," *Advances in Neural Information Processing Systems*, vol. 17, L. Saul, Y. Weiss, and L. Bottou, Eds., pp. 1601–1608, 2005.
- [15] K. Hara, I. Suzuki, K. Kobayashi, K. Fukumizu, and M. Radovanović, "Flattening the density gradient for eliminating spatial centrality to reduce hubness," in *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 2016, pp. 1659–1665.
- [16] T. Low, C. Borgelt, S. Stober, and A. Nürnberger, "The hubness phenomenon: Fact or artifact?" In *Towards Advanced Data Analysis by Combining Soft Computing and Statistics*, C. Borgelt, M. Á. Gil, J. M. Sousa, and M. Verleysen, Eds., Springer Berlin Heidelberg, 2013, pp. 267–278.
- [17] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intell. Data Anal.*, vol. 11, no. 5, pp. 561–580, Oct. 2007.
- [18] E. M. Hoover, "The measurement of industrial localization," *The Review of Economics and Statistics*, vol. 18, no. 4, pp. 162–171, 1936. [Online]. Available: <http://www.jstor.org/stable/1927875>.
- [19] D. Arthur and S. Vassilvitskii, "K-means++: The advantages of careful seeding," in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.
- [20] S. García, J. Luengo, and F. Herrera, "Tutorial on practical tips of the most influential data preprocessing algorithms in data mining," *Knowledge-Based Systems*, vol. 98, pp. 1–29, 2016.
- [21] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.
- [22] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, ACM, 1998, pp. 604–613.
- [23] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2011.
- [24] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, "Approximate nearest neighbor algorithm based on navigable small world graphs," *Information Systems*, vol. 45, pp. 61–68, 2014.
- [25] N. Tomašev, M. Radovanović, D. Mladenović, and M. Ivanović, "The role of hubness in clustering high-dimensional data," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 739–751, 2014.
- [26] R. Feldbauer and A. Flexer, "Centering versus scaling for hubness reduction," in *Artificial Neural Networks and Machine Learning – ICANN 2016*, A. E. Villa, P. Masulli, and A. J. Pons Rivero, Eds., ser. Lecture Notes in Computer Science, vol. 9886, Cham: Springer International Publishing, 2016, pp. 175–183.
- [27] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using Hierarchical Navigable Small World graphs," *ArXiv e-prints*, Mar. 2016. arXiv: 1603.09320 [cs.DS].
- [28] M. Aumüller, E. Bernhardsson, and A. Faithfull, "ANN-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms," in *International Conference on Similarity Search and Applications*, Springer, 2017, pp. 34–49.
- [29] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, "Practical and optimal LSH for angular distance," in *Advances in Neural Information Processing Systems* 28, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 1225–1233.
- [30] L. Boytsov and B. Naidan, "Engineering efficient and effective non-metric space library," in *Proceedings of the 6th International Conference on Similarity Search and Applications SISAP, A Coruña, Spain*, 2013, pp. 280–293.
- [31] I. Guyon, S. R. Gunn, A. Ben-Hur, and G. Dror, "Result analysis of the NIPS 2003 feature selection challenge," in *Advances in Neural Information Processing Systems 17 (Vancouver, Canada)*, 2004, pp. 545–552.
- [32] M. Lichman, *UCI machine learning repository*, 2013. [Online]. Available: <http://archive.ics.uci.edu/ml>.
- [33] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, 27:1–27:27, May 2011.
- [34] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml: Networked science in machine learning," *SIGKDD Explorations*, vol. 15, no. 2, pp. 49–60, 2013.
- [35] M. A. Shafique and E. Hato, "Travel mode detection with varying smartphone data collection frequencies," *Sensors*, vol. 16, no. 5, p. 716, 2016.
- [36] M. Leodolter, *IncDTW: Incremental calculation of dynamic time warping*, R package version 1.0.3, 2018. [Online]. Available: <https://CRAN.R-project.org/package=IncDTW>.
- [37] J. Chen, H.-r. Fang, and Y. Saad, "Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection," *Journal of Machine Learning Research*, vol. 10, pp. 1989–2012, 2009.
- [38] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proceedings of the 20th international conference on World wide web*, ACM, 2011, pp. 577–586.
- [39] V. Athitsos, M. Potamias, P. Papapetrou, and G. Kollios, "Nearest neighbor retrieval using distance-based hashing," in *24th IEEE International Conference on Data Engineering*, Apr. 2008, pp. 327–336.

4. Implementation of hubness reduction

This doctoral thesis investigates the effect of hubness reduction on protein sequence similarity networks. Hubness reduction algorithms are evaluated in Chapter 2, and refined in Chapter 3. This work package comprises a reference implementation of hubness reduction methods in a free open source software package: `scikit-hubness`.

The program is written in PEP8-compliant Python3, and fully compatible to the `scikit-learn` machine learning framework. Development follows best practices, including extensive software tests and documentation. This package not only provides a reliable foundation for hubness-related experiments in this thesis, but also enables the scientific community and machine learning practitioners in industry to effortlessly employ hubness reduction and approximate neighbor search for their tasks on high-dimensional data.

Hubness investigations in the SIMAP2 database of protein sequence similarities (see Chapter 7) as well as explorations on the usage of approximate nearest neighbor algorithms for sub-quadratic protein sequence similarity search (see Chapter 5) build upon `scikit-hubness`.

Published work Feldbauer, R., Rattei, T., and Flexer, A. (2020). *scikit-hubness: Hubness Reduction and Approximate Neighbor Search*. Journal of Open Source Software, 5(45), 1957, <https://doi.org/10.21105/joss.01957>.

Contributions RF implemented the `scikit-hubness` package, wrote the software documentation, setup all online services, and wrote the manuscript. TR and AF supervised the work.



scikit-hubness: Hubness Reduction and Approximate Neighbor Search

Roman Feldbauer¹, Thomas Rattei¹, and Arthur Flexer²

¹ Division of Computational Systems Biology, Department of Microbiology and Ecosystem Science, University of Vienna, Althanstraße 14, 1090 Vienna, Austria ² Austrian Research Institute for Artificial Intelligence (OFAI), Freyung 6/6/7, 1010 Vienna, Austria

DOI: [10.21105/joss.01957](https://doi.org/10.21105/joss.01957)

Software

- [Review](#) ↗
- [Repository](#) ↗
- [Archive](#) ↗

Editor: [Yuan Tang](#) ↗

Reviewers:

- [@ryEllison](#)
- [@aozorahime](#)

Submitted: 10 December 2019

Published: 17 January 2020

License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC-BY](#)).

Summary

scikit-hubness is a Python package for efficient nearest neighbor search in high-dimensional spaces. Hubness is an aspect of the *curse of dimensionality* in nearest neighbor graphs. Specifically, it describes the increasing occurrence of *hubs* and *antihubs* with growing data dimensionality: Hubs are objects, that appear unexpectedly often among the nearest neighbors of others objects, while antihubs are never retrieved as neighbors. As a consequence, hubs may propagate their information (for example, class labels) too widely within the neighbor graph, while information from antihubs is depleted. These semantically distorted graphs can reduce learning performance in various tasks, such as classification (Radovanović, Nanopoulos, & Ivanović, 2010), clustering (Schnitzer & Flexer, 2015), or visualization (Flexer, 2015). Hubness is known to affect a variety of applied learning systems (Angiulli, 2018), causing—for instance—overrepresentation of certain songs in music recommendations (Flexer & Stevens, 2018), or improper transport mode detection (Feldbauer, Leodolter, Plant, & Flexer, 2018).

Multiple hubness reduction algorithms have been developed to mitigate these effects (Flexer & Schnitzer, 2013; Hara, Suzuki, Kobayashi, Fukumizu, & Radovanovic, 2016; Schnitzer, Flexer, Schedl, & Widmer, 2012). We compared these algorithms exhaustively in a recent survey (Feldbauer & Flexer, 2019), and developed approximate hubness reduction methods with linear time and memory complexity (Feldbauer et al., 2018).

Currently, there is a lack of fully-featured, up-to-date, user-friendly software dealing with hubness. Available packages miss critical features and have not been updated in years (“Hub-Miner,” 2015), or are not particularly user-friendly (“Hub-Toolbox,” 2019). In this paper we describe scikit-hubness, which provides powerful, readily available, and easy-to-use hubness-related methods:

- hubness analysis (“Is my data affected by hubness?”): Assess hubness with several measures, including *k*-occurrence skewness (Radovanović et al., 2010), and Robin-Hood index (Feldbauer et al., 2018).
- hubness reduction (“How can we improve neighbor retrieval in high dimensions?”): Mutual proximity, local scaling, and DisSimLocal are currently supported, as they performed best in our survey. Exact methods as well as their approximations are available.
- approximate neighbor search (“Does it work for large data sets?”): Several methods are currently available, including locality-sensitive hashing (Aumüller, Christiani, Pagh, & Vesterli, 2019) and hierarchical navigable small-world graphs (Malkov & Yashunin, 2018).



`scikit-hubness` builds upon the SciPy stack (Virtanen, Gommers, Oliphant, Haberland, & others, 2019) and is integrated into the `scikit-learn` environment (Pedregosa, Varoquaux, Gramfort, Michel, & others, 2011), enabling rapid adoption by Python-based machine learning researchers and practitioners. Convenient interfaces to hubness-reduced neighbors-based learning are available in the `skhubness.neighbors` subpackage. It acts as a drop-in replacement for `sklearn.neighbors`, featuring all its functionality, and adding support for hubness reduction, where applicable. This includes, for example, the supervised `KNeighborsClassifier` and `RadiusNeighborsRegressor`, `NearestNeighbors` for unsupervised learning, and the general `kneighbors_graph`.

`scikit-hubness` is developed using several quality assessment tools and principles, such as PEP8 compliance, unit tests with high code coverage, continuous integration on all major platforms (Linux, MacOS, Windows), and additional checks by LGTM. The source code is available at <https://github.com/Varlr/scikit-hubness> under the BSD 3-clause license. The online documentation is available at <https://scikit-hubness.readthedocs.io/>. Install from the Python package index with `$ pip install scikit-hubness`.

Outlook

Future plans include adaption to significant changes of `sklearn.neighbors` introduced in version 0.22 in December 2019: The `KNeighborsTransformer` and `RadiusNeighborsTransformer` transform data into sparse neighbor graphs, which can subsequently be used as input to other estimators. Hubness reduction and approximate search can then be implemented as Transformers. This provides the means to turn `skhubness.neighbors` from a drop-in replacement of `sklearn.neighbors` into a `scikit-learn` plugin, which will (1) accelerate development, (2) simplify addition of new hubness reduction and approximate search methods, and (3) facilitate more flexible usage.

Acknowledgements

We thank Silvan David Peter for testing the software. This research is supported by the Austrian Science Fund (FWF): P27703 and P31988

References

- Angiulli, F. (2018). On the behavior of intrinsically high-dimensional spaces: Distances, direct and reverse nearest neighbors, and hubness. *J. Mach. Learn. Res.*, 18, 170:1–170:60.
- Aumüller, M., Christiani, T., Pagh, R., & Vesterli, M. (2019). PUFFINN: parameterless and universally fast finding of nearest neighbors. In *27th annual european symposium on algorithms, ESA* (Vol. 144, pp. 10:1–10:16). doi:[10.4230/LIPIcs.ESA.2019.10](https://doi.org/10.4230/LIPIcs.ESA.2019.10)
- Feldbauer, R., & Flexer, A. (2019). A comprehensive empirical comparison of hubness reduction in high-dimensional spaces. *Knowledge and Information Systems*, 59(1), 137. doi:[10.1007/s10115-018-1205-y](https://doi.org/10.1007/s10115-018-1205-y)
- Feldbauer, R., Leodolter, M., Plant, C., & Flexer, A. (2018). Fast approximate hubness reduction for large high-dimensional data. In *Proc. IEEE Int. Conf. Big Knowledge* (pp. 358–367). doi:[10.1109/ICBK.2018.00055](https://doi.org/10.1109/ICBK.2018.00055)
- Flexer, A. (2015). Improving visualization of high-dimensional music similarity spaces. In *Proceedings of the 16th international society for music information retrieval conference, ISMIR 2015* (pp. 547–553).



- Flexer, A., & Schnitzer, D. (2013). Can shared nearest neighbors reduce hubness in high-dimensional spaces? In *Proc. IEEE 13th Int. Conf. Data Mining workshops* (pp. 460–467). doi:[10.1109/ICDMW.2013.101](https://doi.org/10.1109/ICDMW.2013.101)
- Flexer, A., & Stevens, J. (2018). Mutual proximity graphs for improved reachability in music recommendation. *Journal of New Music Research*, 47(1), 17–28. doi:[10.1080/09298215.2017.1354891](https://doi.org/10.1080/09298215.2017.1354891)
- Hara, K., Suzuki, I., Kobayashi, K., Fukumizu, K., & Radovanovic, M. (2016). Flattening the density gradient for eliminating spatial centrality to reduce hubness. In *AAAI conference on artificial intelligence*.
- HubMiner. (2015). Retrieved from <http://ailab.ijs.si/tools/hub-miner/>
- Hub-Toolbox. (2019). Retrieved from <https://github.com/OFAI/hub-toolbox-python3/>
- Malkov, Y. A., & Yashunin, D. A. (2018). Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1. doi:[10.1109/TPAMI.2018.2889473](https://doi.org/10.1109/TPAMI.2018.2889473)
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., & others. (2011). Scikit-learn: Machine learning in Python. *J. Mach. Learn. Res.*, 12, 2825–2830.
- Radovanović, M., Nanopoulos, A., & Ivanović, M. (2010). Hubs in space: Popular nearest neighbors in high-dimensional data. *J. Mach. Learn. Res.*, 11, 2487–2531.
- Schnitzer, D., & Flexer, A. (2015). The unbalancing effect of hubs on k-medoids clustering in high-dimensional spaces. In *Proc. Int. Joint Conf. Neural Networks (IJCNN)* (pp. 1–8). doi:[10.1109/IJCNN.2015.7280303](https://doi.org/10.1109/IJCNN.2015.7280303)
- Schnitzer, D., Flexer, A., Schedl, M., & Widmer, G. (2012). Local and global scaling reduce hubs in space. *J. Mach. Learn. Res.*, 13(1), 2871–2902.
- Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., & others. (2019). SciPy 1.0—fundamental algorithms for scientific computing in Python. *arXiv preprint*. Retrieved from <http://arxiv.org/abs/1907.10121v1>

5. Protein sequence embedding

The main goal of this work is investigating hubness reduction in large protein similarity networks. Approximate hubness reduction algorithms are presented in Chapter 3. Since they operate on numerical vector data only (as opposed to sequence strings or similarity data), protein sequences must be embedded in vector spaces in order to apply hubness reduction to large data sets. In this work package strategies for protein sequence vector embeddings are explored in the general context of fast similarity search (working title *frenetiq*), testing Hypothesis H₄.

Preliminary results were presented at the following conferences:

Feldbauer, R., Flexer, A., and Rattei, T. *Protein vector representations for fast similarity search*. German Conference on Bioinformatics (GCB) 2018, Vienna, Austria, (poster).

Feldbauer, R., Flexer, A., and Rattei, T. *Deep learning for extremely fast protein similarity search*. Austrian HPC Meeting 2019, Grundlsee, Austria, (talk).

Contributions RF designed the study, developed and implemented the methods, performed the experiments and analyzed the results. AF and TR supervised the work.

5.1. Introduction and background

Sequence analysis is one of the corner stones of computational biology. It covers a wide range of methods to process nucleic acid or amino acid sequences to gain insights into structural, functional, or evolutionary patterns. Sequence similarity and database search have numerous applications in comparative genomics, metagenomics, and many other fields. Database search requires some measure of sequence similarity. These are often derived from sequence alignments, for which a multitude of algorithms have been conceived. Dynamic programming algorithms guarantee to find optimal global [29] or local [30] alignments, but are prohibitively slow for large-scale comparisons. Various approaches accelerate the process, for example, by optimized usage of parallelism and specific features of CPUs, GPUs, and other hardware [36]–[40], or by search heuristics [42]–[46]. While computing time for sequence comparison can be reduced dramatically by these approaches, they do not reduce computational complexity of database search: Processor time of individual database queries scales linearly with respect to the number of indexed sequences, because each query must be compared against all indexed sequences. Consequently, all-against-all approaches such as the SIMAP database [52] which drives downstream projects like databases of protein interaction (STRING [173]) or orthology (eggNOG [174]), scale at least quadratically with the number of sequences.

Sequence analysis can become a bottleneck when high-throughput sequencing in genomic or metagenomic projects yields massive amounts of data. Biological

databases are growing perpetually. For a prominent example, the UniProtKB protein database [175], has grown from 60 million to 120 million sequences between 2016 and 2018. The SIMAP2 database contains sequence similarities between all pairs of approximately 30 million proteins. Computation of exact alignments required approximately ten million CPU core hours on the Vienna Scientific Cluster 3, and results necessitate 60 TiB storage capacity.

Given this rapid accumulation of sequences in public databases, similarity search puts increasingly heavy burden on computational infrastructure. Methodological improvements are required to handle this massive computational challenges. Ideally, methods should provide (i) a fast scoring function for sequence similarity, and (ii) provide heuristics that allow for database queries in sublinear time.

Alignment-free methods often provide computationally more efficient alternatives to alignment-based approaches. Multiple algorithms have been developed building upon statistics of subword (k-mer, n-gram) frequencies [47]–[49]. Approximate neighbor search with locality-sensitive hashing has been applied to biological sequences since 2001 [176]–[178], but the approach has not gained much traction in the community.

In recent years, deep learning has found widespread use in life science [163]–[166], [179]. In this work package (working title *frenetiq*), self-supervised deep learning-based, alignment-free methods for sequence comparison and database search are investigated. This is motivated by the following ideas: (i) Deep learning may yield protein sequence vector representations that meet certain criteria by careful selection of architecture and loss functions. The embedding space can be created so that (dis-)similarities of a protein network are reflected by standard distance functions with comparatively little computational cost. For example, (squared) Euclidean distances or Hamming distances might be employed with real valued or binary vectors, respectively. (ii) Protein sequence representations in Euclidean or Hamming spaces immediately allow for the application of most approximate neighbor search algorithms, including locality-sensitive hashing [74], [180], product quantization [181], or graph-based approaches [75], [182], [183]. (iii) The resulting protein sequence representations can be used in approximate hubness reduction, relevant to the investigation of hubness in protein similarity networks (see Chapter 7).

A similar approach for fixed-length nucleic acid sequence representation learning (SENSE, SiamEse Neural network for Sequence Embedding) was published during the execution of this work package [184].

5.2. Materials and methods

Protein sequence vector representations are trained here with deep learning from precomputed alignment similarity matrices. The following sections describe the process of choosing suitable network architectures and learning procedures, detail the used networks, and outline the employed data sets.

5.2.1. Protein alignment similarity scores

This work makes heavy use of ground-truth similarity scores of pairs of protein sequences obtained from exact Smith-Waterman (SW) alignments precomputed for the SIMAP2 database [53].

Raw SW alignment scores depend on query and index sequence lengths. Undesired consequences and other properties include

- variable scores for perfect matches:
 $\exists x_1, x_2 \in X : SW(x_1, x_1) \neq SW(x_2, x_2),$
- non-symmetric scores (in certain configurations of sequence content dependent substitution matrices):
 $\exists x_1, x_2 \in X : SW(x_1, x_2) \neq SW(x_2, x_1),$
- semibounded scores without upper limit (in case of SIMAP2, which sets all similarities below a certain threshold to zero):
 $SW : X \times X \mapsto \mathbb{R}^+,$

where SW gives the raw Smith-Waterman alignment score, and X is the protein sequence universe. These properties are not only detrimental to any “intuitive” interpretation of scores across alignments, but also violate the requirements of metric spaces.

We define the Smith-Waterman Ratio as follows:

$$SWR(x_1, x_2) = \frac{SW(x_1, x_2)}{\max[SW(x_1, x_1), SW(x_2, x_2)]} \quad (5.1)$$

Normalization by the larger self-score yields scores in the $[0, 1]$ interval, and self scores are uniformly 1. If not already guaranteed by SW, trivial expansion enforces SWR symmetry:

$$SWR_{\text{symm}}(x_1, x_2) = \frac{1}{2} \frac{SW(x_1, x_2) + SW(x_2, x_1)}{\max[SW(x_1, x_1), SW(x_2, x_2)]} \quad (5.2)$$

For SIMAP2, $SWR = SWR_{\text{symm}}$, because its SW function is carefully constructed for guaranteeing raw alignment score symmetry. An SWR dissimilarity measure emerges naturally as

$$SWR^{\text{dist}}(x_1, x_2) = 1. - SWR(x_1, x_2), \quad (5.3)$$

which features several necessary properties of distance metrics: (i) non-negativity (in SIMAP2, as explained above), (ii) identity of indiscernibles (even perfect matches of single domains within a multidomain protein do not yield zero distance due to the normalization), and (iii) symmetry (by SIMAP2 design). Thus, SWR^{dist} constitutes a semimetric, but not a metric, because the triangle inequality does not hold in general. Empirical experiments indicate that it is albeit *nearly* metric: Among 100 000 randomly sampled triples with nonzero similarities, $< 1\%$ of all triplets were found to not fulfill the triangle inequality. As a consequence, we hypothesize that many algorithms built for metric spaces perform sufficiently well in SWR^{dist} space.

5.2.2. Proof of concept

During the conception of this work, baseline methods for sequence embedding were examined in order to establish a proof of concept. For distance metrics, such as the Euclidean norm, vectors can be reconstructed analytically from distances matrices (subject to translational and rotational invariance). For non-metric dissimilarity measures, no general method for exact vector reconstruction exists. However, various methods create object embeddings from arbitrary dissimilarity measures.

The SMACOF algorithm [185] is used for iterative stress majorization in multi-dimensional scaling (MDS, [186]). It minimizes the stress σ w.r.t. the loss function

$$\sigma(X) = \sum_{i < j \leq n} w_{ij} [d_{ij}(X) - \delta_{ij}]^2, \quad (5.4)$$

where i, j index two objects from a dataset X of size n ; and d_{ij} computes the (Euclidean) distance between objects X_i and X_j . Furthermore, δ_{ij} describes their distance in another space, for example, a higher dimensional space for dimensionality reduction, or a dissimilarity space, as investigated here. Weights $w_{ij} \geq 0$ modify the influence of object pairs to the overall stress function. SMACOF iteratively yields solutions X that monotonically decrease stress. Monotonicity might not hold for SWR^{dist} , which is not fully metric (see Section 5.2.1).

The FastMap algorithm [187] was designed specifically for embedding objects in some m -dimensional space, when only distances between objects are known. Its main proposition is to treat objects as if they were points in an unknown m -dimensional space, an idea we will follow throughout this chapter. Vectors are created by projecting the points onto mutually orthogonal lines between pivot points.

Each coordinate of the embedding vector is computed by applying the cosine law:

$$x_i = \frac{d_{a,i}^2 + d_{a,b}^2 - d_{b,i}^2}{2d_{a,b}}, \quad (5.5)$$

where a, b index two pivot objects, i indexes the current object to be embedded, and d indicates the distance between two objects with the given indices. Repeating the process m times with m projections (with $2m$ pivot points) yields an m -dimensional vector representation of object i . Clearly, the formulation only requires to provide distances between objects. Incremental update of a FastMap database are cheap: Two terms involve only pivot points, and can be precomputed once at database creation. The other two terms require one distance computation (sequence alignment) each, totaling to $2m$ alignments per new sequence. For non-Euclidean spaces, the geometric interpretation does not apply (“pseudoprojections”), and orthogonality may not be established.

Distance-based hashing (DBH, [188]) combines FastMap pseudoprojections with hashing strategies. The approach yields an approximate neighbor search algorithm similar to locality-sensitive hashing for distance databases. The *frenetiq* project was partially inspired by this study, and aims at improving the DBH pipeline for protein sequences with more sophisticated embedding functions using deep networks, and state-of-the-art approximate search methods.

5.2.3. Considerations on deep network architectures

A wide range of deep learning approaches is available for representation learning. Several examples are outlined in the paragraphs below.

AutoEncoders [162] consist of an encoder and decoder, which overlap in a hidden layer in the middle. Input and output layers have identical shapes. The training objective is to minimize reconstruction error. Successful training yields (a possibly compressed) *latent representation* through the intermittent hidden layer, or ‘bottle-neck’ layer.

Generative adversarial networks (GANs) [189] consist of two subnetworks. The *generative* network is trained to generate new samples complying with some data distribution (“forgery”), while the *discriminative* network is trained to recognize counterfeit. The output of the generative network provides a latent representation of the input.

Doublet networks (also known as Siamese networks [190]) consist of two feed-forward networks sharing parameters. They operate on object pairs: Each subnetwork processes one object, and maps them to a latent space. Both representations are compared with some measure of dissimilarity (e.g., Euclidean distance). Doublet networks are often trained to minimize distance for similar objects, while maintaining a *margin* between dissimilar objects using a contrastive loss function [191].

Triplet networks use a similar strategy as doublet networks. Three objects are considered in each step, an anchor object plus a positive (similar) and a negative (dissimilar) object, subject to a triplet loss function [192], [193].

“Language models” form the basis of state-of-the-art natural language processing in 2020. These models map sentences to a latent space, typically by incorporating bidirectional LSTM [154] or Transformer [194] networks. Self-supervised training objectives include masked word or next sentence prediction.

Due to limited resources, not all available architectures could be investigated. Work focused on the doublet networks, as their formulation suggests a straightforward approach to create protein sequence representations in vector spaces with certain properties. Careful selection of loss functions yields representations that feature, for example, Euclidean or Hamming distances that reflect dissimilarity scores obtained from sequence alignments. The use of standard distance measures is crucial here, as approximate neighbor algorithms are typically not formulated for arbitrary dissimilarity measures. While computing distance functions on vectors is already significantly faster than computing sequence alignments, similarity search effort still grows proportionally with database size. Only by application of approximate search can query complexity be reduced to sublinear. Thus, protein sequence vector representations are trained with consideration of approximate search properties.

In principle, any of the above mentioned approaches can be combined with doublet networks. Notably, pretrained representations could be utilized inside a doublet network by adding, for example, an additional fully connected layer to the output, which would subsequently be finetuned for Euclidean properties. Protein sequence representations from unsupervised pretraining on large datasets with LSTM [195] or Transformer [196] languages models appeared, however, only after experiments in this work package were concluded.

5.2.4. Pipeline

The general pipeline of *frenetiq* is outlined in Figure 5.1. During training, two sequences and their corresponding SWR score are retrieved from SIMAP2. Each sequence is first one-hot encoded, and subsequently embedded via the deep network. Distance in vector space between both embeddings is compared to the precomputed alignment distance. Network weights are updated by backpropagating the loss, e.g., minimizing the mean absolute error between embedding and alignment distances. During inference, a new sequence is one-hot encoded and passed to one of the twin networks, which outputs the embedding.

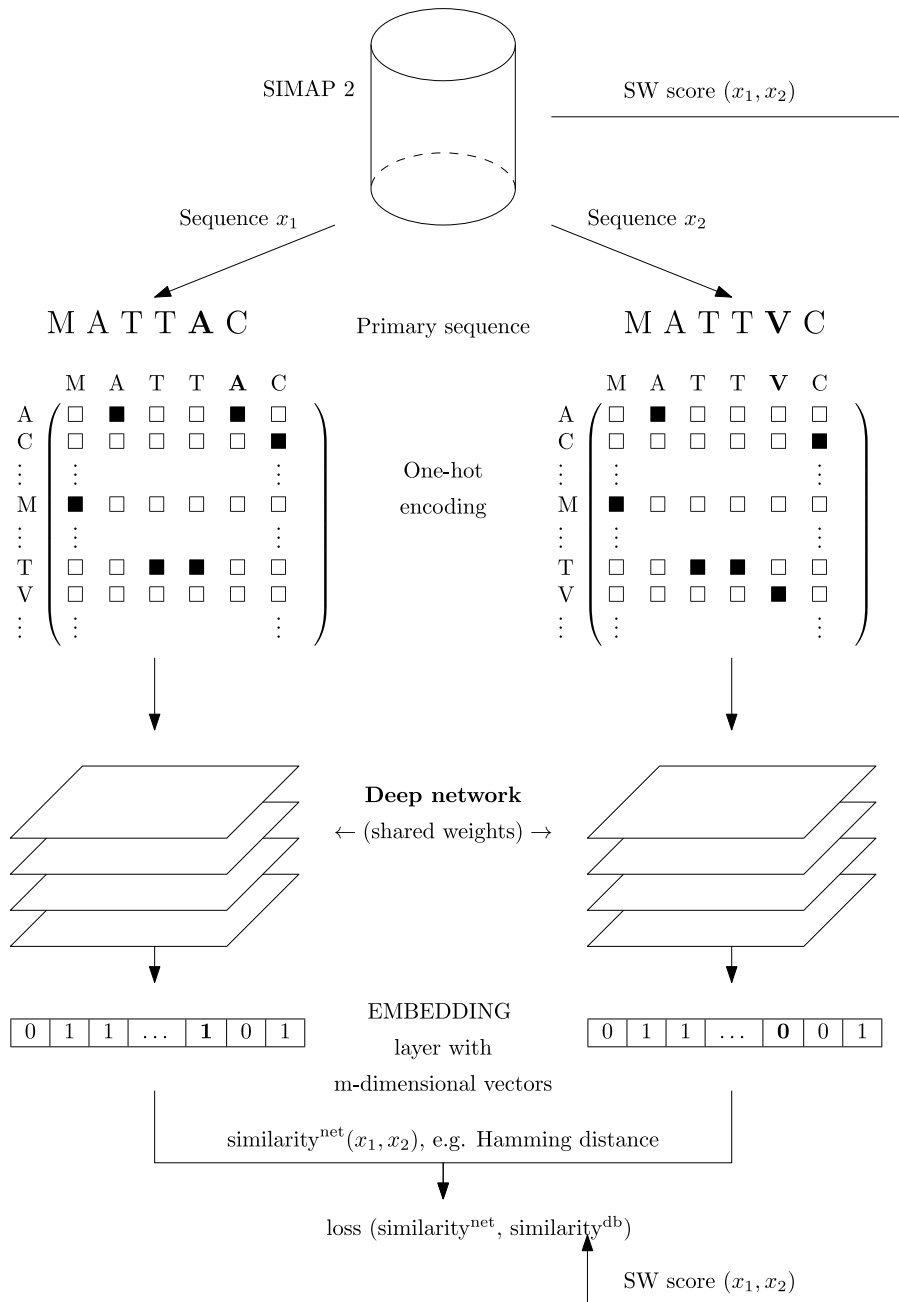


Figure 5.1.: Overview of the *frenetiq* pipeline. SW...Smith-Waterman

5.2.5. Representation learning with convolutional networks

Shortly after the general conception of this work, the SENSE network for biological sequence embedding was published [184]. The approach is not directly applicable to the task here primarily due to the following differences:

- Nucleic acid sequences versus amino acid sequences
- Fixed length (amplicon) sequences versus variable length
- Global alignment scores versus local alignment scores

SENSE, however, provides useful guidelines on design decisions, hyperparameter choices, and training procedures. This project’s goal is to create a function $\Psi: \Omega^L \mapsto \mathbb{R}^m$, that maps sequences from an amino acid alphabet Ω to an m -dimensional space so that alignment distance are maximally preserved in the embedding space. Ψ will here take the form of a deep network. For Ω the extended IUPAC protein single letter alphabet from Biopython [197] is employed. For practical purposes, sequence length is fixed to $L = 500$. Shorter sequences are zero-padded, while longer sequences are cut after position 500, affecting approximately 19% of sequences in SIMAP2. Technical improvements that allow processing full sequences of arbitrary length are discussed in the context of the follow-up study of orthologous group prediction in Chapter 6.

We create a convolutional network [198] for protein sequences by adapting the SENSE architecture, increasing the numbers of convolutional layers and filters, exchanging the activation function, and streamlining the dense subnetwork.

The network consists of five convolutional layers with variable numbers of filters, followed by continuously differentiable exponential linear unit (CELU, [199]) activation functions, batch normalization, and max-pooling. A fully connected layer maps the activations of the last convolutional layer to 128 output cells. The exact layout is detailed in Code Listing 5.1 (p. 70).

The network is trained subject to minimize mean absolute error of ground truth and embedding distances between two training sequences x_1, x_2 . Ground truth distances $\text{SWR}^{\text{dist}}(x_1, x_2)$ are obtained from SIMAP2. Euclidean distances are employed for the sequence embeddings. Optimization follows the Adam variant of stochastic gradient descent [200] with a step scheduler reducing the base learning rate by a factor ten after each epoch, starting from $\eta_0 = 0.01$. Mini-batches consisting of one query sequence and its hundred best hits from SIMAP2 are used for stochastic optimization. A maximum of three epochs is trained, due to computational limitations and time constraints. All experiments are performed on a single Nvidia P6000 GPU (“Pascal” architecture, 2016) with 24 G memory.

5.2.6. Approximate neighbor search

For efficient indexing of embedded sequences, approximate neighbor search structures are employed. In particular, we consider Hierarchical Navigable Small-World (HNSW) graphs [75], which showed most promising performance in the ANN-benchmark suite [201] and its recent updates¹. Indexes are built with default parameters used in the Python bindings of the non-metric space library `nmslib` [202].

¹ANN-benchmarks: <https://github.com/erikbern/ann-benchmarks>, accessed 2020/05/28

```
ConvNet5_CELU_BN_MaxPool_v2(  
  (cnn): Sequential(  
    (0): Conv1d(26, 64, kernel_size=(4,), stride=(1,), padding=(2,))  
    (1): CELU(alpha=1.0)  
    (2): BatchNorm1d(64, eps=1e-05, momentum=0.1, affine=True,  
      track_running_stats=True)  
    (3): MaxPool1d(kernel_size=2, stride=2, padding=0,  
      dilation=1, ceil_mode=False)  
    (4): Conv1d(64, 128, kernel_size=(4,), stride=(1,), padding=(2,))  
    (5): CELU(alpha=1.0)  
    (6): BatchNorm1d(128, eps=1e-05, momentum=0.1, affine=True,  
      track_running_stats=True)  
    (7): MaxPool1d(kernel_size=2, stride=2, padding=0,  
      dilation=1, ceil_mode=False)  
    (8): Conv1d(128, 256, kernel_size=(4,), stride=(1,), padding=(2,))  
    (9): CELU(alpha=1.0)  
    (10): BatchNorm1d(256, eps=1e-05, momentum=0.1, affine=True,  
      track_running_stats=True)  
    (11): MaxPool1d(kernel_size=2, stride=2, padding=0,  
      dilation=1, ceil_mode=False)  
    (12): Conv1d(256, 512, kernel_size=(4,), stride=(1,), padding=(2,))  
    (13): CELU(alpha=1.0)  
    (14): BatchNorm1d(512, eps=1e-05, momentum=0.1, affine=True,  
      track_running_stats=True)  
    (15): MaxPool1d(kernel_size=2, stride=2, padding=0,  
      dilation=1, ceil_mode=False)  
    (16): Conv1d(512, 1024, kernel_size=(4,), stride=(1,), padding=(2,))  
    (17): CELU(alpha=1.0)  
    (18): BatchNorm1d(1024, eps=1e-05, momentum=0.1, affine=True,  
      track_running_stats=True)  
    (19): MaxPool1d(kernel_size=2, stride=2, padding=0,  
      dilation=1, ceil_mode=False))  
  (fc): Sequential(  
    (0): Linear(in_features=16384, out_features=128, bias=True)))
```

Code Listing 5.1: *Frenetiq* network architecture. List elements refer to sequential layers in the network with early layers on top. Non-keyword arguments refer to input or output dimensions. See also: Section 5.2.5

5.2.7. Data sets

The experiments use data extracted from the SIMAP2 database, which comprises 28 934 810 sequences from several thousand fully assembled genomes. Sequences are split into training, validation, and test sets by a ratio of 0.8/0.1/0.1. The data set comprises the all-against-all similarity matrix. Sequence pairs consist of two training sequences (64%), two validation sequences or test sequences (1% each), or mixtures of training-validation and training-test (16% each), or validation-test (2%). Training, validation and testing are solely performed on homogeneous pairs.

5.3. Results and discussion

In this section, results of similarity search in protein sequence embedding spaces are presented and discussed.

5.3.1. Proof of concept

The *frenetiq* project started with initial experiments on protein sequence vector embedding with established and easily applicable methods in order to demonstrate the potential of the approach.

Multi-dimensional scaling with SMACOF was found to be too time consuming in initial experiments. Moreover, it does not yield an explicit model for embedding new sequences. Incremental updates are not straight-forward, not implemented in standard machine learning packages², and would require at least $\mathcal{O}(n)$ time [187], where n is the number of indexed objects. MDS is, therefore, not considered for further experiments.

The proof-of-concept is performed with FastMap embeddings. Among the $\approx 30\text{M}$ sequences in SIMAP2, 100 pivot sequences are selected by a greedy heuristic for a dominating set [203], (selecting the sequence with most stored hits in SIMAP2, discarding its hits, and repeating the process 99 times). The total dominating set contains more than 500 000 sequences, among which the selected 100 sequences of highest node degree cover more than half of all vertices in the SIMAP2 graph. 128 pseudoprojections are selected from the pivot sequences. SIMAP2 sequences are embedded accordingly, and split into indexed and query objects. Top-1 accuracy (fraction of queries for which the SIMAP2 best hit is found as the FastMap nearest neighbor) is 13.54%, where a null model with random neighbor selection would achieve $3 * 10^{-6}\%$. This shows that space geometry created by simple embedding methods encode sequence similarity to some degree, and proves the concept. Clearly, the approach described here is suboptimal. For example, a large fraction of the protein space is not covered by the graph spanned by the pivot sequences. In order to cover the whole sequence universe, too many projections are necessary. Further experiments, therefore, focus solely on deep learning.

5.3.2. Deep learning

Having established feasibility of the general approach for similarity search in protein sequence vector embeddings, all further experiments focus on deep learning. Protein sequences representations are trained in a doublet network comprising two convolutional networks as described in Section 5.2.5. Mean absolute error between alignment and embedding distances serves as a proxy for similarity retrieval error, which is expensive to evaluate, and not easily differentiable.

After training the doublet network, all SIMAP2 sequences are embedded into the learned vector space. Figure 5.2 visualizes the correlation between 10 000 randomly sampled ground-truth and embedding distances. While linear Pearson $r = 0.96$ suggests strong correlation, the plot also displays significant heteroscedasticity, indicating difficulties to model distant relationships. However, in many use cases best hits are more relevant than distant objects.

Various databases are built, indexing different sets of sequences: Training only, validation only, test only, and heterogeneous databases from combinations. Similarity search performance is estimated with top- k accuracy, which measures whether the true best hit is found among the k -nearest neighbor in the embedding space.

²scikit-learn MDS: <https://scikit-learn.org/stable/modules/generated/sklearn.manifold.MDS.html#sklearn.manifold.MDS>, accessed 2020/05/28

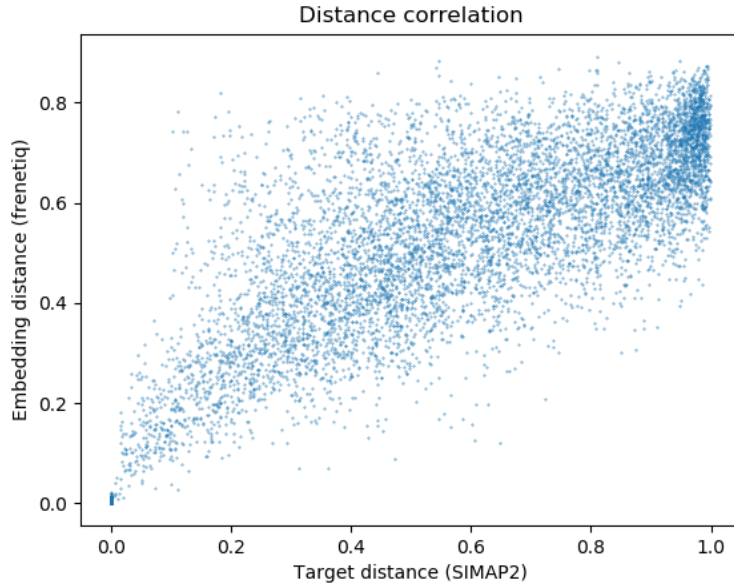


Figure 5.2.: Correlation between ground truth and embedding distances

This is motivated from employing *freneti* in a filter-and-refine approach: Some $k \ll n$ candidate neighbors are filtered from n indexed objects rapidly in vector space. Exact alignments subsequently sort the candidate list. For a fixed low value of k , this is sufficiently fast, and does not increase computational complexity.

In Figure 5.3, top-k accuracy is reported for $k = 1..100$. Top-1 accuracy is at least doubled compared to the proof-of-concept, increasing to approximately 50% and 70% in validation and training, respectively, indicating worse than optimal generalization.

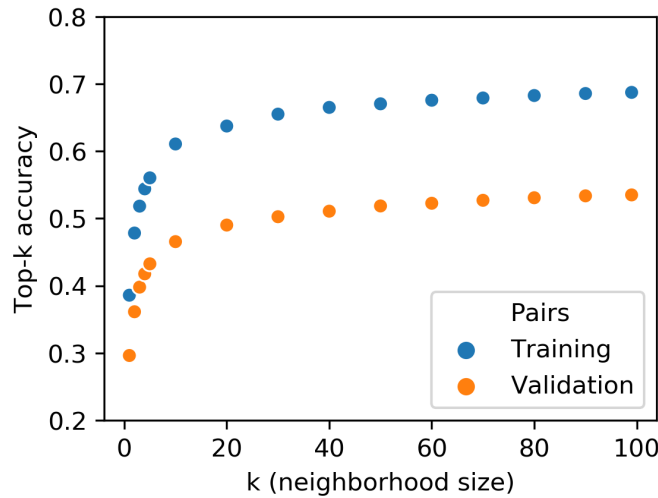


Figure 5.3.: Similarity search accuracy in *freneti* embedding space with SIMAP2-based ground truth

We investigate similarity search speed by querying sequences against a database of 1 000 000 on a multicore CPU. Exact nearest neighbor search takes 32.0 ms, where the embedding and search each consume approximately half of the total time. For

comparison, the same queries take 24 ms with MMseqs2 (mmseqs2faster parametrization [46]). Employing the HNSW index, each query takes 15.4 ms. All values are averages over 5000 query sequences, of mean length 361. HNSW indexing takes 4 h. When using a GPU for embedding, time is reduced significantly to 19 ms for exact search, and 2.5 ms for HNSW search (see also Figure 5.4). This shows the potential of *frenetiq*+HNSW for very fast protein similarity search. No efforts were made to optimize the HNSW index, as it was deemed unnecessary, before the underlying *frenetiq* embeddings reached satisfactory quality. For the same reason, accuracy was not investigated.

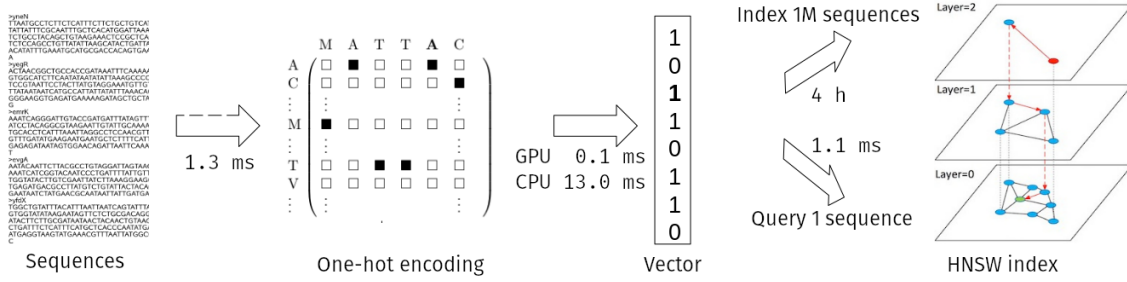


Figure 5.4.: Timings of *frenetiq* embeddings with HNSW index

Exploratory experiments are performed with a number of different network architectures:

- Residual networks (ResNet [156]) with residual connections or densely connected convolutional networks (DenseNets [204]) that allow to train very deep networks
- Temporal convolutional networks (TCN) with dilated convolutions [205], which increase the receptive field exponentially with increasing network depth [206]
- Recurrent networks (LSTM, GRU) with internal states that allow to model long-range interactions and variable length sequences [154], [207]
- Transformer networks (BERT) with dot-product attention [68], [194], [208] that have defined the state-of-the-art in many natural language processing tasks since 2018 [209]

Due to time constraints and limits in computational resources no comprehensive evaluation of these networks could be carried out. In short training runs, none of the above could learn faster than the standard convolutional network described above to similar accuracy. This is in contrast to initial expectations, that especially TCNs, RNNs, and Transformers should be better suited to sequential data. The reasons for performance below expectations remain unknown. Possible explanations include inefficient training procedures, suboptimal parametrization, too short training, implementation bugs, or general inaptitude to solve this particular problem.

5.4. Conclusion and outlook

In the *frenetiq* project, protein sequence vector embeddings were investigated. Deep convolutional networks are used to embed protein sequences from a one-letter amino

acid code to m -dimensional vector spaces. The networks are used in doublets, and trained so that Euclidean distances between sequence embeddings reflect ground-truth dissimilarities obtained from the SIMAP2 database of exact Smith-Waterman sequence alignments.

Experiments on doublet convolutional networks showed the potential to learn semantically meaningful protein sequence vector representations, preserving nearest neighbor relations from alignment similarity networks to some degree. Initial experiments with more sophisticated architectures yielded inconclusive results, and will require in-depth analysis in order to identify bottlenecks towards high retrieval accuracy.

Multiple improvement possibilities are easily conceived, and several examples are outlined below in reference to the *frenetiq* pipeline as depicted in Figure 5.1.

- The inflexible one-hot encoding can be easily replaced by a tunable word embedding layer, as used, for example, in DeepNOG (see Chapter 6). Biochemical similarities between amino acids can then be modeled already in the earliest layer.
- Vast optimization potential is expected for the ‘Deep network’ part. Convolutional layers can not model long-range interactions. Pretrained protein “language” models [195], [196] are a promising alternative. Additional models from recent methods, such as the Reformer [210] might be expected to appear.
- Learning Hamming spaces could boost retrieval speed [211], [212] further.
- Loss function modifications could penalize errors for close neighbors stronger than those for distant neighbors, emphasizing on the most relevant relations.
- Margin-based losses might improve training [213].
- Regularization, such as Dropout [214], should improve generalization.
- Experiments could not use all scores stored in SIMAP2 due to hardware limitations. More GPU/other accelerators would allow to incorporate more observations, train more epochs, and evaluate more hyperparameter settings.
- Doublet networks perform self-supervised learning, ignoring the vast biological knowledge available for many protein sequences used here. Additional supervised training signals might help the network to learn better representations. Indeed, preliminary experiments were performed with multitask learning incorporating protein family classification losses (eggNOG and Pfam). While neighbor retrieval was not improved, classification showed promising results, motivating the protein orthology prediction project presented in Chapter 6.

Very recently, promising results on protein sequence embeddings using information from structure were reported [215]. We consider this project’s output preliminary results, and aim to continue work. If the protein sequence representations can be improved further, and approximate neighbor search indexed can be tuned for high retrieval accuracy with short query times, this methodology would allow to alleviate the computational burden of sequence similarity search. The resulting tools would be highly valuable resources for researchers working on proteins in many disciplines of life science.

The learned representations will be used in the course of this dissertation for hubness-related experiments on large protein datasets.

6. Protein orthologous group prediction

The primary aim of this doctoral thesis is the investigation of hubness in protein sequence similarity networks. Approximate hubness reduction methods (see Chapter 3) necessitate vector objects. Protein sequence vector representations obtained by deep learning are, therefore, explored in Chapter 5.

During method development, various strategies for improving the representations were examined. One particular concept involved *multitask* learning: Instead of solely training the deep network for distance properties between pairs of sequences, additional supervised learning criteria were introduced. For example, protein family [216] and/or orthologous group [174] prediction criteria were jointly trained. While no apparent improvements concerning the distance properties were achieved, conspicuous classification performance potential was observed.

In this work package, a deep learning method for protein orthology inference (DeepNOG) is conceived, evaluated, and implemented in the free open source software package `deepnog`, confirming Hypothesis H₅.

Plans for future work include DeepNOG integration in the workflow of microbial phenotypic trait prediction with `phenotrex` (cf. Section A.2.3), and PhenDB [217]. Additional work will investigate the application of DeepNOG for mapping viral proteins to orthologous groups in the context of VOGDB [218]. These works are beyond the scope of this dissertation.

Submitted for publication Feldbauer, R., Gosch, L., Lüftinger, L., Hyden, P., Flexer, A., and Rattei, T. *DeepNOG: Fast and accurate protein orthologous group prediction*. Submitted for publication in *Bioinformatics* (Oxford University Press), confirmation of receipt: April 24th, 2020.

Contributions RF designed the study, and created deep network and software prototypes. LG created the software package. RF, LL, PH contributed to software development. LG, RF, LL wrote software documentation. RF and LG conducted experiments and analyzed results. PH conducted additional experiments. RF wrote the manuscript draft, partially building upon a technical report by LG. All authors provided feedback on the manuscript. TR and AF supervised the work.

Sequence analysis

DeepNOG: Fast and accurate protein orthologous group prediction

Roman Feldbauer¹, Lukas Gosch¹, Lukas Lüttinger^{1,2}, Patrick Hyden¹, Arthur Flexer³ and Thomas Rattei^{1,*}

¹Department of Microbiology and Ecosystem Science, University of Vienna, Vienna, 1090, Austria

²Ares Genetics GmbH, Vienna, 1030, Austria

³Institute of Computational Perception, Johannes Kepler University Linz, Linz, 4040, Austria

*To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: Protein orthologous group databases are powerful tools for evolutionary analysis, functional annotation, or metabolic pathway modeling across lineages. Sequences are typically assigned to orthologous groups with alignment-based methods, such as profile hidden Markov models, which has become a computational bottleneck.

Results: Here we present DeepNOG, an extremely fast and accurate, alignment-free orthology inference method based on deep convolutional networks. We compare DeepNOG against the state-of-the-art tools HMMER and DeepFam on two orthology databases. DeepNOG consistently achieves higher or equal prediction accuracy, while computing time is reduced up to an order of magnitude on CPUs. A lightweight command-line tool enables rapid adoption by users, with optional GPU usage for further increased throughput.

Availability: Source code and packages are freely available at <https://github.com/univieCUBE/deepnog>. Install the platform-independent Python program with `$ pip install deepnog`.

Contact: thomas.rattei@univie.ac.at

Supplementary information: Supplementary material is available at *Bioinformatics* online.

1 Introduction

Understanding protein function is a fundamental problem in molecular biology. Characterizing function through biological experiment has very slow throughput, compared to the ongoing growth of sequence data. While functional characterization is time consuming even in model organisms, experiments become particularly complex for currently uncultivable or otherwise inaccessible organisms, such as, for example, intracellular symbionts. The continuous massive stream of sequence data emerging from large-scale genomic and metagenomic projects widens the gap between available raw information and precise data from experiments. Highly efficient and accurate computational methods are, thus, required to extract functional and evolutionary information from all sequence data.

Orthologous relationships are highly informative about protein function (Fitch, 2000; Gabaldón and Koonin, 2013). Phylogenetic and functional information is largely stored in the primary structure

of proteins, i.e. the amino acid sequence. Several public resources provide precomputed orthologous groups of protein sequences to aid, for example, comparative genomics and phylogenetics. Among these resources are hand-curated databases, such as COG (Galperin *et al.*, 2015), as well as (semi-)automatically created databases, such as KEGG Orthology (Kanehisa and Goto, 2000), OMA (Altenhoff *et al.*, 2018), and eggNOG (Huerta-Cepas *et al.*, 2019). The latter is a superset of COG, and additionally comprises over 4 M orthologous groups automatically created from fully sequenced and assembled genomes of 5090 organisms in its latest iteration eggNOG 5. Mapping protein sequences against these well-annotated orthology resources is a popular approach to functional inference, evolutionary classification, metabolic pathway analysis, and more (Galperin *et al.*, 2019).

Several orthology mapping methods exist, including alignment-based and alignment-free techniques. Probabilistic profile hidden Markov models (pHMMs) are derived from multiple sequence alignments (Eddy, 1998), and achieve state-of-the-art performance in detecting homologous protein

sequences. The exact alignment of multiple protein sequences is known to be an NP-complete problem (Wang and Jiang, 1994), and heuristic algorithms are used in practice (see, for example, Sievers et al., 2011). Hidden Markov models are trained from multiple alignments of protein families or orthologous groups. Inference with all models is then required to assign one particular family or group to a sequence of interest. This has proven to be a limiting factor in previous studies (Feldbauer et al., 2015; Seo et al., 2018). Alignment-based methods are, thus, becoming a computational bottleneck in protein function prediction. This is especially the case for current metagenomic projects. Due to readily available low-cost high-throughput sequencing technologies, millions and soon to be billions of proteins await analysis (see, for example, Pasolli et al., 2019; Nayfach et al., 2019; Almeida et al., 2019).

Recently, alignment-free deep learning approaches have been applied successfully to numerous biological and biomedical tasks (Angermueller et al., 2016; Min et al., 2017; Eraslan et al., 2019). A deep convolutional network for orthologous group predictions called DeepFam has been published by Seo et al. (2018). While it achieves high prediction accuracy and significant speed-up compared to the commonly used pHMM tool HMMER3 (Eddy, 2009), we identify several limitations:

- Sub-optimal scaling to larger databases, and therefore
- COG only (no eggNOG, etc.)
- Sequence length restrictions
- Missing user interface for inference

To fill these gaps, we introduce *DeepNOG*, a new convolutional network architecture which successfully overcomes the above-mentioned limitations. DeepNOG features superior prediction accuracy and computational efficiency on COG, and, in addition, allows to predict orthologous groups from eggNOG 5. It handles proteins of arbitrary sequence lengths. Finally, the Python package *deepnpg* provides users with the practical means to straight-forward deep learning-based orthologous group prediction.

2 Materials and methods

In this section, we introduce the orthology databases and deep network architectures used for our experiments.

2.1 Orthology databases

We use two orthology databases for our experiments: Our primary interest is eggNOG, a large public database with semi-supervised, fine-grained orthologous groups on multiple levels of the tree of life. We will demonstrate fast and reliable orthologous groups prediction in the latest eggNOG version 5. COG is a comparatively small, manually curated database by NCBI that serves here for comparison to previous results of competing methods.

Table 1 gives an overview of the most important characteristics and differences between COG (2014) and eggNOG 5. For the latter, we consider sequences with exactly one orthologous group (OG) assignment on the bacteria level. The population of orthologous groups is typically

highly skewed. While the expected population size of each OG on top taxonomic levels is roughly on the order of one hundred members, there exist many OGs comprising few members, or even a single sequence (singletons). Conversely, relatively few groups with high cardinality contain a large fraction of all sequences in the database. In the machine learning context, datasets highly imbalanced with respect to class cardinalities pose a challenge to both traditional and modern methods (Johnson and Khoshgoftaar, 2019), some consequences of which will be discussed in the results section 4.1. When technically necessary, we will disregard OGs below a certain member threshold. This typically only affects a relatively low number of sequences.

2.1.1 COG: Clusters of orthologous genes

The Clusters of Orthologous Genes¹ database was introduced in 1997 to provide evolutionary classification of protein families. COG is a relatively small, but manually curated and, thus, high quality orthology resource. Orthologous groups are constructed by considering bidirectional best hits of sequences comparing complete genomes.

COG is used by Seo et al. (2018) to evaluate the convolutional network architecture DeepFam, and to compare its speed and accuracy with HMMER. Here, we use COG to reproduce and verify the results reported for DeepFam, and on the other hand as a baseline to compare the new DeepNOG architecture with its competing methods DeepFam and HMMER.

2.1.2 eggNOG 5

The *evolutionary genealogy of genes: non-supervised orthologous groups* database (eggNOG) builds upon the COG database. In addition to this supervised part, a large fraction of the database is constructed by an unsupervised clustering algorithm (Jensen et al., 2007). The resulting orthologous groups are often referred to as NOGs or ENOGs. In its current version 5, eggNOG consists of 4.4 million orthologous groups distributed across 379 taxonomic levels. For this work, we primarily consider single-label proteins² from the bacterial level. The bacteria level is highly relevant for metagenomic studies. For example, reliable phenotypic trait prediction from complete or incomplete genomes has so far been demonstrated for bacteria only (Feldbauer et al., 2015; Weimann et al., 2016), and is used, for instance, in the PhenDB pipeline.³ The methodology is, however, not limited to specific taxonomic levels, as will be shown exemplarily on the eggNOG 5 root level.

2.2 Convolutional networks for OG prediction

Both the newly developed DeepNOG architecture and DeepFam rely on convolutional units to extract informative subsequence patterns from proteins for orthologous group prediction. That is, both are supervised *end-to-end* learning methods, which do not require manual feature extraction, such as kmer frequencies, or others. Instead, they take raw protein sequences as input and return class labels for each sequence. In this section, we will describe common features and important differences between these methods.

2.2.1 Architectural comparison of DeepNOG and DeepFam

DeepNOG is a convolutional network architecture inspired by DeepFam with multiple improvements to overcome its limitations including the restriction to fixed length protein sequences, and applicability to eggNOG. Figure 1 provides a visual overview of the DeepNOG architecture (see Suppl. Fig. S1 for a comparison to DeepFam). The following subsections give a

Table 1. Main characteristics of orthologous groups databases

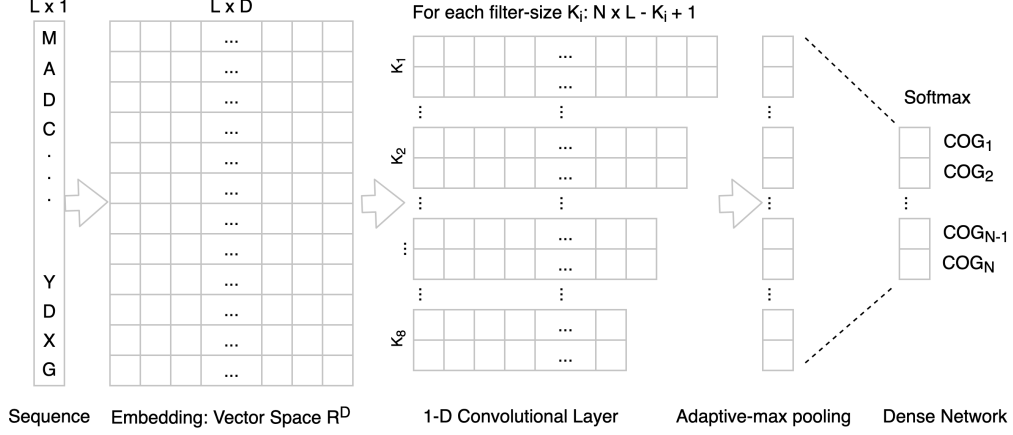
	COG 2014	eggNOG 5 (bacteria)
Number of proteins	1 674 176	13 836 642
Number of OGs	4631	206 782
OG population range	1–10 632	1–96 355
Sequence length range	21–29 202	23–24 921

¹ Alternative name: "Clusters of Orthologous Groups of proteins"

² Single-label proteins are associated with only one NOG, which excludes, for example, multi-domain proteins.

³ PhenDB: <https://phenodb.csbs.univie.ac.at/>

Fig. 1. DeepNOG network architecture



brief overview of DeepFam, and introduce the architectural improvements incorporated into DeepNOG.

2.2.2 Encoding layer: word embeddings

Proteins are commonly represented in human-readable form, such as the FASTA format, which contains the sequence encoded in the IUPAC one-letter amino acid code. For deep learning, sequences must be transformed into a format processible by deep networks, typically numerical vectors. Therefore, categorical data are frequently embedded into vector spaces by one-hot encoding: Vector length is determined by the alphabet size (for example, 22 amino acids), and the vector for a particular amino acid is 1 in a cell specific for this amino acid, and 0 everywhere else. One-hot encodings hence exhibit equidistance between all categories. Clearly, this is an undesirable feature for amino acids, which naturally cluster with respect to chemical and biological features or size. DeepFam employs a pseudo one-hot encoding, in which the 21 standard amino acids are one-hot encoded, and three ambiguous codes are allowed to interpolate between their manifestations (e.g. $J = \frac{1}{2}I + \frac{1}{2}L$). In sequence alignments, substitution matrices such as BLOSUM, PAM, or PSSMs are employed to factor in similarities between individual amino acids. DeepNOG accounts for these similarities by embedding each amino acid into $\mathbb{R}^D$ ('word embedding'). These **D -dimensional vector representations** are jointly trained with the network. That is, the encoding layer has flexibility to learn common properties of amino acids by embedding them in a way so that distances in $\mathbb{R}^D$ reflect biochemical (dis-)similarities. The encoded sequences are passed to feature extraction in the convolutional layer (see Section 2.2.3). Conveniently, the learned amino acid embeddings can be visually inspected for biological plausibility, which enhances the interpretability of trained models (see Figure 2). The word embedding dimension can be lower than the alphabet size, and thus, lower than the one-hot embedding dimension. For DeepNOG we find an amino acid embedding dimension $D = 10$ to give best results on validation data.

2.2.3 Convolutional layer

After the encoding, a 1-D convolutional layer is employed in DeepFam. The i -th row in the convolutional layer corresponds to a filter of size K_i which is applied to $L - K_i + 1$ consecutive (encoded) protein subsequences of length K_i resulting in $L - K_i + 1$ convolutional units. 1-D means that the filter treats each column in the encoding as a separate input channel. If the

amino-acid alphabet consists of C letters, this results in the filter learning C vectors of weights, each of length K_i . Therefore, the k -th element in a weight vector corresponds to the weight given to a certain amino acid occurring in the k -th position of the looked upon protein subsequence. The 1-D convolutional operation defining the activation of the convolution unit h_{ij} is formalized as

$$h_{ij} = \sigma \left(b_i + \sum_{c=1}^C \sum_{l=1}^{K_i} X_{j+l-1,c} W_{i,c,l} \right),$$

where b_i is the bias term, σ the activation function (non-linearity), and W denotes weights, that is, the tunable network parameters. DeepFam uses rectified linear units (ReLU) as activation functions (Glorot *et al.*, 2011). Furthermore, it chooses eight different filter sizes (8, 12, 16, 20, 24, 28, 32, and 36). In the most successful parametrization, it uses 250 different filters for each filter-size resulting in 2000 independent filters. Seo *et al.* (2018) showed that the learned filters are sensitive to sequence motifs typical for certain orthologous groups, and their occurrences can distinguish orthologous groups, underlining successful feature extraction by the convolutional layer.

DeepNOG's convolutional layer closely resembles the architecture of DeepFam, but exchanges the ReLU activation function for scaled exponential linear units (SELU) to obtain a **self-normalizing network**: SELUs guarantee activations to converge towards zero mean and unit variance, without an additional explicit normalization step, such as batch normalization. This eliminates the vanishing or exploding gradients problems, enables strong regularization, and yields stable learning (Klambauer *et al.* (2017); see also Section 2.2.6).

In DeepNOG, employing a 1-D convolution means that each dimension in the embedding vector space $\mathbb{R}^D$ is treated by the filter as an independent input feature vector. The filter then weights the importance of the positive or negative strength of a feature at a certain position in the input protein subsequence for each input feature vector separately. The positive or negative strength of a feature at a certain position is determined by the concrete amino acid in this specific position in the subsequence. Thus, the interpretation of filters getting sensitive to discriminatory motifs for OG predictions, as put forth by Seo *et al.* (2018), holds.

2.2.4 Pooling layer

DeepFam employs a standard 1-max pooling layer. This means, the i -th node in the pooling layer corresponds to the maximum value in the i -th row of the convolutional layer and all other values are discarded. 1-max pooling layers require the length of the input rows to be specified exactly. For this reason, DeepFam fixed the length L of input sequences to 1000, zero-padded smaller input sequences, and discarded longer ones. Both COG and eggNOG, however, comprise sequences of very high length variability (see Table 1). Therefore, this is a significant limitation for the applicability of this architecture to arbitrary sequences in COG and eggNOG as well as for classifying arbitrary user sequences.

To this end, DeepNOG exchanges the pooling layer for an **adaptive max-pooling layer**. It also extracts the maximum value in an input row of the convolutional layer, but can handle arbitrary input sizes and has, therefore, no upper bound on the length of the input sequences. Extremely short input sequences of lengths smaller than the biggest filter size K are zero-padded to length K for training and inference (where $\max(K) = 36$ in DeepNOG). DeepNOG has, consequently, **no restriction on protein sequence length**.

2.2.5 Classification layer

DeepFam aggregates activations from the feature extraction subnetwork in a fully-connected subnetwork with one hidden layer followed by a softmax for classification. In the most successful parametrization of DeepFam, it uses 2000 units in the hidden layer. The output at each node can be interpreted as the confidence the network has in associating the input protein sequence with the specific orthologous group corresponding to the node (Goodfellow *et al.*, 2016). The OG prediction is finally based on the highest confidence among the output nodes.

DeepNOG directly places the softmax output layer after the adaptive max pooling layer, thereby omitting the additional hidden layer. This **reduces the total number of parameters** and improves run time.⁴ Since softmax layers are equivalent to logistic regression, DeepNOG can be understood as a two-stage machine learning model. In the first stage, it extracts the (strength of) occurrence of sequence motifs with discriminatory information regarding OGs. In the second stage, it applies logistic regression, using the (strength of) occurrence of sequence motifs as input, to calculate probabilities of all orthologous groups in the model. Finally, the input sequence is assigned to the OG with the highest probability.

2.2.6 Hyperparameters and training procedure

DeepNOG is trained as a self-normalizing network by employing scaled exponential linear units (SELU) as activation functions. DeepNOG with SELUs achieves better empirical results (on validation data) than with ReLU plus batch-normalization (Ioffe and Szegedy, 2015) approach taken by DeepFam, while at the same time being computationally more efficient. The convolutional and dense layers of DeepNOG are initialized as described in Klambauer *et al.* (2017). Dropout with $p = 0.3$ is employed for regularization. Additionally, alpha-dropout with $p = \{0.05, 0.10, 0.20, 0.30\}$ as suggested in Klambauer *et al.* (2017) as well as dropout with $p = 0.5$ as suggested by Hinton *et al.* (2012) were investigated, but found to perform inferior (on validation data). Due to computational limitations and the similarity of COG to eggNOG, the performance of hyperparameter choices on COG was used as an estimator for the performance on eggNOG. Therefore, DeepNOG uses identical parametrization and the same filter sizes 8, 12, 16, 20, 24, 28, 32 and

36 as DeepFam and 150 filters for each size. For stochastic optimization, Adam (Kingma and Ba, 2014) is used with a learning rate set to 0.01 in conjunction with a learning rate scheduler decreasing the learning rate by a factor 0.75 after each epoch. Additional Adam parameters are set to default values. Batch size is set to 16. Training is performed on an Nvidia P6000 GPU with 24G memory. For some experiments, the memory requirements of the best DeepFam parametrization exceed the total memory on the available hardware. Therefore, an additional parametrization is trained, which uses 150 instead of 250 convolutional filters for each filter size, and 1500 instead of 2000 hidden units in the classification layer. These design decisions are in accordance with investigations of different hyperparameter choices in Seo *et al.* (2018), which reported only a minor increase in error ($< 1\%$ -point) for the more lightweight parametrization. We will refer to these models as 'DeepFam light'.

2.2.7 Training, validation, test splits

Unbiased estimation of the true performance of machine learning models requires splitting data into independent training, validation, and test sets, or nested cross-validation. We employ different split strategies for the experiments described below.

Experiments on COG employ a single, 'flat' cross-validation. While likely resulting in too optimistic performance estimates, this procedure is identical to the estimation in Seo *et al.* (2018). Using the exact same splits allows for fair comparison of DeepNOG to previous results of DeepFam.

For eggNOG experiments, we refrain from cross-validation given the large database size. Models are trained and selected on training/validation splits. Prediction performance is estimated on an independent test set. All splits are stratified according to class labels, with a split ratio of 81%, 9%, and 10% of all sequences, for training, validation, and test sets, respectively.

This procedure might still report optimistic error rates for biological sequences, if highly similar sequences are present in both training and test set. For this reason, we train and evaluate an additional model on eggNOG 5 sequences clustered according to UniRef50 (release 2019_11, Suzek *et al.*, 2014) to estimate possible biases of this type. We do not employ clustering generally, however, because it reduces the number of sequences drastically, so that fewer ENOGs have enough members for robust predictions. In addition, we do not strive to eliminate homology effects, since the very purpose of DeepNOG is to detect orthologous relationships.

2.3 Profile hidden Markov models

DeepNOG is also compared against alignment-based profile hidden Markov models (pHMMs), which achieve state-of-the-art performance in orthologous group prediction. For experiments on COG, we transfer results reported in Seo *et al.* (2018), because identical datasets and splits are used.

For eggNOG experiments, multiple sequence alignments are built for each orthologous group with FAMSA v1.3.2 (Deorowicz *et al.*, 2016) restricted to training set proteins. Profile HMMs are created from these alignments with `hmmbuild` (HMMER 3.3). Test set inference is performed by `hmmsearch`. Default parameter values are used, unless explicitly specified otherwise. Each query is assigned to the NOG with the lowest corresponding inference e-value. Queries without significant hits (e-value > 0.05) are treated as misclassified. Inference speed is measured with `hmmsearch` with parameters `--noali --cpus 1`.

3 Implementation

The Python tool `deepnog` provides fast protein orthology predictions with PyTorch deep networks (Paszke *et al.*, 2019). At the time of writing, it supports the DeepNOG architecture as described in Section 2.2 trained on the root (tax 1) and bacteria (tax 2) levels of the eggNOG 5 database out of the box. Models for further taxonomic levels can be trained on demand. In

⁴ The number of model parameters is not necessarily a good measure of its complexity, and deep networks have been shown to generalize better in overparametrized regimes (Belkin *et al.*, 2018). Here, reducing the number of parameters should be seen in the light of increasing training and inference speed rather than as a regularization method.

general, the tool is independent of specific network architectures. Additional networks for arbitrary orthology databases can be registered easily. This requires only the PyTorch model definition, and the corresponding trained weights. For *deepnog* usage details, and extension to support custom networks, the reader is referred to the user & developer guide available online.⁵

deepnog can be installed from the Python Package Inventory by `$ pip install deepnog` on all major operating systems. Source code, issue tracker, and additional links are available on <https://github.com/univieCUBE/deepnog>. Further information about software development principles, tests, continuous integration, or documentation is available in Suppl. Section A. Refer to Suppl. Section B for notes on data availability.

4 Results

We evaluate DeepNOG on both COG and eggNOG 5, and compare it with DeepFam and HMMER, where applicable. DeepFam was originally implemented with the TensorFlow framework. For fair comparison, we re-implemented the network in PyTorch, and reproduce the findings reported by Seo *et al.* (2018) for the COG database in Section 4.2, where we also compare DeepNOG with HMMER.

We then proceed to the larger eggNOG 5 database, for which we report fast and accurate predictions with DeepNOG in Section 4.3.

4.1 Addressing imbalanced orthologous group populations

Both COG and eggNOG 5 constitute highly imbalanced datasets with respect to group membership. Learning to correctly assign orthologous groups with population sizes below a certain threshold is challenging. For example, in the extreme case of singletons, it is impossible to perform train-test splits. Generalization performance cannot be estimated in such cases. Therefore, Seo *et al.* (2018) introduced three minimum population thresholds for OGs (100, 250 and 500) and investigate, how the method scales with the number of OGs and level of imbalance. We consider the minimum population thresholds of 100 and 500 for comparison between DeepNOG and DeepFam.

Imbalanced datasets also require a careful choice of performance measures. For direct comparison with previous results we report classification accuracy, which is the fraction of correct predictions in all predictions. This measure is heavily biased towards the performance of the largest groups. In order to investigate prediction performance for orthologous groups with few members, we also report *mean accuracy*, which we define as the accuracy per group, averaged over all groups. Each group, therefore, contributes equally to the overall performance. If accuracy is higher than mean accuracy, this suggests the model performs better on large groups than on small groups.

4.2 DeepNOG vs. state-of-the-art methods on COG

DeepFam processes sequences to a maximum length of 1000. Applying the length threshold to COG removes 1.3 % of the protein sequences (see Suppl. Table S1). Additionally requiring a minimum population of 100 removes roughly 6.5 % of sequences.

Table 2 reports classification accuracy of DeepNOG and competing inference methods on two COG subsets. The results are obtained by averaging three-fold cross-validation results. Datasets and splits are identical to those used by Seo *et al.* (2018) to allow for a fair comparison. Profile hidden Markov models (pHMMs) generated with HMMER serve as baseline for orthologous group prediction. We investigate a lightweight

Table 2. Prediction accuracy for COG

Method	(Total) accuracy in %		Mean accuracy in %	
	COG-500	COG-100	COG-500	COG-100
pHMMs*	91.75	91.67	-	-
DeepFam	95.40*	91.40*	95.68	85.87
DeepFam light	94.40	87.27	94.35	77.10
DeepNOG	95.66	94.24	96.07	93.03

Notes: COG-500 and COG-100 refer to COG-500-1074 and COG-100-2892 in Seo *et al.* (2018), resp. Results marked with * taken from the same source. Bold scores indicate best performance per dataset and measure.

version ('DeepFam light') in addition to the best parametrization of DeepFam, which was not trainable on the available resources for eggNOG (see Section 2.2.6). DeepNOG yields the best performance on all datasets, and improves upon the other methods especially on larger datasets, indicating superior scaling behaviour to higher numbers of orthologous groups.

We observe marked differences between total and mean accuracy on COG-100 for both DeepFam parametrization with 6–10%-pts lower mean accuracy, indicating reduced performance on less populated OGs. On the other hand, DeepNOG performs approximately equally considering total and mean accuracy, implying robust prediction among orthologous groups of different sizes.

Inference speed is an important constraint for applications of orthologous group prediction. Again, the deep learning based methods are compared against the HMMER baseline. For fair comparison, all timings were performed on the same machine on a single CPU core. (1000 proteins, Intel(R) Xeon(R) CPU E5-2609 v2 @ 2.50GHz, average of 5 trials) DeepNOG is slightly faster than DeepFam, and **up to an order of magnitude faster than HMMER** (see Table 6).

We conclude that DeepNOG (i) provides the highest prediction performance, (ii) scales better to higher numbers of groups and more imbalanced datasets, and (iii) is faster than its competitors on COG.

4.3 Evaluation of eggNOG prediction

Having established favourable performance and speed of DeepNOG compared with its competitors, we turn to the larger eggNOG 5 database, which is our primary interest.

Table 3. eggNOG 5 (bacteria level) datasets due to population thresholds

Name	Min. pop.	OGs	% sequences	length restriction
NOG	-	206 782	100.00	-
NOG-100	100	6217	90.20	-
NOG-500	500	3373	85.87	-

Similar to our experiments on COG, we apply minimum orthologous group population thresholds to eggNOG. Since DeepNOG is designed to handle sequence of arbitrary length, we do not apply any sequence length restrictions to eggNOG. Table 3 describes the resulting datasets used for the experiments in this section.

Table 4 reports prediction performance of DeepNOG for eggNOG 5 datasets. We compare our method with DeepFam light, since these datasets contain more classes (orthologous groups), so that the best DeepFam parametrization exceeds the available memory (see Section 2.2.6).

DeepNOG shows very high prediction accuracy on the eggNOG 5 bacteria datasets. While the number of orthologous groups is more than doubled compared to COG, performance measures are nearly stable, with

⁵ User & developer guide: <https://deepnog.rtdf.io>

Table 4. Prediction accuracy for eggNOG 5 (bacteria level). Best results bold.

	(Total) accuracy in %		Mean accuracy in %	
Method	NOG-500	NOG-100	NOG-500	NOG-100
pHMMs	93.10	93.07	92.47	89.41
DeepFam*	n/a	n/a	n/a	n/a
DeepFam light	83.23	78.95	68.59	37.81
DeepNOG	93.18	93.17	93.14	92.73

Note: * DeepFam not trainable on 24G GPU for eggNOG

accuracy slightly below COG results. Again, we observe only minute differences between total and mean accuracy, clearly demonstrating DeepNOG's excellent scaling to larger numbers of classes and population imbalance. DeepNOG also outperforms DeepFam light with at least ten percent-points difference in accuracy, and a striking 55%-pts difference in mean accuracy on NOG-100, indicating substantial difficulties of DeepFam to predict rare orthologous groups correctly (see also Suppl. Figure S2). Hidden Markov model performance is on par with DeepNOG.

DeepNOG processes arbitrary taxonomic levels, and uses one unique model per level. An additional model is trained for the eggNOG 5 root level. Data is prepared analogously to the NOG-100 bacteria dataset, yielding a total of 18 275 264 sequences in 14 460 NOGs. Table 5 (top row) demonstrates high performance for orthologous groups spanning the three domains of life.

Table 5. DeepNOG prediction accuracy for eggNOG 5 (root level) clustered to different sequence identity thresholds.

Train/val	Test set	(Total) accuracy in %	Mean accuracy in %
UniRef100	UniRef100	93.20	89.94
UniRef50	UniRef50	85.00	83.70
UniRef50	UniRef100	89.36	88.44

In order to control possible biases stemming from similar sequences in training and test sets, we repeat the experiment with sequences clustered to 50% identity (UniRef50). Clustering results in a dataset comprising 1 849 409 sequences, a size reduction of 87%. We investigate two scenarios in order to control the influence of sequence identity versus dataset size (see Table 5). Testing independent UniRef50 or UniRef100 sequences reveals contributions of both effects toward reduced accuracy, and indicates slightly elevated performance for highly similar sequences compared to less similar sequences.

Table 6 provides execution times on the eggNOG 5 bacteria datasets. The experiment setup is identical to the setup for COG (see Section 4.2). The timings for profile hidden Markov models are in alignment with our initial hypothesis, that they are becoming a computational bottleneck. As initially hypothesized, pHMM inference time scales unfavorably with the number of classes, and runtime increases significantly with increasing numbers of classes (see also Suppl. Figure S3). The deep learning based methods are faster overall, and scale decisively better to many classes. At the same time, DeepNOG models have a smaller memory footprint than pHMMs. The network weights for NOG-500 and NOG-100 amount to 51.9M and 92.9M, respectively. In comparison, the corresponding pHMMs require 1.2G and 1.7G, respectively.

In summary, we find that DeepNOG scales well from smaller databases like COG to larger databases like eggNOG. DeepNOG shows powerful performance on eggNOG 5 in terms of prediction accuracy, execution speed, and memory footprint. It clearly outperforms DeepFam in all of the evaluated metrics, and provides accuracy on par with pHMMs at a fraction of their computational cost.

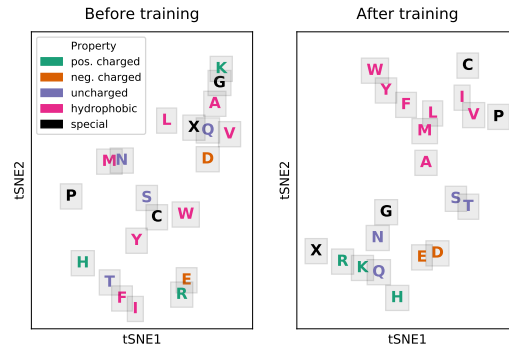
Table 6. Inference time (seconds/1000 sequences) for COG and eggNOG (bacteria level). Fastest method bold.

Method	COG-500	COG-100	NOG-500	NOG-100
pHMMs	96.3	207.0	218.9	253.7
DeepFam	34.1	35.6	n/a	n/a
DeepFam light	23.1	24.7	24.9	26.3
DeepNOG	21.8	22.9	23.5	24.7

4.3.1 Amino Acid Embedding

One important feature of the DeepNOG architecture is the word embedding layer used for encoding the input amino acids. Figure 2 visualizes the amino acid embeddings of the DeepNOG eggNOG 5 (bacteria) model before and after training. The embeddings in $\mathbb{R}^{10}$ are reduced to $\mathbb{R}^2$ by t-SNE (Maaten and Hinton, 2008). Amino acids are initially positioned randomly in $\mathbb{R}^{10}$. Consequently, no meaningful cluster structures are visible before training. After training, the learned embeddings exhibit biologically plausible clusters matching well with biochemical properties. Furthermore, a finer topology can be identified. For example the aromatic amino acids Phenylalanine (F), Tryptophan (W), and Tyrosine (Y) are distinctly grouped together. Furthermore, amino acids with unique features, such as disulphide-bond forming Cysteine (C), or the structural disruptor Proline (P), are located on cluster borders. Note that inter-cluster distances typically hold no meaning in t-SNE visualizations, which optimize distance distributions between neighbors only.

Fig. 2. Amino acid representations in the encoding layer of DeepNOG. Random initialization before training (left), and tuned representations after DeepNOG training on NOG-100 (right). Amino acids are coloured based on biochemical properties.



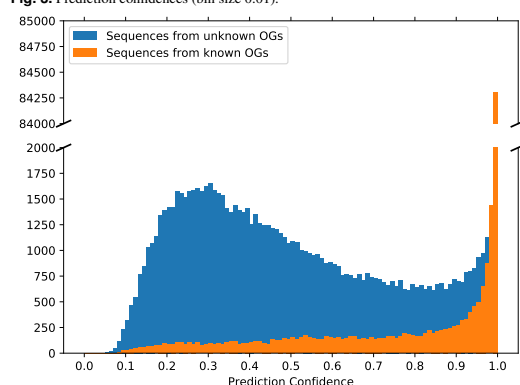
4.3.2 Prediction Confidence Threshold

Classifiers typically *partition* data: Every object is assigned to one of the known classes. This is not always desirable, especially, when there are additional classes that are not part of a model. Any objects from these classes would be labeled incorrectly. For example, human protein sequences fed into a bacteria-level model would be assigned to inappropriate orthologous groups. Instead, one might not want to assign any OG labels in these cases.

Deep networks with a softmax classification layer provide output that represents a probability distribution over the available classes. The output at a single neuron can, therefore, be interpreted as the network's confidence, that the input belongs to the corresponding class. DeepNOG supports setting a prediction confidence threshold to decide, whether an input protein can be associated with any OG included in the model. We investigate the

threshold empirically on the NOG-100 dataset by comparing the probability distribution of sequences from *in-model* OGs to the distribution of sequences from *out-model* OGs. 100 000 proteins are randomly sampled from the NOG-100 test set. These are members of orthologous groups included in the model, but have not been seen during training. Furthermore, 100 000 proteins are randomly sampled from orthologous groups in eggNOG 5 (bacteria) that are not included in NOG-100. Their true corresponding OGs are not included in the model. All proteins are classified with DeepNOG and the probabilities of the assigned groups are investigated. Figure 3 shows the overlaid histograms. In-model sequences show a distinct distribution, well separable from the distribution of out-model sequences. That is, the network is typically highly confident for proteins of known OGs, while in most cases it is much less confident about proteins from OGs unknown to the model.

Fig. 3. Prediction confidences (bin size 0.01).



A sensible confidence threshold can be set to a value on the Pareto boundary of minimizing false positives and false negatives, depending on the specific requirements of downstream experiments. We set the default threshold to 99%. In Figure 3, there are 1438 out-model sequences and 1434 in-model sequences at this level. Bins below are clearly dominated by unknown OGs, while the bin above contains a vast majority of all in-model sequences. This is a rather strict threshold avoiding false positives. *deepnog* allows to set this threshold individually per experiment.

5 Discussion and outlook

We introduced DeepNOG, a deep convolutional network for protein orthologous group inference. It has been shown to achieve high prediction accuracy for both the COG and eggNOG databases on par with alignment-based models (profile hidden Markov models), and superior to the state-of-the-art alignment-free tool DeepFam. At the same time DeepNOG is computationally more efficient, allowing for a higher throughput of protein sequences. We believe that this feature will help researchers to keep up with the never-ending stream of newly sequenced genomes and metagenomes. In addition, we showed that the network extracts biologically meaningful information from sequences, and automatically clusters amino acids in a way that matches biochemical properties. The Python program *deepnog* allows users to effortlessly apply the introduced methods, and classify their sequences for eggNOG 5 groups.

Some limitations of DeepNOG remain to be tackled in future work. We currently provide models only for eggNOG 5 root and bacteria levels.

Training models for additional levels is in principle straight-forward, but requires non-negligible compute resources. We will provide additional models on demand. Second, orthologous groups with very few members still pose a challenge for deep networks, as the number of parameters increases with the number of groups, and few examples are available per class. To this end, we want to investigate softmax approximations, such as, for example, presented by Grave *et al.* (2017). Third, DeepNOG is currently limited to single-label classification, that is, it cannot predict several orthologous group for a single sequence. Future experiments will show, whether eliminating the softmax layer can yield reliable multi-label classifiers.

Finally, we follow recent advances in bioinformatics based on deep learning with great interest, especially the advent of transfer learning. It is widely believed, that transfer learning played a major role in machine learning breakthroughs in computer vision with convolutional networks pretrained on ImageNet (Deng *et al.*, 2009), or in natural language processing with recurrent networks or transformer networks pretrained on large text corpora (e.g., Devlin *et al.*, 2018; Lan *et al.*, 2019). Recently, Rives *et al.* (2019) and Strodthoff *et al.* (2020) reported on unsupervised pretraining of protein sequence representations. While preliminary experiments with UDSProt finetuned for OG prediction did not yet yield improved results, we believe this direction of research to be highly promising.

Funding

This work was supported by the Austrian Science Fund (FWF): [P27703, P31988]; and by the GPU grant program of Nvidia corporation.

Conflict of interest

The authors declare no conflict of interest. LL is employed by Ares Genetics GmbH.

References

- Almeida, A. *et al.* (2019). A new genomic blueprint of the human gut microbiota. *Nature*, **568**, 499–504.
- Altenhoff, A. M. *et al.* (2018). The OMA orthology database in 2018: retrieving evolutionary relationships among all domains of life through richer web and programmatic interfaces. *Nucleic acids research*, **46**, D477–D485.
- Angermueller, C. *et al.* (2016). Deep learning for computational biology. *Molecular systems biology*, **12**, 878.
- Belkin, M. *et al.* (2018). Reconciling modern machine learning practice and the bias-variance trade-off. *arXiv e-prints*, page arXiv:1812.11118.
- Deng, J. *et al.* (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.
- Deorowicz, S. *et al.* (2016). FAMSA: Fast and accurate multiple sequence alignment of huge protein families. *Scientific reports*, **6**(1), 1–13.
- Devlin, J. *et al.* (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv e-prints*, page arXiv:1810.04805.
- Eddy, S. R. (1998). Profile hidden markov models. *Bioinformatics (Oxford, England)*, **14**, 755–763.
- Eddy, S. R. (2009). A new generation of homology search tools based on probabilistic inference. *Genome informatics. International Conference on Genome Informatics*, **23**, 205–211.
- Eraslan, G. *et al.* (2019). Deep learning: new computational modelling techniques for genomics. *Nature reviews. Genetics*, **20**, 389–403.

- Feldbauer, R. *et al.* (2015). Prediction of microbial phenotypes based on comparative genomics. *BMC Bioinformatics*, **16**(14), S1.
- Fitch, W. M. (2000). Homology a personal view on some of the problems. *Trends in Genetics*, **16**, 227–231.
- Gabaldón, T. and Koonin, E. V. (2013). Functional and evolutionary implications of gene orthology. *Nature Reviews Genetics*, **14**, 360–366.
- Galperin, M. Y. *et al.* (2015). Expanded microbial genome coverage and improved protein family annotation in the COG database. *Nucleic acids research*, **43**, D261–D269.
- Galperin, M. Y. *et al.* (2019). Microbial genome analysis: the COG approach. *Briefings in bioinformatics*, **20**(4), 1063–1070.
- Glorot, X. *et al.* (2011). Deep sparse rectifier neural networks. *PMLR*, **15**, 315–323.
- Goodfellow, I. J. *et al.* (2016). *Deep Learning*. Adaptive computation and machine learning. MIT Press.
- Grave, É. *et al.* (2017). Efficient softmax approximation for GPUs. In D. Precup and Y. W. Teh, editors, *34th International Conference on Machine Learning*, volume 70, pages 1302–1310. Proceedings of Machine Learning Research.
- Hinton, G. E. *et al.* (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv e-prints*, page arXiv:1207.0580.
- Huerta-Cepas, J. *et al.* (2019). eggNOG 5.0: a hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses. *Nucleic acids research*, **47**, D309–D314.
- Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In F. R. Bach and D. M. Blei, editors, *32nd International Conference on Machine Learning, ICML*, volume 37, pages 448–456. Proceedings of Machine Learning Research.
- Jensen, L. J. *et al.* (2007). eggNOG: automated construction and annotation of orthologous groups of genes. *Nucleic Acids Research*, **36**(S1), D250–D254.
- Johnson, J. M. and Khoshgoftaar, T. M. (2019). Survey on deep learning with class imbalance. *Journal of Big Data*, **6**(1), 27.
- Kanehisa, M. and Goto, S. (2000). KEGG: kyoto encyclopedia of genes and genomes. *Nucleic acids research*, **28**, 27–30.
- Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv e-prints*, page arXiv:1412.6980.
- Klambauer, G. *et al.* (2017). Self-normalizing neural networks. In I. Guyon *et al.*, editors, *Advances in Neural Information Processing Systems 30: NIPS 2017, Long Beach, CA, USA*, pages 971–980.
- Lan, Z. *et al.* (2019). ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. *arXiv e-prints*, page arXiv:1909.11942.
- Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-SNE. *Journal of machine learning research*, **9**(Nov), 2579–2605.
- Min, S. *et al.* (2017). Deep learning in bioinformatics. *Briefings in bioinformatics*, **18**, 851–869.
- Nayfach, S. *et al.* (2019). New insights from uncultivated genomes of the global human gut microbiome. *Nature*, **568**, 505–510.
- Pasolli, E. *et al.* (2019). Extensive unexplored human microbiome diversity revealed by over 150,000 genomes from metagenomes spanning age, geography, and lifestyle. *Cell*, **176**, 649–662.e20.
- Paszke, A. *et al.* (2019). PyTorch: An imperative style, high-performance deep learning library. In H. Wallach *et al.*, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.
- Rives, A. *et al.* (2019). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*.
- Seo, S. *et al.* (2018). DeepFam: deep learning based alignment-free method for protein family modeling and prediction. *Bioinformatics (Oxford, England)*, **34**, i254–i262.
- Sievers, F. *et al.* (2011). Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular systems biology*, **7**, 539.
- Strodthoff, N. *et al.* (2020). UDSMProt: Universal deep sequence models for protein classification. *Bioinformatics (Oxford, England)*.
- Suzek, B. E. *et al.* (2014). UniRef clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, **31**(6), 926–932.
- Wang, L. and Jiang, T. (1994). On the complexity of multiple sequence alignment. *Journal of computational biology : a journal of computational molecular cell biology*, **1**, 337–348.
- Weimann, A. *et al.* (2016). From genomes to phenotypes: Traitair, the microbial trait analyzer. *mSystems*, **1**(6).

Supplementary material for: DeepNOG: Fast and accurate protein orthologous group prediction

Roman Feldbauer Lukas Gosch Lukas Löfflinger
Patrick Hyden Thomas Rattei
Department of Microbiology and Ecosystem Science,
University of Vienna, Austria.

Arthur Flexer
Institute of Computational Perception,
Johannes Kepler University Linz, Austria.

April 2020

A Software development

The **deepnog** software package is developed with best-practices in mind. Source code, issue tracker, and possibilities to contribute are available at <https://github.com/univieCUBE/deepnog>. All resources are freely available under the BSD 3-clause open source license. The Python package can be installed from the public PyPI repository with `$ pip install deepnog` from the command line.

Online documentation is available at <https://deepnog.readthedocs.io/> and covers installation, quick-start examples, and a comprehensive API reference.

All Python code is compliant to PEP8 code style guidelines, heavily tested with **pytest** (at the time of writing, unit test coverage is $\approx 95\%$ of lines) with continuous integration (CI) enabled on all major platforms (Linux, MacOS, Windows) to ensure continual function and avoiding regressions. Additional code style and security checks are performed by LGTM.

The following DevOps resources are used for these purposes.

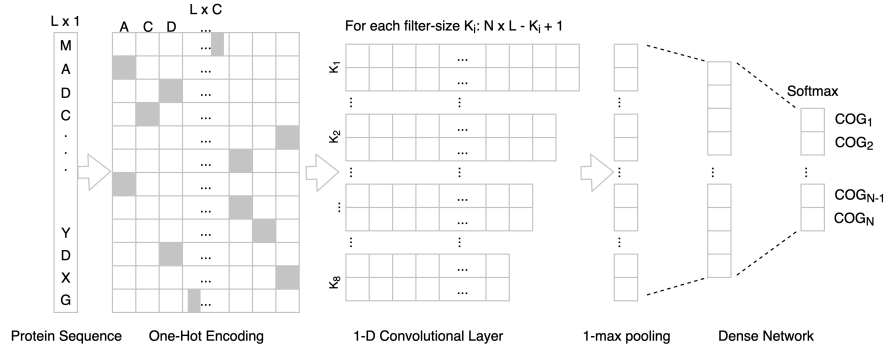
- CI for Linux/MacOS on <https://travis-ci.com/univieCUBE/deepnog/>
- CI for Windows on <https://ci.appveyor.com/project/VarIr/deepnog>
- PEP8 checks with **flake8** on Travis
- Additional checks on <https://lgtm.com/projects/g/univieCUBE/deepnog>

B Data availability

Trained network parameters are stored in files too large for GitHub and PyPI. They are, therefore, hosted on the server infrastructure of the authors' faculty,¹ which has served public databases for over ten years. The data is available from <https://fileshare.csb.univie.ac.at/deepnog/> and automatically fetched by the deepnog client, when required. It is cached locally, so that download is only necessary once.

C Additional material and methods

Figure S1: DeepFam architecture with onehot-encoding and additional fully-connected layer.



D Additional results

Table S1: Datasets derived from the COG database based on different restrictions.

Resulting COG datasets				
Name	Min. pop.	OGs	% sequences	length restriction
COG	-	4659	100	-
-	-	4655	98.7	1000
COG-100-2892	100	2892	93.5	1000
COG-500-1074	500	1074	67.5	1000

¹Public resources: <https://cube.univie.ac.at/resources>

Figure S2: Mean accuracy in dependence of number of classes. Note: DeepEncoding refers to DeepNOG in the main paper.

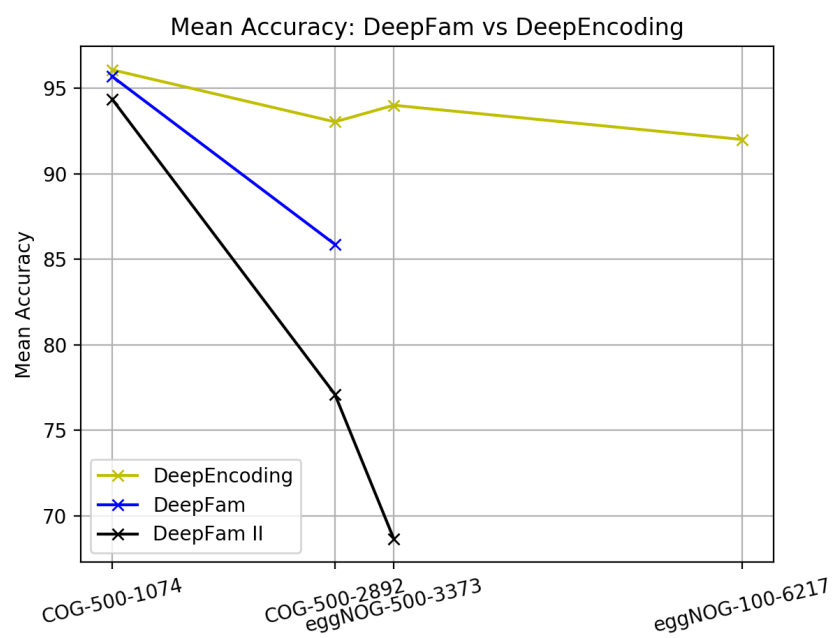
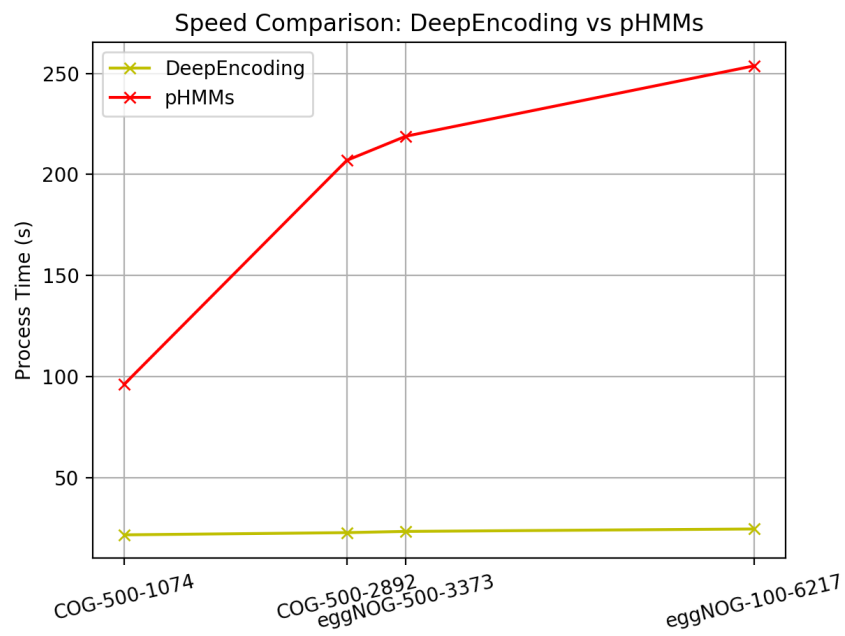


Figure S3: Speed in dependence of number of classes. Note: DeepEncoding refers to DeepNOG in the main paper.



7. Hubness in protein similarity databases

The main goal of this doctoral thesis is investigating the hubness phenomenon in large protein sequence similarity networks. To this end, hubness reduction methods are comprehensively evaluated in Chapter 2, defining the state-of-the-art in hubness reduction. Approximate hubness reduction methods for large-scale experiments are developed in Chapter 3 and implemented in a free open source software package in Chapter 4. Protein sequence vector representations described in Chapter 5 allow to apply approximate hubness reduction to protein similarity networks.

In this chapter, these threads come together. Hubness is investigated in the SIMAP2 database of protein sequence similarity. Various subnetworks as well as the full similarity network are considered. Hubness reduction is investigated in the context of multiple classification and clustering tasks, considering enzymatic activity, structural features, and orthology. Testing the central scientific Hypothesis H_1 concludes this work.

Dissemination The results of this work package have not been presented to the public at this point. The original findings below are published through this doctoral thesis. No further publications are currently planned.

Contributions Roman Feldbauer conceived the study, performed experiments, analyzed results, and wrote the report. Arthur Flexer and Thomas Rattei supervised the work.

7.1. Introduction and background

Proteins are essential building blocks of all cells, for example, providing enzymatic activities, participating in signaling pathways, or serving as structural components. The structure and function of each protein is determined by its primary structure, that is, its composition as an amino acid chain. These protein sequences are some of the most important study objects in bioinformatics. The entirety of all proteins of all extinct and extant species is referred to as *protein universe*. Technological and methodological advances during the last two decades allowed deeper and less biased sampling of the protein universe, and led to rapidly growing databases of protein sequences.

The protein universe is expanding, and a substantial fraction of sequences do not match any profile of known protein families [19]. The proportion of this so-called *dark matter* decreases slowly, but its absolute size increases due to database growth [219]. Especially metagenomic studies yield many sequences that can not be mapped to known proteins [220], hinting at undiscovered regions of the protein

universe. Computational protein design studies show, that known proteins occupy only a fraction of possible sequence and structure space [20]. Additional regions may be used in nature, but have not been sampled yet, or may not be used, but could be populated through biomolecular engineering.

Computational analysis pipelines need to keep up with the tremendous amounts of data, in order to generate knowledge from the accumulated information. The identification of groups of functionally related or interacting objects is a common early step in analysis pipelines [221]. Many inference tasks can be performed automatically, or semi-automatically with computational tools, especially when biological experiments are too expensive (e.g., metagenomics), difficult due to inaccessible systems (e.g., intracellular endosymbionts), or, dangerous.¹

Machine learning is widely applied in bioinformatics [235], [236]. The protein universe poses a range of difficulties for learning algorithms, including

1. large numbers of objects, resulting in high computational cost and memory requirements;
2. high-dimensional data spaces (as proposed, for example in [237]), which is detrimental to many machine learning algorithms, due to the *curse of dimensionality* [81]; and
3. highly variable primary sequence lengths (ranging from 2 to 35 213 amino acids in Swiss-Prot²).

Additional challenges include non-trivial relations between values at each position in the sequence (biochemical amino acid similarities), high-level features, like secondary structures and protein domains, varying evolutionary rates, or spurious similarities between sequence pairs. One particular aspect of problem 2 is *hubness*, that is the emergence of *hubs* and *antihubs*: Hubs are objects that are particularly often among the nearest neighbors of other objects. Conversely, antihubs are rarely or never nearest neighbors to other objects. Hubness has previously been described as a general phenomenon in high-dimensional spaces [87], that is detrimental to learning tasks, such as clustering, classification, visualization, and others [92], [114], [123]. Nearest neighbor relations in high-hubness regimes are prone to semantic incorrectness: Hubs propagate their encoded information too widely in corresponding distances spaces, while information carried by antihubs is essentially lost. Consequently, these distance spaces do not reflect semantic meaning of the data well, which can lead to suboptimal classification or clustering. Several methods for hubness reduction have been proposed, that were shown to improve performance measures in learning tasks with data sets showing high hubness. Most notable examples include *mutual proximity* and *local scaling* [92], as well as *DisSim^{Local}* [103]. Preliminary results show high hubness in protein sequence databases, underlining the importance of investigating the influence of the curse of dimensionality on protein classification and clustering. To the best of the author’s knowledge, this dissertation project will be the first to examine this connection.

¹For example, lab work on SARS-CoV-2 requires at least BSL-3 facilities. Virus identification, characterization, or drug development rely extensively on computational methods. [222]–[234]

²Swiss-Prot queries <https://www.uniprot.org/uniprot/?query=length%3A%5B0+T0+5%5D+AND+reviewed%3Ayes&sort=length&desc=no>, and [https://www.uniprot.org/uniprot/?query=length:\[30000%20T0%201000000\]&fil=reviewed%3Ayes&sort=score](https://www.uniprot.org/uniprot/?query=length:[30000%20T0%201000000]&fil=reviewed%3Ayes&sort=score), retrieved on 2020/04/30.

7.2. Materials and methods

This section describes the methods employed for hubness estimation, hubness reduction, classification and evaluation metrics. It also introduces the data sets used in the experiments.

7.2.1. Hubness estimation

Several methods of hubness estimation exist. Evaluations here focus on two criteria. First, the skewness of the k -occurrence histogram [87] (see Equation 1.3, p. 7) is considered in all experiments. Second, experiments conducted after the initial publication suggesting the Robin Hood index for hubness estimation (see Chapter 3), also use this measure.

7.2.2. Hubness reduction

Hubness reduction methods are selected based on the results of Chapter 2, which identify mutual proximity (MP), local scaling (LS), and DisSim^{Local} (DSL) as state-of-the-art algorithms. Experiments are performed on similarity data and vector data using MP and LS. These two methods are applicable to both data types, whereas DSL is tailored to vectors only. Hubness reduction in similarity data is performed with MP and LS as introduced in Chapter 2, Equations 2 and 5, p. 22, and implemented in the Hub-Toolbox [172]. MP refers to mutual proximity with the empirical distance distributions, unless specified otherwise. Some experiments employ Mutual Proximity with distances modeled as Gamma distributions. Approximate hubness reduction is employed in large vector-based data sets using the methods described in Chapter 3, paper section III, p. 49–52, and is implemented in `scikit-hubness` (see Chapter 4).

7.2.3. Classification

Classification is performed with supervised k -nearest neighbors (k NN). The k most similar (or least distant) objects with respect to some (dis-)similarity measure are retrieved from the training set for each test object. In single-label classification each test object is assigned the most frequent label among its k -nearest neighbors. Ties are broken by distance. In the multi-label case, each class is considered individually, and assigned to a test object, if the majority of its k NN have the corresponding label.

Classification performance is assessed with the following measures. In single-label experiments performance is measured with *accuracy*, i.e., the fraction of all correctly labeled instances over all examined instances.

In multi-label experiments the F-measure is applied, which is calculated as the harmonic mean of precision and recall. For the micro-average true positives, false negatives, and false positives are counted globally, and the F-measure is computed from the derived precision and recall. In case of class imbalance, densely populated classes dominate the value. The macro-average reports the arithmetic mean over the F-measures for all individual classes, thus, putting equal emphasis on all classes independent from membership.

For some experiments R-Precision is considered, additionally. It is a measure typically used in information retrieval to assess retrieval systems globally: For each query object indexed by i , let R_i the number of relevant items in the database, for example, all sequences with the same label. Further, let r_i the number of correct predictions among the R_i nearest neighbors. Then $\text{R-Prec}_i = \frac{r_i}{R_i}$. Again, macro-R reports the arithmetic mean of R-Prec, whereas micro-R computes an average weighted by the number of relevant objects per query.

7.2.4. Clustering

Unsupervised learning is performed with Markov clustering (MCL, [238]), an established method frequently used in biology [62], [239]–[242].

Cluster quality is assessed by measuring semantic similarity of proteins within each cluster, employing experimentally validated Gene Ontology (GO, [243]) terms of molecular function (MF), and the G-SESAME [244] index with the best match average (BMA) strategy as implemented in the fastSemSim library [245], 0.9.4. Only clusters containing at least two sequences with available GO MF terms are considered. Singleton clusters and clusters without GO annotations are discarded.

7.2.5. Data set EC: Enzymes

The Enzyme Commission provides a nomenclature for enzymatic reactions. EC numbers comprise four numerical blocks, that describe a hierarchy of reaction classes. Identical EC numbers in different proteins may result from homology or convergent evolution [246]. Sequences with exactly one EC label are retrieved from UniProtKB/SwissProt [247] and clustered to UniRef50 [58]³. A SIMAP2 subnetwork is extracted for the resulting 44 218 sequences.

7.2.6. Data set PDB: Protein Data Bank

The Protein Data Bank [248] is the comprehensive source of experimentally solved 3D protein structures. Sequences are obtained from the PDB, and enriched with GO molecular function annotations⁴. The corresponding SIMAP2 subnetwork comprises 69 496 vertices and 3 994 551 edges. For comparison, an additional SIMAP2 subnetwork is extracted for 69 496 randomly selected sequences, yielding 155 403 edges.

7.2.7. Data set SCOPe: Protein structure

The Structural Classification of Proteins databases (SCOP 1 [249] and SCOP extended, SCOPe 2, [250]) classify related protein structures obtained from the Protein Data Bank on three hierarchical levels. Sequences with one or multiple labels are retrieved from the fully manually curated SCOP 1 database⁵ and clustered to UniRef100. The resulting SIMAP2 subnetwork contains 23 608 sequences.

³EC data retrieval: 2017/01

⁴PDB and GO data retrieval: 2016/05

⁵SCOP data retrieval: 2017/02

7.2.8. Data set eggNOG: Orthologous groups

The evolutionary genealogy of genes: non-supervised orthologous groups database (eggNOG) provides evolutionary classification of protein families. Building upon the expert-curated COG database [60] and an unsupervised clustering algorithm, eggNOG assigns proteins to orthologous groups on various taxonomic levels. Two data sets are created stemming from eggNOG: The first data set consists of all sequences from eggNOG 1 [64] with one label on the root level.⁶ Sequences are clustered to UniRef100, and cross-referenced to SIMAP2. The obtained similarity subnetwork consists of 753 174 sequences.

The second data set consists of sequences from eggNOG 5 [174] with one or multiple NOG labels on the bacteria level. The sequences are further clustered to different levels of sequence identify for individual experiments. The full data set comprises 14 400 164 sequences. Additional data sets are created by clustering to UniClust90 (6 818 682 sequences), UniClust50 (2 462 654 sequences), and UniClust30 [59] (976 124 sequences)⁷.

7.3. Results and discussion

The influence of hubness on the protein universe is investigated in several subnetworks of SIMAP2. The following sections describe experiments for hubness estimation as well as hubness reduction, and investigate its influence on classification and clustering performance.

7.3.1. Hubness estimation

Hubness is a phenomenon of intrinsically high-dimensional data. Since proteins are not naturally represented by vectors, it is not obvious that hubness could affect them. Previous studies describing proteins in terms of vectors embedded sequences in 60-dimensional space [237], or embedded each amino acid in vector spaces of various dimensionality (e.g., 64 dimensions in [251], or 100 dimensions in [252]). These results indicate that useful representations of protein sequences might require high dimensionality. However, there is no notion of what the “true” dimensionality of protein sequences could be. Instead of deriving first a concept for determining such dimensionality, hubness is directly measured in the protein sequences similarity networks. Figure 7.1 provides hubness measurements in the complete SIMAP2 database. Here, the standard measure for hubness estimation, k -occurrence skewness S^k , is employed, and presented for various values of k . As hubness is closely tied to the concept of nearest neighbor, typically low values of k are considered, e.g., 1, 5, 10, or 20.⁸ Two different protein similarity scores are considered: (i) the Smith-Waterman ratio (SWR, as defined in Section 5.2.1), and (ii) the percentage of alignment positions with positive substitution scores (SIM). While SIM is not used for other experiments, it serves here as control, whether hubness values are specific for certain similarity measures. Indeed, both measures indicate high hubness in

⁶eggNOG 1 data retrieval: 2017/03

⁷eggNOG 5 and UniClust data retrieval: 2020/03

⁸Note, that skewness converges to 0 when $k \rightarrow N$, where N is the number of objects in the database, because then all objects become “nearest” neighbor to each object.

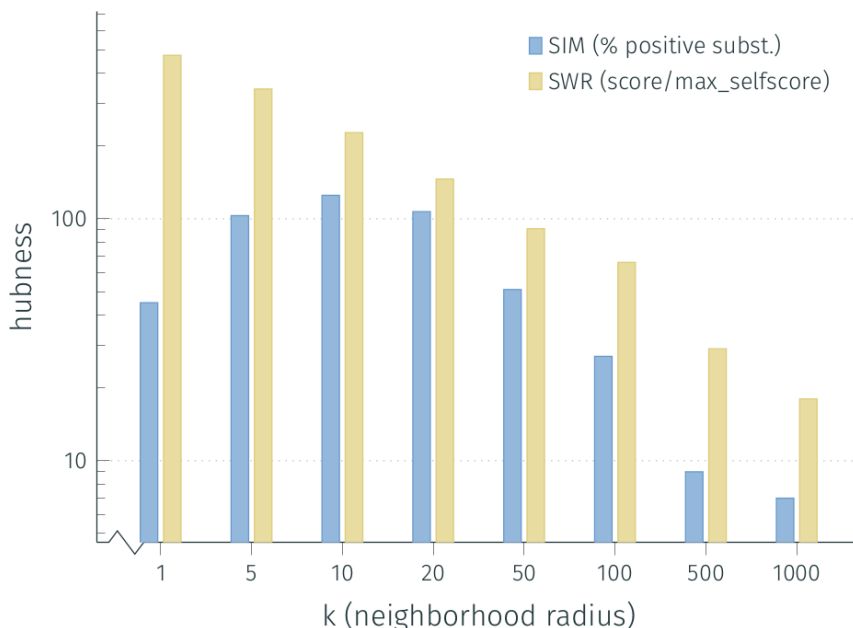


Figure 7.1.: SIMAP2 hubness (skewness of the k -occurrence distribution)

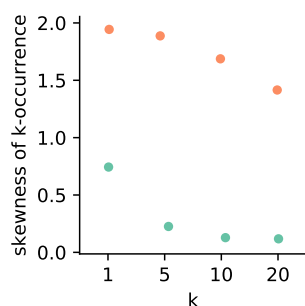
SIMAP2. For low values of k , skewness values between 100 and 500 are observed. These are the highest values observed in any dataset considered in this work (see also Chapter 2, Table 3 p. 30–31, where $\max S^{k=10} = 15.5$). Given these extremely high numbers, further experiments with hubness reduction are performed below.

7.3.2. Enzymes

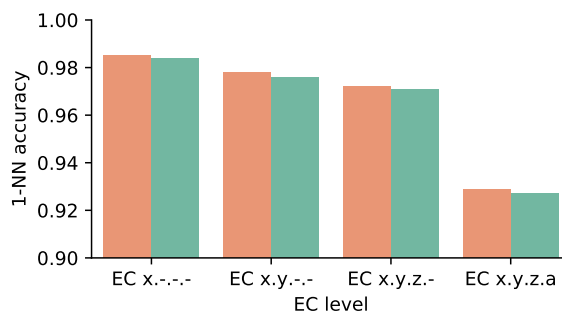
Several classification schemes for proteins are considered here. First, proteins with enzymatic activity are investigated. Nearest neighbor classification with respect to sequence similarity can classify enzymatic activity in the EC scheme, when related proteins are available in the database. Non-homologous sequences can code for reactions of the same EC class, which will not be detected. The goal here is not to improve upon state-of-the-art tools as presented, for example, in [253]. Instead, EC class label prediction with nearest neighbors is used as a proxy for the semantics of the similarity space. The same applies to the other data sets described in the sections below. Figure 7.2a and 7.2b show results of hubness estimation and nearest neighbor classification, both before and after hubness reduction with mutual proximity. The EC dataset shows moderate levels of hubness ($S^{k=1} \approx 2$), which is markedly reduced by mutual proximity. 1-NN classification achieves high accuracy on the first three levels of the EC hierarchy. While there is little room for improvements, mutual proximity rather diminishes performances, though by a small margin. Hubness reduction does not improve this task.

7.3.3. SCOPe

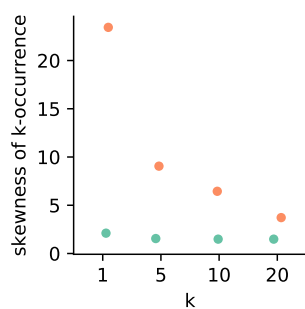
Structural properties of proteins are considered as a second proxy for semantic properties of protein sequence similarity spaces. They are of high interest, because protein structure determines function. Hubness estimation is depicted in Figure 7.2c. The SCOPe data set shows strong hubness, especially considering the 1-occurrence



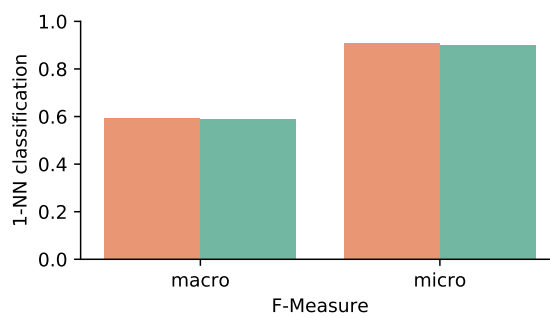
(a) Hubness in EC dataset



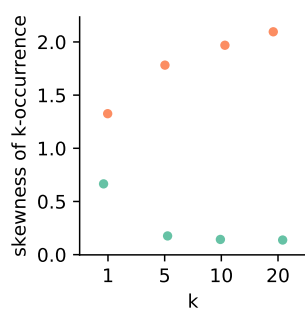
(b) EC classification



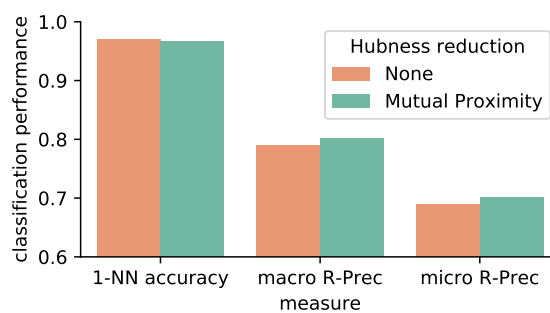
(c) Hubness in SCOPe dataset



(d) SCOPe classification



(e) Hubness in EggNOG 1 data



(f) EggNOG 1 classification

Figure 7.2.: Hubness reduction and classification performance in various data sets.

distribution with $S^{k=1} \approx 24$. Mutual proximity reduces hubness dramatically. One possible explanation for the marked differences in hubness between the EC and the SCOPE data sets stems from certain biases in the Protein Data Bank, which often contains clusters of highly similar structures. When structures are derived from point mutations of a common ancestor, this ancestor can be trivially identified as a hub object for its mutations, while the global structure of the similarity space will be hardly affected. Proteins can comprise several domains, and can have multiple SCOPE labels. Therefore, macro- and micro-averages of the F-measure are considered (Figure 7.2d). Micro-averages are substantially larger than macro-averages, hinting at large classes dominating the overall score in nearest neighbor classification. Again, no improvements in classification performance due to hubness reduction are observed with F-measures slightly below baseline.

7.3.4. EggNOG 1

The third SIMAP2 subnetwork comprises sequence from the eggNOG 1 database. The classification task here is identifying orthologous sequences. With over 700 000 sequences, this data set is over an order of magnitude larger than the previous two data sets. This size entails significant computational and memory requirements in hubness reduction. Larger data sets profit tremendously from approximate hubness reduction methods in terms of hardware requirements. Moderate levels of hubness comparable to the EC data set are observed (Figure 7.2e). Nearest neighbor classification performance is high (Figure 7.2f). Also in case of eggNOG 1, hubness reduction with mutual proximity does not improve 1-NN classification. In order to investigate not only highly local properties of the similarity space (only one neighbor), R-Precision is measured here additionally. Assessing global properties is relevant for methods that incorporate larger parts of the similarity network, for example, clustering methods, such as Markov clustering, spectral clustering [254], [255] or affinity propagation [256]. In both macro- and micro-average R-Precision, slightly increased values are observed with hubness reduction.

7.3.5. EggNOG 5

The fourth SIMAP2 subnetwork consists of bacteria domain sequences from the eggNOG 5 database. The task is identical as in the section above. Again, nearest neighbor classification is employed. In addition to using the SWR similarity space, *frenetiq* vector spaces (see Chapter 5) are investigated. This allows for approximate hubness reduction.

Figure 7.3 reports hubness estimation results in terms of the Robin Hood index, as proposed in Chapter 3, paper section IIIA, p. 50. Moderate hubness is observed in all configurations before hubness reduction. Robin Hood indices $\mathcal{H}^k \approx 0.4$ across differently clustered data sets and data spaces, with slightly decreased values in UniClust90. The learned vector space appears to have inherited hubness properties from the SIMAP2 SWR space it was trained on.

In general, the Robin Hood hubness values contrast with the hubness estimates relying on the k -occurrence skewness as depicted in Figure 7.1. For the subnetworks here, skewness $S^{k=1}$ values are found between 2–12. These observations suggest that the extreme hubness values measured in the full SIMAP2 network stem from the

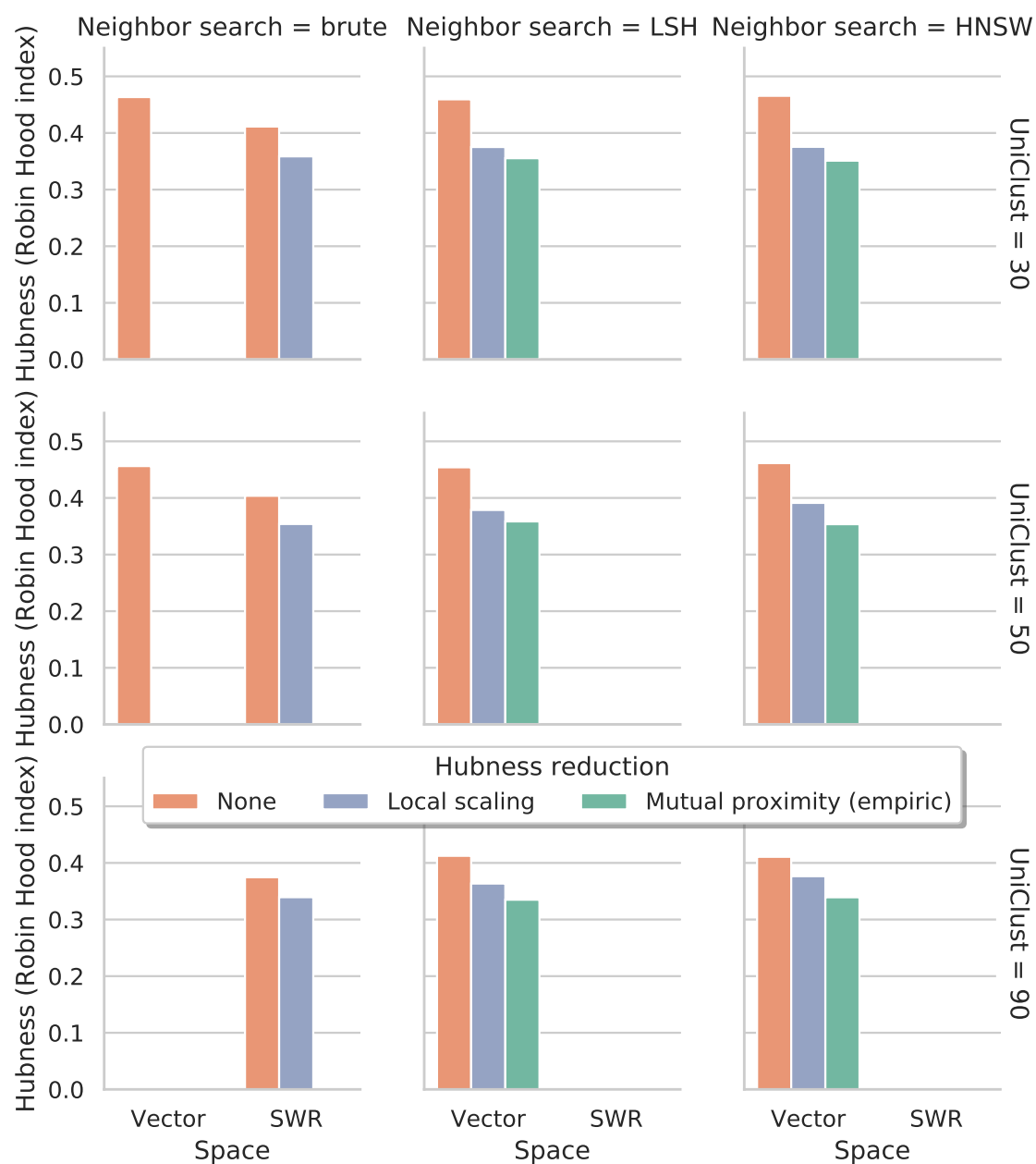


Figure 7.3.: EggNOG 5 hubness estimation in learned vector space (*frenetiq*, see Chapter 5), and in SIMAP2 SWR similarity space with with exact and approximate neighbor search, with or without hubness reduction.

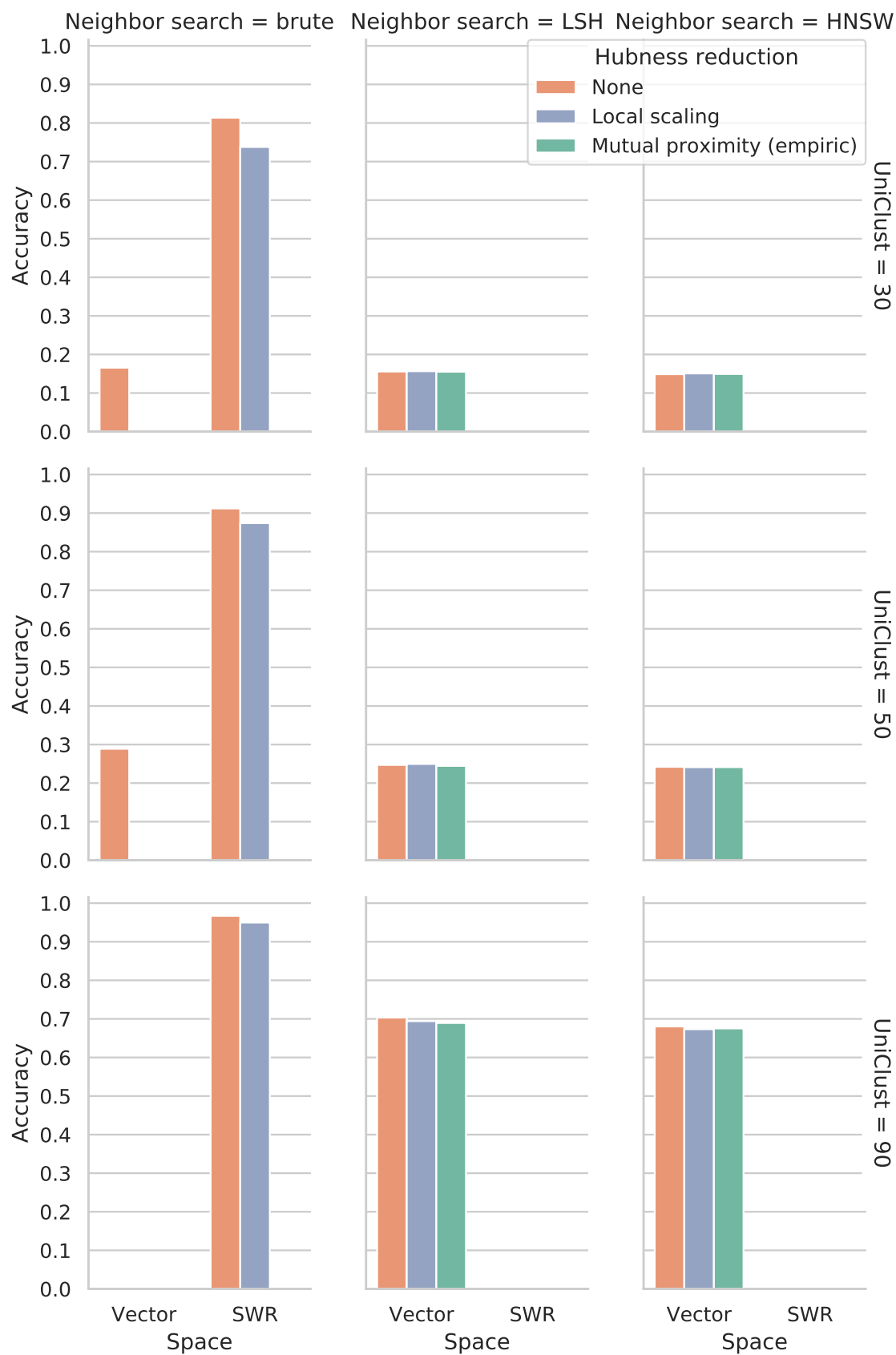


Figure 7.4.: EggNOG 5 classification in learned vector space (*freneti*, see Chapter 5), and in SIMAP2 SWR similarity space with with exact and approximate neighbor search, with or without hubness reduction.

data set size rather than particular properties of the data space. The findings are in alignment with the results presented in Chapter 3, paper section III—A, p. 50, where the influence of data set size to hubness is investigated: Comparison with Figure 1b suggests hubness in SIMAP2 to compare to 32–64 dimensional Gaussian data, that is, only moderately high-dimensional.

Both examined hubness reduction methods, local scaling, and mutual proximity, reduce hubness in terms of the Robin Hood index.

Similar to results for the eggNOG 1 data set, classification performance is not improved by hubness reduction. While accuracy in *frenetiq* vector space is hardly affected, it is reduced in SWR similarity space, when applying local scaling. These results discourage the use of hubness reduction for protein orthology detection.

Approximate search methods retrieve neighbors of similar quality compared to the brute force baseline. However, the required vector embeddings yield at least 20% lower accuracy (UniClust90). Results deteriorate in regimes clustered to lower sequence identity, indicating insufficient sensitivity of the current state of the *frenetiq* approach.

Neighbor search with LSH and HNSW is not directly applicable to similarity spaces, such as SWR. Hubness reduction in combination with brute force neighbor search exceeds the computational budget⁹. The same applies to neighbor search in UniClust90 vector space.

7.3.6. PDB clustering

Two additional data sets are employed to investigate the influence of hubness reduction on clustering protein sequences. A SIMAP2 subnetwork containing sequences from the PDB is clustered with Markov clustering, (i) using the SWR similarity measure (no hubness reduction), and (ii) using secondary distances obtained by applying mutual proximity to the SWR distance matrix (hubness reduction). Here, two MP variants, empiric and gamma, are considered that model distances with observed empirical distributions, or Gamma distributions [92], respectively. Sequence alignment scores are often modeled by an extreme value distribution, type I (Gumbel distribution, [257]), and other distributions have been proposed (for example, in [258], [259]). Some evidence that Gamma distributions fit well to protein sequence alignment scores has been brought forth previously [260]. Hubness reduction with an experimental MP Gumbel variant did not improve results (data not shown).

Cluster coherence ranges from 0 to 1 in experiments with and without hubness reduction (weighted and unweighted averages, see Figure 7.5, top row). Significant differences are observed for mutual proximity with empiric distance distributions compared to the SWR baseline. The differences in coherence distributions when using weighted or unweighted averages hints at improvements mainly in smaller clusters. Experiments with mutual proximity using Gamma distributions do not yield evidence for improved clustering.

Results are compared to a random PDB-sized data set (see Figure 7.5, bottom row). In case of the unweighted averages, both mutual proximity variants improve cluster coherence significantly. However, the weighted averages reveal that there are large fully incoherent clusters. One possible explanation might be found in the extremely low network density. In contrast to the highly interconnected PDB

⁹CPU budget: 30d on Intel Xeon Gold 5115, 40 cores

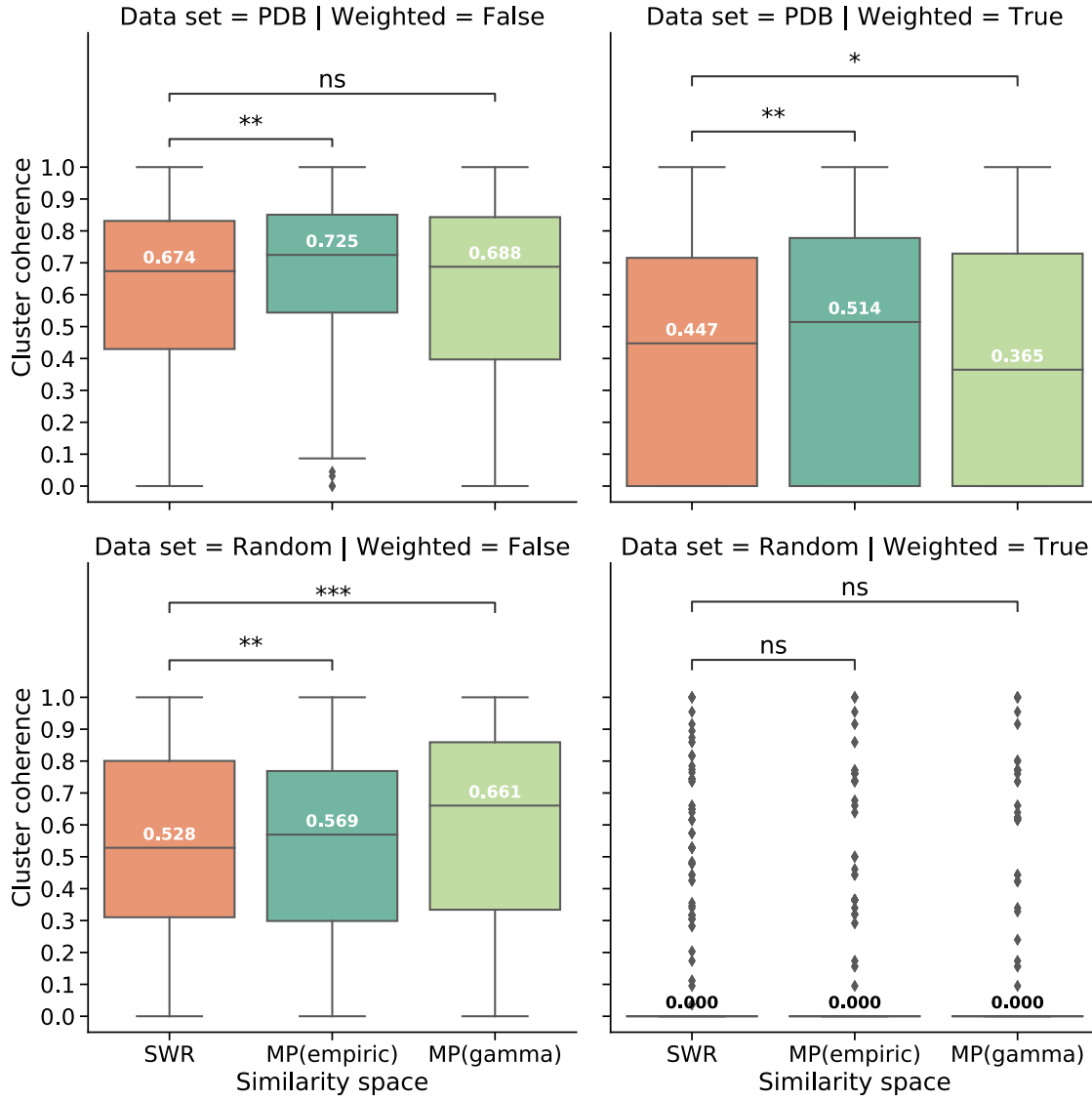


Figure 7.5.: Clustering SIMAP2 subnetworks with Markov clustering. Cluster coherence measured with G-SESAME semantic similarity index of experimental GO molecular function labels. In rows: PDB data set and random data set. In columns: Weighting cluster coherence average by individual cluster sizes. Significance levels: ns ($p \geq 0.5$), * ($p < 0.05$), ** ($p < 0.01$), *** ($p < 0.001$), two-sided Mann-Whitney-Wilcoxon test with Bonferroni correction.

similarity subnetwork, sequences in the random SIMAP2 subnetwork have lower degree, that is, few alignment hits. Indeed, examination of one large cluster revealed its lower density compared to small clusters. Presumably, slightly lower similarities would have resulted in all singleton clusters.

Overall, consistent improvements are found only for mutual proximity using empiric distance distributions. This approach comes at a high cost: Sequence similarity matrix, hubness reduction, and clustering each necessitate $\mathcal{O}(n^2)$ memory (with n the number of sequences), and mutual proximity (empiric) requires $\mathcal{O}(n^3)$ time. The high complexity limits the approach to small or medium-sized data sets. Clustering the complete SIMAP2 network, or even larger data sets such as UniProtKB with millions of sequences is impossible with reasonable resources.

7.4. Conclusion and outlook

Initial experiments showed extremely high hubness in the SIMAP2 database of all-against-all sequence alignments of nearly 30 M proteins, that is, an order of magnitude higher than any other data set investigated previously. The investigation of smaller subnetworks as well as the introduction of the Robin Hood index for hubness estimation adjust this perspective. High k -occurrence skewness values appear to be artifacts stemming from the large data set size. Measuring hubness with the Robin Hood index yields values similar to 32–64 dimensional random Gaussian data.

Considering this perspective, the rather small effects of hubness reduction on supervised learning (no observed improvements in any of the classification tasks) and unsupervised learning (small improvements in retrieval precision and cluster coherence) are not surprising.

Hubness has been identified as a detrimental phenomenon for various tasks in several research domains. Little evidence was found that this aspect of the curse of dimensionality impairs learning in protein sequence similarity networks to an extent relevant for downstream analyses. The cost-benefit ratio of expensive hubness reduction in the protein sequence universe is, therefore, not particularly promising. The null hypothesis of no learning improvements due to hubness reduction could not be rejected.

8. Conclusion and outlook

This thesis investigated the influence of hubness on learning in protein sequence similarity spaces.

Chapter 2 reported the results of a comprehensive empirical evaluation of available hubness reduction methods with regard to their ability to reduce hubness and improve classification performance in a large variety of real-world benchmark data sets from multiple application domains. Hypothesis H_2 was rejected: There are significant performance differences between different algorithms. Both improvement and deterioration compared to baseline methods were observed, and three algorithms were identified as state-of-the-art in hubness reduction: mutual proximity, local scaling, and $\text{DisSim}^{\text{Local}}$. These methods were further considered for experiments.

Chapter 3 described the conception and development of approximate hubness reduction methods for vector data based on the best algorithms identified before. The filter-and-refine approach combining approximate neighbor search and hubness reduction features subquadratic complexity. This allows to apply hubness reduction to large data sets, confirming Hypothesis H_3 . In addition, a new measure for hubness was proposed. The Robin Hood index has preferable properties compared to the k -skewness, the latter growing with sample sizes, whereas the former allows to reliably estimate hubness in large data sets from small samples. A case study on accelerometer time series revealed improved transport mode detection by hubness reduction.

Chapter 4 introduced `scikit-hubness`, a free open source software package implementing hubness analysis and hubness reduction algorithms (both exact and approximate). The Python program builds upon the SciPy stack and integrates into `scikit-learn` pipelines. Software development follows best practices, such as PEP8-compliance, a large coverage test suite, and comprehensive documentation. The package is available on the PyPI repository and can be installed on Linux, MacOS, and Windows. It allows machine learning researchers and practitioners to readily apply hubness analysis, and enabled large-scale hubness-related experiments in this thesis.

Large-scale hubness reduction requires vectors representations of the investigated objects. Approaches to protein sequence vector embeddings were, therefore, assessed in Chapter 5. A doublet convolutional network was trained to embed protein sequences in vector space, so that Euclidean distances reflect similarity scores obtained from exact local sequence alignments. Assessing Hypothesis H_4 , it was found that the obtained embeddings preserve a large fraction of nearest neighbor relations of the protein similarity network SIMAP2. Experiments on exact and approximate neighbor search in the learned space showed promising results of extremely fast protein similarity search. Retrieval sensitivity remained suboptimal, however, and the approach might be improved by a number of proposed enhancements. Experiments on multitask learning involving supervised classification criteria motivated an additional work package concerning orthology inference.

Chapter 6 reported on fast and accurate protein orthologous group inference based on deep learning. While orthology classification tasks did not improve neighborhood relations in the embedding spaces discussed above, deep networks were found to infer orthology of protein sequences faster than competing algorithms on equal levels of prediction accuracy, confirming Hypothesis H_5 . The proposed network was implemented in the **deepnog** software package, applying the same principles of software development as outlined above for **scikit-hubness**. The tool is available for all major operating systems. **deepnog** allows biologists to rapidly infer eggNOG 5 orthologous groups for protein sequences they experiment with.

Chapter 7 investigated the effect of hubness on learning in the protein sequence universe. Building upon the method evaluation, development, and implementation described in Chapters 2–5, hubness reduction was applied to multiple protein sequence networks in both similarity and vector spaces. Hubness was measured with the k -skewness and Robin Hood index. Classification and clustering performance with and without hubness reduction was compared. All investigated classification tasks were found to be impaired by hubness reduction rather than improved. Mild improvements were reported for clustering tasks with the highly expensive mutual proximity variant using empirical distance distributions. The obtained evidence does not support the central Hypothesis H_1 of improved learning in the protein sequence universe due to hubness reduction. The protein space dimensionality might not be as high as initially assumed. While hubness appears extremely large when measured as k -skewness, moderate values were obtained for the Robin Hood index of the k -occurrence distribution.

The results presented here advance the field of hubness analysis. Machine learning researchers and practitioners are provided with guidelines on which methods to choose for analyzing their high-dimensional data, and with software packages implementing these methods. Hubness reduction is possible in large data set for the first time due to the development of approximate methods. The **scikit-hubness** package provides these hubness-related functionalities as well as general approximate neighbor search, making it widely applicable to large data sets in arbitrary domains.

Assessing the impact of recently published research software tools is nontrivial, but some proxy measures are available: The package is downloaded from the PyPI repository approximately 100 times per month.¹ The project’s GitHub page reports several forks and twenty stars from users of various backgrounds, including mathematics, public health, and finance in addition to the expected fields, such as music information retrieval, natural language processing, and general data science.² Considering one particular downstream project, **sesame-street** analyzes “BERTology” models³, and employs hubness estimation and reduction for the task. It will be interesting to see, whether hubness analysis might be an integral part of deep learning language model assessment in the future.

During the course of this work, hubness-related research shifted its scope towards approximate neighbor search. Using approximate algorithms to enable hubness re-

¹**scikit-hubness** statistics from <https://pypistats.org/packages/scikit-hubness>, retrieved 2020/05/30. Note that raw download counts are influenced by mirrors, continuous integration (CI) platforms, and others. The website excludes known mirror traffic.

²Impact: <https://github.com/VarIr/scikit-hubness/>, retrieved 2020/05/30.

³**sesame-street**: <https://github.com/ChenghaoMou/sesame-street>, retrieved 2020/05/30

duction in large data sets is one side of the story. Approaching from a different angle, the results of Chapter 3 can be read as: not only exact algorithms but also approximate neighbor search algorithms, such as LSH and HNSW, are impaired by hubness. This perspective has been investigated in detail for the NN-descent algorithm [119], [120]. Future work could investigate this connection in more detail for additional algorithms and in various contexts, including image search, information retrieval [261], or natural language processing. For instance, the Reformer model replaces dot-product attention employed in Transformer models (BERT) with LSH attention, improving time and memory efficiency, while retaining state-of-the-art learning performance. LSH attention typically operates in high-dimensional spaces, e.g. 256–1024 [210]. It is intriguing to investigate, whether incorporating hubness reduction into LSH attention can further improve classification in natural language processing tasks, advancing the state-of-the-art.

Considering hubness in the protein sequence universe, initial expectations were not met. Given the ostensibly high hubness values obtained in preliminary experiments, and the success of hubness reduction in other application domains, effects on protein classification and clustering were hypothesized. However, closer investigation revealed that protein sequence spaces are not particularly prone to hubness. At the same time, little influence of hubness reduction on classification and clustering performance was observed. It is nevertheless important to report negative results, as this process contributes to a less biased view on nature that can emerge from publishing scientific studies only in case of positive results. In addition, negative results help in planning future research projects. Regarding effects of hubness on learning in the protein universe, the gathered evidence discourages further investigation in this direction.

Protein sequence embeddings, however, could prove a valuable resource for computational biology. Future work will show, whether retrieval performance can be further improved. Sensitivity of the approach will need to be increased for successful application in similarity search. Protein representations fulfilling these requirements might then be combined with approximate neighbor search algorithms in order to remove the bottleneck of similarity search computational cost in ever-growing databases.

Spin-off results of this thesis will provide useful resources for biologists. The DeepNOG architecture enables quick and reliable protein orthologous group inference. It is already employed in the **phenotrex** pipeline for microbial phenotypic trait prediction, which gives in-depth insights into the functional capacities of fully sequenced genomes as well as metagenomic assembled genomes (see Appendix A.2.3). Additional projects involving the method are already planned, including investigations on viral evolution.

Contributions to scientific software packages improved their functionality and quality (see Appendix A). Projects like SciPy and scikit-learn rely on the scientific community to participate in development. In turn they drive research projects in all domains of natural sciences as well as quantitative social sciences or digital humanities, with hundreds of thousands dependent software packages.

A. Contribution to scientific software

All of the work presented in the main body of this dissertation builds upon prior research results, as well as scientific software. The author believes that it is imperative to aid and promote free open source software in order to facilitate rapid scientific progress.

To this end, the author made an effort to not only use available software tools, but also contribute to their ongoing development. Unfortunately, little funds are typically available for scientific software engineering. For SciPy 1.0, development cost is estimated in excess of ten million USD, virtually none of which was covered by third-party funds specific for SciPy development (for details, see the paper below).

Contributions, therefore, often conform to the following pattern:

1. A user of the upstream software package, for example, a graduate student, requires an additional feature, or exposes a bug.
2. The user implements the required feature or creates a bugfix/workaround in their custom scripts/packages.
3. The new feature or bugfix is requested to be merged into the upstream package.

A.1. Published work

The author's contribution to the `sparse` package of SciPy is a typical example of the pattern described above. The SIMAP2 database of protein sequence alignments can be represented as a sparse matrix of similarity scores. While studying SciPy source code for efficient handling of sparse matrices, such as SIMAP2, the author observed certain limitations in access patterns to compressed sparse matrices. Dense matrices allow for $\mathcal{O}(1)$ access of single elements, because the memory location can be directly computed from row and columns indices. In compressed sparse matrices, this location can not be computed in constant time. SciPy before version 1.0 employed a linear scan through a row to extract single elements from *compressed sparse row* matrices, resulting in $\mathcal{O}(n)$ access, where n is the number of nonzero elements. However, when column indices are sorted per row, binary search may be exploited to find the required element.¹ Access is then in $\mathcal{O}(\log n)$, which improves speed particularly for large matrices. The author contributed this functionality to SciPy.

¹The same concept applies analogously to compressed sparse column matrices and block sparse row matrices.

OPEN

SciPy 1.0: fundamental algorithms for scientific computing in Python

Pauli Virtanen¹, Ralf Gommers^{2*}, Travis E. Oliphant^{2,3,4,5,6}, Matt Haberland^{7,8*}, Tyler Reddy^{9*}, David Cournapeau¹⁰, Evgeni Burovski¹¹, Pearu Peterson^{12,13}, Warren Weckesser¹⁴, Jonathan Bright¹⁵, Stéfan J. van der Walt¹⁶, Matthew Brett¹⁶, Joshua Wilson¹⁷, K. Jarrod Millman^{14,18}, Nikolay Mayorov¹⁹, Andrew R. J. Nelson²⁰, Eric Jones⁵, Robert Kern⁵, Eric Larson²¹, C J Carey²², İlhan Polat²³, Yu Feng²⁴, Eric W. Moore²⁵, Jake VanderPlas²⁶, Denis Laxalde²⁷, Josef Perktold²⁸, Robert Cimrman²⁹, Ian Henriksen^{6,30,31}, E. A. Quintero³², Charles R. Harris^{33,34}, Anne M. Archibald³⁵, Antônio H. Ribeiro³⁶, Fabian Pedregosa³⁷, Paul van Mulbregt³⁸ and SciPy 1.0 Contributors³⁹

SciPy is an open-source scientific computing library for the Python programming language. Since its initial release in 2001, SciPy has become a de facto standard for leveraging scientific algorithms in Python, with over 600 unique code contributors, thousands of dependent packages, over 100,000 dependent repositories and millions of downloads per year. In this work, we provide an overview of the capabilities and development practices of SciPy 1.0 and highlight some recent technical developments.

SciPy is a library of numerical routines for the Python programming language that provides fundamental building blocks for modeling and solving scientific problems. SciPy includes algorithms for optimization, integration, interpolation, eigenvalue problems, algebraic equations, differential equations and many other classes of problems; it also provides specialized data structures, such as sparse matrices and *k*-dimensional trees. SciPy is built on top of NumPy^{1,2}, which provides array data structures and related fast numerical routines, and SciPy is itself the foundation upon which higher level scientific libraries, including scikit-learn³ and scikit-image⁴, are built. Scientists, engineers and others around the world rely on SciPy. For example, published scripts^{5,6} used in the analysis of gravitational waves^{7,8} import several subpackages of SciPy, and the M87 black hole imaging project cites SciPy⁹.

Recently, SciPy released version 1.0, a milestone that traditionally signals a library's API (application programming interface) being mature enough to be trusted in production pipelines. This version numbering convention, however, belies the history of a project that

has become the standard others follow and has seen extensive adoption in research and industry.

SciPy's arrival at this point is surprising and somewhat anomalous. When started in 2001, the library had little funding and was written mainly by graduate students—many of them without a computer science education and often without the blessing of their advisors. To even imagine that a small group of 'rogue' student programmers could upend the already well-established ecosystem of research software—backed by millions in funding and many hundreds of highly qualified engineers^{10–12}—was preposterous.

Yet the philosophical motivations behind a fully open tool stack, combined with an excited, friendly community with a singular focus, have proven auspicious in the long run. They led not only to the library described in this paper, but also to an entire ecosystem of related packages (<https://wiki.python.org/moin/NumericAndScientific>) and a variety of social activities centered around them (<https://wiki.python.org/moin/PythonConferences>). The packages in the SciPy ecosystem share high standards of

¹University of Jyväskylä, Jyväskylä, Finland. ²Quansight LLC, Austin, TX, USA. ³Ultrasound Imaging, Mayo Clinic, Rochester, MN, USA. ⁴Electrical Engineering, Brigham Young University, Provo, UT, USA. ⁵Enthought, Inc., Austin, TX, USA. ⁶Anaconda Inc., Austin, TX, USA. ⁷BioResource and Agricultural Engineering Department, California Polytechnic State University, San Luis Obispo, CA, USA. ⁸Department of Mathematics, University of California Los Angeles, Los Angeles, CA, USA. ⁹Los Alamos National Laboratory, Los Alamos, NM, USA. ¹⁰Independent researcher, Tokyo, Japan. ¹¹National Research University Higher School of Economics, Moscow, Russia. ¹²Independent researcher, Saue, Estonia. ¹³Department of Mechanics and Applied Mathematics, Institute of Cybernetics at Tallinn Technical University, Tallinn, Estonia. ¹⁴Berkeley Institute for Data Science, University of California Berkeley, Berkeley, CA, USA. ¹⁵Independent researcher, New York, NY, USA. ¹⁶School of Psychology, University of Birmingham, Edgbaston, Birmingham, UK. ¹⁷Independent researcher, San Francisco, CA, USA. ¹⁸Division of Biostatistics, University of California Berkeley, Berkeley, CA, USA. ¹⁹WayRay LLC, Skolkovo Innovation Center, Moscow, Russia. ²⁰Australian Nuclear Science and Technology Organisation, Lucas Heights, NSW, Australia. ²¹Institute for Learning and Brain Sciences, University of Washington, Seattle, WA, USA. ²²College of Information and Computing Sciences, University of Massachusetts Amherst, Amherst, MA, USA. ²³Independent researcher, Amsterdam, the Netherlands. ²⁴Berkeley Center for Cosmological Physics, University of California Berkeley, Berkeley, CA, USA. ²⁵Brucker Biospin Corp., Billerica, MA, USA. ²⁶University of Washington, Seattle, WA, USA. ²⁷Independent researcher, Toulouse, France. ²⁸Independent researcher, Montreal, Quebec, Canada. ²⁹New Technologies Research Centre, University of West Bohemia, Plzeň, Czech Republic. ³⁰Department of Mathematics, Brigham Young University, Provo, UT, USA. ³¹Oden Institute for Computational Engineering and Sciences, The University of Texas at Austin, Austin, TX, USA. ³²Independent researcher, Belmont, Massachusetts, USA. ³³Space Dynamics Laboratory, North Logan, UT, USA. ³⁴Independent researcher, Logan, Utah, USA. ³⁵Anton Pannekoek Institute, Amsterdam, The Netherlands. ³⁶Graduate Program in Electrical Engineering, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil. ³⁷Google LLC, Montreal, Quebec, Canada. ³⁸Google LLC, Cambridge, MA, USA. ³⁹A list of members and affiliations appears at the end of the paper. *e-mail: scipy.articles@gmail.com

PERSPECTIVE

NATURE METHODS

implementation, documentation and testing, and a culture eager to learn and adopt better practices—both for community management and software development.

Background

Here we capture a selective history of some milestones and important events in the growth of SciPy. Despite what we highlight here, it is important to understand that a project like SciPy is only possible because of the contributions of very many contributors—too many to mention individually, but each bringing an important piece to the puzzle.

Python is an interpreted, high-level, general-purpose computer programming language, designed by Guido van Rossum in the late 1980s, with a dynamic type system and an emphasis on readability and rapid prototyping¹³ (<https://github.com/python/cpython>). As a general-purpose programming language, it had no special support for scientific data structures or algorithms, unlike many of the other established computation platforms of the time. Yet scientists soon discovered the language's virtues, such as its ability to wrap C and Fortran libraries, and to then drive those libraries interactively. Scientists could thereby gain access to a wide variety of existing computational libraries without concerning themselves with low-level programming concepts such as memory management.

In 1995, Jim Hugunin, a graduate student at the Massachusetts Institute of Technology, wrote the first message in a new Python Matrix Special Interest Group (Matrix-SIG) mailing list¹⁴:

"There seems to be a fair amount of interest in the Python community concerning the addition of numeric operations to Python. My own desire is to have as large a library of matrix based functions available as possible (linear algebra, eigenfunctions, signal processing, statistics, etc.). In order to ensure that all of these libraries interoperate, there needs to be agreement on a basic matrix object that can be used to represent arrays of numbers."

Over the next several months, conversations on that mailing list by, among others, Jim Fulton, Jim Hugunin, Paul Dubois, Konrad Hinsen and Guido van Rossum led to the creation of a package called Numeric with an array object that supported a high number of dimensions. Jim Hugunin explained the utility of Python for numerical computation¹⁵:

"I've used almost all of the available numerical languages at one time or another over the past 8 years. One thing I've noticed is that over time, the designers of these languages are steadily adding more of the features that one would expect to find in a general-purpose programming language."

This remains a distinguishing feature of Python for science and one of the reasons why it has been so successful in the realm of data science: instead of adding general features to a language designed for numerical and scientific computing, here scientific features are added to a general-purpose language. This broadens the scope of problems that can be addressed easily, expands the sources of data that are readily accessible and increases the size of the community that develops code for the platform.

SciPy begins. By the late 1990s, discussions appeared on Matrix-SIG expressing a desire for a complete scientific data analysis environment¹⁶. Travis Oliphant, a PhD student at the Mayo Clinic, released a number of packages^{17,18} that built on top of the Numeric array package, and provided algorithms for signal processing, special functions, sparse matrices, quadrature, optimization, fast Fourier transforms and more. One of these packages, Multipack (<http://pylab.sourceforge.net/multipack.html>), was a set of extension modules that wrapped Fortran and C libraries to solve nonlinear equations and least-squares problems, integrate differential equations and fit splines. Robert Kern, then an undergraduate student

Box 1 | Excerpt from the SciPy 0.1 release announcement (typos corrected), posted 20 August 2001 on the Python-list mailing list

SciPy is an open-source package that builds on the strengths of Python and Numeric, providing a wide range of fast scientific and numeric functionality. SciPy's current module set includes the following:

- Special functions (Bessel, Hankel, Airy and others)
- Signal/image processing
- 2D plotting capabilities
- Integration
- ODE solvers
- Optimization (simplex, BFGS, Newton-CG and others)
- Genetic algorithms
- Numeric to C++ expression compiler
- Parallel programming tools
- Splines and interpolation
- Other items

(and currently a SciPy core developer), provided compilation instructions under Windows. Around the same time, Pearu Peterson, a PhD student from Estonia, released F2PY¹⁹, a command line tool for binding Python and Fortran codes, and wrote modules for linear algebra and interpolation. Eric Jones, while a graduate student at Duke University, wrote packages to support his dissertation, including a parallel job scheduler and genetic optimizer. Gary Strangman, a postdoctoral fellow at Harvard Medical School, published several descriptive and inferential statistical routines²⁰.

With a rich programming environment and a numerical array object in place, the time was ripe for the development of a full scientific software stack. In 2001, Eric Jones and Travis Vaught founded Enthought Scientific Computing Solutions (now Enthought, Inc.) in Austin, Texas, USA. To simplify the tool stack, they created the SciPy project, centered around the SciPy library, which would subsume all the above-mentioned packages. The new project quickly gained momentum, with a website and code repository²¹ appearing in February, and a mailing list announced²² in June 2001. By August 2001, a first release was announced²³, an excerpt of which is shown in Box 1. In September, the first documentation was published²⁴. The first SciPy workshop²⁵ was held in September 2002 at Caltech—a single track, two-day event with 50 participants, many of them developers of SciPy and surrounding libraries.

At this point, scientific Python started attracting more serious attention; code that started as side projects by graduate students had grown into essential infrastructure at national laboratories and research institutes. For example, Paul Dubois at Lawrence Livermore National Laboratory (LLNL) took over the maintenance of Numeric and funded the writing of its manual²⁶, and the Space Telescope Science Institute (STScI), which was in charge of Hubble Space Telescope science operations, decided to replace their custom scripting language and analysis pipeline with Python²⁷. As STScI continued to use Python for an increasingly large portion of the Hubble Space Telescope data analysis pipeline, they encountered problems with the Python numerical array container. Numeric, the original array package, was suitable for small arrays, but not for the large images processed by STScI. With the Numeric maintainer's blessing, the decision was made to write NumArray²⁸, a library that could handle data on a larger scale. Unfortunately, NumArray proved inefficient for small arrays, presenting the community with a rather unfortunate choice. In 2005, Travis Oliphant combined the best elements of Numeric and NumArray, thereby solving the dilemma. NumPy 1.0 was released²⁹ in October 2006, paving the way for the reunified scientific Python community to mature.

NATURE METHODS

PERSPECTIVE

Box 2 | Package organization

The SciPy library is organized as a collection of subpackages. The 16 subpackages include mathematical building blocks (for example, linear algebra, Fourier transforms, special functions), data structures (for example, sparse matrices, *k*-D trees), algorithms (for example, numerical optimization and integration, clustering, interpolation, graph algorithms, computational geometry) and higher-level data analysis functionality (for example, signal and image processing, statistical methods).

Here we summarize the scope and capabilities of each subpackage. Additional information is available in the SciPy tutorial (<https://docs.scipy.org/doc/scipy/reference/tutorial/>) and API reference (<https://docs.scipy.org/doc/scipy/reference/index.html#api-reference>).

cluster

The `cluster` subpackage contains `cluster.vq`, which provides vector quantization and *k*-means algorithms, and `cluster.hierarchy`, which provides functions for hierarchical and agglomerative clustering.

constants

Physical and mathematical constants, including the CODATA recommended values of the fundamental physical constants¹¹⁹.

fftpack

Fast Fourier Transform routines. In addition to the FFT itself, the subpackage includes functions for the discrete sine and cosine transforms and for pseudo-differential operators.

integrate

The `integrate` subpackage provides tools for the numerical computation of single and multiple definite integrals and for the solution of ordinary differential equations, including initial value problems and two-point boundary value problems.

interpolate

The `interpolate` subpackage contains spline functions and classes, one-dimensional and multi-dimensional (univariate and multivariate) interpolation classes, Lagrange and Taylor polynomial interpolators, and wrappers for FITPACK³³ and DFITPACK functions.

io

A collection of functions and classes for reading and writing Matlab (<https://www.mathworks.com/products/matlab.html>), IDL, Matrix Market¹²⁰, Fortran, NetCDF¹²¹, Harwell-Boeing¹²², WAV and ARFF data files.

linalg

Linear algebra functions, including elementary functions of a matrix, such as the trace, determinant, norm and condition number; basic solver for $Ax = b$; specialized solvers for Toeplitz matrices, circulant matrices, triangular matrices and other structured matrices; least-squares solver and pseudo-inverse calculations; eigenvalue and eigenvector calculations (basic and generalized); matrix decompositions, including Cholesky, Schur, Hessenberg, *LU*, *LDL^T*, *QR*, *QZ*, singular value and polar; and functions to create specialized matrices, such as diagonal, Toeplitz, Hankel, companion, Hilbert and more.

ndimage

This subpackage contains various functions for multi-dimensional image processing, including convolution and assorted linear and nonlinear filters (Gaussian filter, median filter, Sobel filter and others); interpolation; region labeling and processing; and mathematical morphology functions.

misc

A collection of functions that did not fit into the other subpackages. Although this subpackage still exists in SciPy 1.0, an effort is underway to deprecate or relocate the contents of this subpackage and remove it.

odr

Orthogonal distance regression, including Python wrappers for the Fortran library ODRPACK³⁴.

optimize

This subpackage includes simplex and interior-point linear programming solvers, implementations of many nonlinear minimization algorithms, a routine for least-squares curve fitting, and a collection of general nonlinear solvers for root-finding.

signal

The `signal` subpackage focuses on signal processing and basic linear systems theory. Functionality includes convolution and correlation, splines, filtering and filter design, continuous and discrete time linear systems, waveform generation, window functions, wavelet computations, peak finding and spectral analysis.

sparse

This subpackage includes implementations of several representations of sparse matrices. `scipy.sparse.linalg` provides a collection of linear algebra routines that work with sparse matrices, including linear equation solvers, eigenvalue decomposition, singular value decomposition and LU factorization. `scipy.sparse.csgraph` provides a collections of graph algorithms for which the graph is represented using a sparse matrix. Algorithms include connected components, shortest path, minimum spanning tree and more.

spatial

This subpackage provides spatial data structures and algorithms, including the *k*-d tree, Delaunay triangulation, convex hulls and Voronoi diagrams. `scipy.spatial.distance` provides a large collection of distance functions, along with functions for computing the distance between all pairs of vectors in a given collection of points or between all pairs from two collections of points.

special

The name comes from the class of functions traditionally known as special functions, but over time, the subpackage has grown to include functions beyond the classical special functions. A more appropriate characterization of this subpackage is simply useful functions. It includes a large collection of the classical special functions such as Airy, Bessel and others; families of orthogonal polynomials; the Gamma function, and functions related to it; functions for computing the PDF, CDF and quantile function for several probability distributions; information theory functions; combinatorial functions `comb` and `factorial`; and more.

stats

The `stats` subpackage provides a large collection of continuous and discrete probability distributions, each with methods to compute the PDF or PMF, CDF, moments and other statistics, generation of random variates and more; statistical tests, including tests on equality of means/medians/variance (such as the *t*-test) and tests whether a sample is drawn from a certain distribution (such as the Kolmogorov-Smirnov test); measures of correlation, including Pearson's *r*, Kendall's τ , and Spearman's ρ coefficients; descriptive statistics including trimmed values; kernel density estimation; and transformations of data such as the Box-Cox power transformation.

PERSPECTIVE

NATURE METHODS

SciPy matures. By the middle to late 2000s, SciPy was starting to mature after a long phase of significant growth and adoption. The scope of the SciPy library narrowed, while the breadth of the ecosystem grew through a new type of auxiliary package: the scikit (<https://www.scipy.org/scikits.html>), a complementary library developed outside SciPy, allowing for more rapid exploration of experimental ideas while maintaining familiar style and development methodology. In SciPy itself, tooling, development, documentation and release processes became more professional. The library was expanded carefully, with the patience affordable in open-source projects and via best practices, which are increasingly common in the scientific Python ecosystem and industry³⁰.

Early versions of SciPy had minimal documentation, but this began to change with the 2006 release of a *Guide to NumPy*¹. In 2007, Sphinx³¹ made it possible to render hypertext and PDF documents automatically from plain text (docstrings) interspersed with Python code, and in 2008, pydocweb³² enabled collaborative documentation development in a wiki-like fashion. The SciPy Documentation Project^{33,34} used these tools to complete documentation of SciPy's user-facing functionality: offering t-shirts to contributors from around the world in exchange for high-quality text, it collected contributions from over 75 people to produce an 884-page manual³⁵. Since then, SciPy has remained committed to maintaining high-quality documentation as part of the normal development cycle.

In the early SciPy workshops, recurrent topics reflected the state of development, with emphasis being placed on the underlying array package, plotting, parallel processing, acceleration/wrapping and user interfaces. By 2004, presentations about the application of SciPy to scientific problems began to appear. The event also started to draw in more keynote speakers from outside the community, such as Guido van Rossum (creator of Python, 2006), Ivan Krstić (One Laptop per Child, 2007), Alex Martelli (Google, 2008) and Peter Norvig (Google Research, 2009). The informal workshop grew from a small gathering of core developers into an international conference with hundreds of attendees, increased funding, a published proceedings and scholarships for attending students. By 2010, the US SciPy conference had multiple tracks, and satellite conferences were being organized by volunteers elsewhere, such as EuroSciPy (since 2008) and SciPy India (since 2009). Special sessions and minisymposia dedicated to scientific Python began appearing at many other events. For example, a three-part minisymposium organized for International Conferences on Computational Science and Engineering (CSE) 2009 was featured in *SIAM News*³⁶.

In 2007, Python had a strong enough presence in science and engineering that the editors of *IEEE Computing in Science and Engineering* solicited a special issue about Python in science³⁷, edited by Paul Dubois. However, Python was still sufficiently niche that the average reader would need additional information to decide whether it would be useful in their own work. The follow-up March/April 2011 Python for Scientists and Engineers special issue³⁸ focused more on the core parts of the scientific Python ecosystem³⁹ including NumPy², Cython⁴⁰ and Mayavi⁴¹. Python became so pervasive that journals began publishing domain-specific special issues. For example, in 2015, *Frontiers in Neuroinformatics* published a collection of 25 articles—covering topics including modeling and simulation, data collection, electrophysiology, visualization as well as stimulus generation and presentation—called Python in Neuroscience⁴².

SciPy today. As of February 2019, the SciPy library consists of nearly 600,000 lines of open-source code organized in 16 subpackages summarized in Box 2. The development team and community currently interact and operate primarily on GitHub, an online version control and task management platform. Over 110,000 GitHub repositories and 6,500 packages depend on SciPy⁴³. Some of the major feature highlights from the three years preceding SciPy 1.0

are discussed in the “Key technical improvements” section below, and milestones in its history are highlighted in Fig. 1.

Architecture and implementation choices

Project scope. SciPy provides fundamental algorithms for scientific computing. The breadth of its scope was derived from the guide to available mathematical software (GAMS) classification system⁴⁴. In areas that move relatively slowly, for example, linear algebra, SciPy aims to provide complete coverage. In other areas it aims to provide fundamental building blocks while interacting well with other packages specialized in that area. For example, SciPy provides what one expects to find in a statistics textbook (probability distributions, hypothesis tests, frequency statistics, correlation functions, and more), whereas Statsmodels⁴⁵ provides more advanced statistical estimators and inference methods, scikit-learn³ covers machine learning, and PyMC3⁴⁶, emcee⁴⁷ and PyStan (<http://mc-stan.org>) cover Bayesian statistics and probabilistic modeling. scikit-image⁴ provides image processing capabilities beyond SciPy's `ndimage`, SymPy⁴⁸ provides a Python interface for symbolic computation, and `sparse.csgraph` and `spatial` offer basic tools for working with graphs and networks compared to specialized libraries such as NetworkX⁴⁹.

We use the following criteria to determine whether to include new functionality in SciPy:

- The algorithm is of relevance to multiple fields of science.
- The algorithm is demonstrably important. For example, it is classic enough to be included in textbooks, or it is based on a peer-reviewed article that has a substantial number of citations.

In terms of software systems and architecture, SciPy's scope matches NumPy's: algorithms for in-memory computing on single machines, with support for a wide range of data types and process architectures. Distributed computing and support for graphics processing units (GPUs) were explicitly out of scope at the 1.0 release point, but this has been revised in our roadmap (see Discussion).

Language choices. According to analysis using the `linguist` library (<https://github.com/github/linguist>), SciPy is approximately 50% Python, 25% Fortran, 20% C, 3% Cython and 2% C++, with a dash of TeX, Matlab, shell script and Make. The distribution of secondary programming languages in SciPy is a compromise between a powerful, performance-enhancing language that interacts well with Python (that is, Cython) and the usage of languages (and their libraries) that have proven reliable and performant over many decades.

Fortran, despite its age, is still a high-performance scientific programming language with continued contemporary usage⁵⁰. Thus, we wrap the following excellent, field-tested Fortran libraries to provide Python convenience while benefiting from their performance: QUADPACK⁵¹ and ODEPACK⁵² for numerical integration and solution of initial value problems; FITPACK⁵³, ODRPACK⁵⁴ and MINPACK⁵⁵ for curve-fitting and least-squares minimization; FFTPACK^{56,57} for performing Fourier transforms; ARPACK⁵⁸ for solving eigenvalue problems; ALGORITHM 644 (ref. ⁵⁹) for computing Bessel functions; and CDFLIB⁶⁰ for evaluating cumulative density functions.

Rounding out the top three languages in SciPy is C, which is also extremely well-established over several decades⁶¹ of scientific computing. The C libraries that we wrap in SciPy include `trlib`⁶² for optimization, SuperLU^{63,64} for solving sparse linear systems, Qhull⁶⁵ for computational geometry and Cephes (<http://www.netlib.org/cephes/>) for special functions.

Cython has been described as a creole language that mixes the best parts of Python and lower-level C/C++ paradigms⁴⁰. We often use Cython as a glue between well-established, low-level scientific

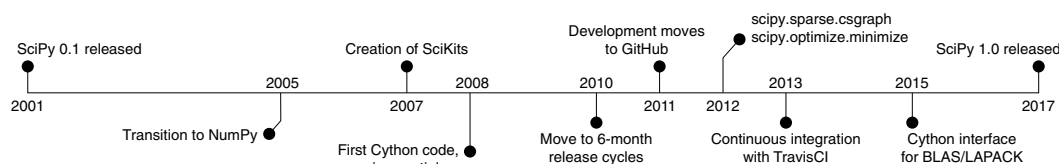


Fig. 1 | Major milestones from SciPy's initial release in 2001 to the release of SciPy 1.0 in 2017. Note that SciKits and GitHub have been introduced in the Background section; more information about Cython and SciPy subpackages (for example, `scipy.sparse`) is available in the 'Architecture and implementation choices' section, BLAS/LAPACK support is detailed in the 'Key technical improvements' section, and continuous integration is discussed in the 'Test and benchmark suite' section.

computing libraries written in C/C++ and the Python interface offered by SciPy. We also use Cython to enable performance enhancements in Python code, especially for cases where heavily used inner loops benefit from a compiled code with static typing.

For implementing new functionality, Python is still the language of choice. If Python performance is an issue, then we prefer the use of Cython followed by C, C++ or Fortran (in that order). The main motivation for this is maintainability: Cython has the highest abstraction level, and most Python developers will understand it. C is also widely known, and easier for the current core development team to manage than C++ and especially Fortran.

The position that SciPy occupies near the foundation of the scientific Python ecosystem is such that adoption of new languages or major dependencies is generally unlikely; our choices are strongly driven by long-term stability. GPU acceleration, new transpiling libraries and the latest JIT compilation approaches (for example, Numba⁶⁶) are very powerful but have traditionally fallen outside the remit of the main SciPy library. That said, we have recently increased our efforts to support compatibility with some of these options, and our full test suite passed with the PyPy JIT compiler⁶⁷ at the 1.0 release point.

API and ABI evolution. The API for SciPy consists of approximately 1,500 functions and classes. Our policy for evolving the API over time is that new functionality can be added, while removing or changing existing functionality can only be done if the benefits exceed the (often significant) costs to users and only after giving clear deprecation warnings to those users for at least one year. In general, we encourage changes that improve clarity in the API of the library but strongly discourage breaking backward compatibility, given our position near the base of the scientific Python computing stack.

In addition to the Python API, SciPy has C and Cython interfaces. Therefore, we also have to consider the application binary interface (ABI). This ABI has been stable for a long time, and we aim to evolve it only in a backward-compatible way.

Key technical improvements

Here we describe key technical improvements made in the last three years.

Data structures. Sparse matrices. `scipy.sparse` offers seven sparse matrix data structures, also known as sparse formats. The most important ones are the row- and column-compressed formats (CSR and CSC, respectively). These offer fast major-axis indexing and fast matrix-vector multiplication, and are used heavily throughout SciPy and dependent packages.

Over the last three years, our sparse matrix handling internals were rewritten and performance was improved. Iterating over and slicing of CSC and CSR matrices is now up to 35% faster, and the speed of coordinate (COO)/diagonal (DIA) to CSR/CSC matrix format conversions has increased. SuperLU⁶³ was updated to version

5.2.1, enhancing the low-level implementations leveraged by a subset of our `sparse` offerings.

From a new features standpoint, `scipy.sparse` matrices and linear operators now support the Python matrix multiplication (`@`) operator. We added `scipy.sparse.norm` and `scipy.sparse.random` for computing sparse matrix norms and drawing random variates from arbitrary distributions, respectively. Also, we made a concerted effort to bring the `scipy.sparse` API into line with the equivalent NumPy API where possible.

cKDTree. The `scipy.spatial.ckdtree` module, which implements a space-partitioning data structure that organizes points in k -dimensional space, was rewritten in C++ with templated classes. Support was added for periodic boundary conditions, which are often used in simulations of physical processes.

In 2013, the time complexity of the k -nearest-neighbor search from `cKDTree.query` was approximately $\log \text{linear}$ ⁶⁸, consistent with its formal description⁶⁹. Since then, we enhanced `cKDTree.query` by reimplementing it in C++, removing memory leaks and allowing release of the global interpreter lock (GIL) so that multiple threads may be used⁷⁰. This generally improved performance on any given problem while preserving the asymptotic complexity.

In 2015, SciPy added the `sparse_distance_matrix` routine for generating approximate sparse distance matrices between `KDTree` objects by ignoring all distances that exceed a user-provided value. This routine is not limited to the conventional L2 (Euclidean) norm but supports any Minkowski p -norm between 1 and infinity. By default, the returned data structure is a dictionary of keys (DOK)-based sparse matrix, which is very efficient for matrix construction. This hashing approach to sparse matrix assembly can be seven times faster than constructing with CSR format⁷¹, and the C++ level sparse matrix construction releases the Python GIL for increased performance. Once the matrix is constructed, distance value retrieval has an amortized constant time complexity⁷², and the DOK structure can be efficiently converted to a CSR, CSC or COO matrix to allow for speedy arithmetic operations.

In 2015, the `cKDTree` dual tree counting algorithm⁷³ was enhanced to support weights⁷⁴, which are essential in many scientific applications, for example, computing correlation functions of galaxies⁷⁵.

Unified bindings to compiled code. `LowLevelCallable`. As of SciPy version 0.19, it is possible for users to wrap low-level functions in a `scipy.LowLevelCallable` object that reduces the overhead of calling compiled C functions, such as those generated using Numba or Cython, directly from Python. Supported low-level functions include `PyCapsule` objects, `ctypes` function pointers and `cfi` function pointers. Furthermore, it is possible to generate a low-level callback function automatically from a Cython module using `scipy.LowLevelCallable.from_cython`.

PERSPECTIVE

NATURE METHODS

Table 1 | Optimization methods from `minimize`

	Nelder-Mead	Powell	COBYLA	CG	BFGS	L-BFGS-G	SLSQP	TNC	Newton-CG	dogleg	trust-ncg	trust-exact	trust-krylov
Version added	0.6*	0.6*	0.6*	0.6*	0.6*	0.6*	0.9	0.6*	0.6*	0.13	0.13	0.19	1.0
Wrapper			✓			✓	✓	✓					✓
First derivatives				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Second derivatives				-	-	-	✓	✓	✓	✓	✓	✓	✓
Iterative Hessian factorization							✓	✓			✓		✓
Local convergence				L	S	L	S	S*	S*	Q	S*	Q	S*
Global convergence					✓	✓	✓	✓	✓	✓	✓	✓	✓
Trust region	Neither	LS	TR	LS	LS	LS	LS	LS	LS	TR	TR	TR	TR
Bound constraints					✓	✓	✓	✓					
Equality constraints							✓						
Inequality constraints			✓				✓						
References	98,99	100	101-103	104,105	105	106,107	108-111	112	105	105,113	105,114	115,116	62,117

Optimization methods from `minimize`, which solves problems of the form $\min_x f(x)$, where $x \in \mathbb{R}^n$ and $f: \mathbb{R}^n \rightarrow \mathbb{R}$. 'Version added' specifies the algorithm's first appearance in SciPy. Algorithms with version added "0.6*" were added in version 0.6 or before. 'Wrapper' indicates whether the implementation available in SciPy wraps a function written in a compiled language (for example, C or FORTRAN). 'First derivatives' and 'second derivatives' indicate whether first or second order derivatives are required. When 'second derivatives' is flagged with "-", the algorithm accepts but does not require second-order derivatives from the user; it computes an approximation internally and uses it to accelerate method convergence. 'Iterative Hessian factorization' denotes algorithms that factorize the Hessian in an iterative way, which does not require explicit matrix factorization or storage of the Hessian. 'Local convergence' gives a lower bound on the rate of convergence of the iteration sequence once the iterate is sufficiently close to the solution: linear (L), superlinear (S) and quadratic (Q). Convergence rates denoted S* indicate that the algorithm has a superlinear rate for the parameters used in SciPy, but can achieve a quadratic convergence rate with other parameter choices. 'Global convergence' is marked for the algorithms with guarantees of convergence to a stationary point (that is, a point x^* for which $\nabla f(x^*) = 0$); this is not a guarantee of convergence to a global minimum. 'Lines-search' (LS) or 'trust-region' (TR) indicates which of the two globalization approaches is used by the algorithm. The table also indicates which algorithms can deal with constraints on the variables. We distinguish among bound constraints ($x^l \leq x \leq x^u$), equality constraints ($c_{\text{eq}}(x) = 0$) and inequality constraints ($c_{\text{ineq}}(x) \geq 0$).

Cython bindings for BLAS, LAPACK and `special`. SciPy has provided special functions and leveraged basic linear algebra subprograms (BLAS) and linear algebra package (LAPACK)⁷⁶ routines for many years. SciPy now additionally includes Cython⁴⁰ wrappers for many BLAS and LAPACK routines (added in 2015) and the special functions provided in the `scipy.special` subpackage (added in 2016), which are available in `scipy.linalg.cython_blas`, `scipy.linalg.cython_lapack` and `scipy.special.cython_special`, respectively. When writing algorithms in Cython, it is typically more efficient to call directly into the libraries SciPy wraps rather than indirectly, using SciPy's Python APIs. These low-level interfaces for Cython can also be used outside of the SciPy codebase to gain access to the functions in the wrapped libraries while avoiding the overhead of Python function calls. This can give performance gains of one or two orders of magnitude for many use cases.

Developers can also use the low-level Cython interfaces without linking against the wrapped libraries⁷⁷. This lets other extensions avoid the complexity of finding and using the correct libraries. Avoiding this complexity is especially important when wrapping libraries written in Fortran. Not only can these low-level wrappers be used without a Fortran compiler, they can also be used without having to handle all the different Fortran compiler ABIs and name mangling schemes.

Most of these low-level Cython wrappers are generated automatically to help with both correctness and ease of maintenance. The wrappers for BLAS and LAPACK are primarily generated using type information that is parsed from the BLAS and LAPACK source files using F2PY¹⁹, though a small number of routines use hand-written type signatures instead. The input and output types of each routine are saved in a data file that is read at build time and used to generate the corresponding Cython wrapper files. The wrappers in `scipy.special.cython_special` are also generated from a data file containing type information for the wrapped routines.

Since SciPy can be built with LAPACK 3.4.0 or later, Cython wrappers are only provided for the routines that maintain a consistent

interface across all supported LAPACK versions. The standard BLAS interface provided by the various existing BLAS libraries is not currently changing, so changes are not generally needed in the wrappers provided by SciPy. Changes to the Cython wrappers for the functions in `scipy.special` follow corresponding changes to the interface of that subpackage.

Numerical optimization. The `scipy.optimize` subpackage provides functions for the numerical solution of several classes of root finding and optimization problems. Here we highlight recent additions through SciPy 1.0.

Linear optimization. A new interior-point optimizer for continuous linear programming problems, `linprog` with `method = 'interior-point'`, was released with SciPy 1.0. Implementing the core algorithm of the commercial solver MOSEK⁷⁸, it solves all of the 90+ NETLIB LP benchmark problems⁷⁹ tested. Unlike some interior point methods, this homogeneous self-dual formulation provides certificates of infeasibility or unboundedness as appropriate.

A presolve routine⁸⁰ solves trivial problems and otherwise performs problem simplifications, such as bound tightening and removal of fixed variables, and one of several routines for eliminating redundant equality constraints is automatically chosen to reduce the chance of numerical difficulties caused by singular matrices. Although the main solver implementation is pure Python, end-to-end sparse matrix support and heavy use of SciPy's compiled linear system solvers—often for the same system with multiple right hand sides owing to the predictor-corrector approach—provide speed sufficient for problems with tens of thousands of variables and constraints.

Nonlinear optimization: local minimization. The `minimize` function provides a unified interface for finding local minima of nonlinear optimization problems. Four new methods for unconstrained optimization were added to `minimize` in recent versions of SciPy: `dogleg`, `trust-ncg`, `trust-exact` and `trust-krylov`.

NATURE METHODS

PERSPECTIVE

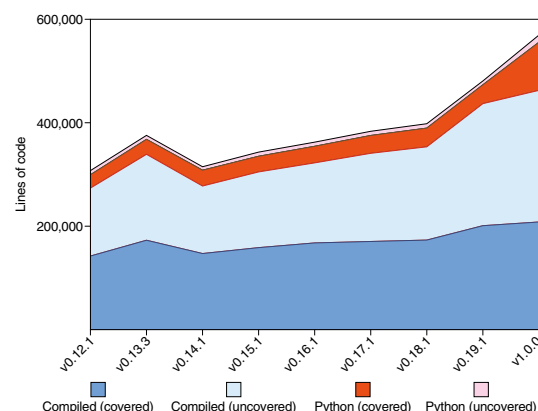


Fig. 2 | Python and compiled code volume in SciPy over time.

All are trust-region methods that build a local model of the objective function based on first and second derivative information, approximate the best point within a local ‘trust region’ and iterate until a local minimum of the original objective function is reached, but each has unique characteristics that make it appropriate for certain types of problems. For instance, `trust-exact` achieves fast convergence by solving the trust-region subproblem almost exactly, but it requires the second derivative Hessian matrix to be stored and factored every iteration, which may preclude the solution of large problems ($\geq 1,000$ variables). In contrast, `trust-ncg` and `trust-krylov` are well suited to large-scale optimization problems because they do not need to store and factor the Hessian explicitly, instead using second derivative information in a faster, approximate way. We compare the characteristics of all `minimize` methods in detail in Table 1, which illustrates the level of completeness that SciPy aims for when covering a numerical method or topic.

Nonlinear optimization: global minimization. As `minimize` may return any local minimum, some problems require the use of a global optimization routine. The new `scipy.optimize.differential_evolution` function^{81,82} is a stochastic global optimizer that works by evolving a population of candidate solutions. In each iteration, trial candidates are generated by combination of candidates from the existing population. If the trial candidates represent an improvement, then the population is updated. Most recently, the SciPy benchmark suite gained a comprehensive set of 196 global optimization problems for tracking the performance of existing solvers over time and for evaluating whether the performance of new solvers merits their inclusion in the package.

Statistical distributions. The `scipy.stats` subpackage contains more than 100 probability distributions: 96 continuous and 13 discrete univariate distributions, and 10 multivariate distributions. The implementation relies on a consistent framework that provides methods to sample random variates, to evaluate the cumulative distribution function (CDF) and the probability density function (PDF), and to fit parameters for every distribution. Generally, the methods rely on specific implementations for each distribution, such as a closed-form expression of the CDF or a sampling algorithm, if available. Otherwise, default methods are used based on generic code, for example, numerical integration of the PDF to obtain the CDF. Key recent distributions added to `scipy.stats` include the histogram-based distribution in `scipy.stats.rv_histogram` and the multinomial distribution in `scipy.`

`stats.multinomial` (used, for example, in natural language processing⁸³).

Polynomial interpolators. Historically, SciPy relied heavily on the venerable FITPACK Fortran library by P. Dierckx^{53,84} for univariate interpolation and approximation of data, but the original monolithic design and API for interaction between SciPy and FITPACK was limiting for both users and developers.

Implementing a new, modular design of polynomial interpolators was spread over several releases. The goals of this effort were to have a set of basic objects representing piecewise polynomials, to implement a collection of algorithms for constructing various interpolators, and to provide users with building blocks for constructing additional interpolators.

At the lowest level of the new design are classes that represent univariate piecewise polynomials: `PPoly` (SciPy 0.13)⁸⁵, `BPoly` (SciPy 0.13) and `BSpline` (SciPy 0.19)⁸⁶, which allow efficient vectorized evaluations, differentiation, integration and root-finding. `PPoly` represents piecewise polynomials in the power basis in terms of breakpoints and coefficients at each interval. `BPoly` is similar and represents piecewise polynomials in the Bernstein basis (which is suitable, for example, for constructing Bézier curves). `BSpline` represents spline curves, that is, linear combinations of B-spline basis elements⁸⁷.

In the next layer, these polynomial classes are used to construct several common ways of interpolating data: `CubicSpline` (SciPy 0.18)⁸⁸ constructs a twice differentiable piecewise cubic function, `Akima1DInterpolator` and `PCHIPInterpolator` implement two classic prescriptions for constructing a C^1 continuous monotone shape-preserving interpolator^{89,90}.

Test and benchmark suite. *Test suite.* Test-driven development has been described as a way to manage fear and uncertainty when making code changes⁹¹. For each component of SciPy, we write multiple small executable tests that verify its intended behavior. The collection of these, known as a ‘test suite’, increases confidence in the correctness and accuracy of the library, and allows us to make code modifications known not to alter desired behavior. According to the practice of continuous integration⁹², all proposed contributions to SciPy are temporarily integrated with the master branch of the library before the test suite is run, and all tests must be passed before the contribution is permanently merged. Continuously monitoring the number of lines of code in SciPy covered by unit tests is one way we maintain some certainty that changes and new features are correctly implemented.

The SciPy test suite is orchestrated by a continuous integration matrix that includes POSIX and Windows (32/64-bit) platforms managed by Travis CI and AppVeyor, respectively. Our tests cover Python versions 2.7, 3.4, 3.5, 3.6, and include code linting with `pyflakes` and `pycodestyle`. There are more than 13,000 unit tests in the test suite, which is written for usage with the `pytest` (<https://docs.pytest.org/en/latest>) framework. In Fig. 2, we show historical test coverage data generated using a Docker-based approach (<https://github.com/tylerjerry/scipy-cov-track>). With the exception of the removal of ~61,000 lines of compiled code for SciPy v0.14, the volume of both compiled (C, C++ and Fortran) and Python code has increased between releases, as have the number of lines covered by unit tests. Test coverage at the SciPy 1.0 release point was at 87% for Python code according to `pytest-cov` (<https://pypi.org/project/pytest-cov/>). Coverage of compiled (C, C++ and Fortran) code was only 45% according to `gcov` (<https://gcc.gnu.org/onlinedocs/gcc/Gcov.html>), but the compiled codebase is much more robust than this figure would suggest as the figure does not correct for the inclusion of reputable vendor code, the original library of which is well-tested; generated code, for which full coverage is impractical; and deprecated code, which does

PERSPECTIVE

NATURE METHODS



Fig. 3 | Results of the `scipy.spatial.cKDTree.query` benchmark from the introduction of `cKDTree` to the release of SciPy 1.0. The benchmark generates a k -d tree from uniformly distributed points in an m -dimensional unit hypercube, then finds the nearest (Euclidean) neighbor in the tree for each of 1,000 query points. Each marker in the figure indicates the execution time of the benchmark for a commit in the master branch of SciPy.

not require unit tests. Documentation for the code is automatically built and published by the CircleCI service to facilitate evaluation of documentation changes/integrity.

Benchmark suite. In addition to ensuring that unit tests are passing, it is important to confirm that the performance of the SciPy codebase improves over time. Since February 2015, the performance of SciPy has been monitored with Airspeed Velocity ([asv](https://github.com/airspeed-velocity/av) <https://github.com/airspeed-velocity/av>). SciPy's `run.py` script conveniently wraps `asv` features such that benchmark results over time can be generated with a single console command. For example, in Fig. 3 we illustrate the improvement of `scipy.spatial.cKDTree.query` over roughly nine years of project history. The tree used in the benchmark was generated without application of toroidal topology (`boxsize = None`), and tests were performed by Airspeed Velocity 0.4 using Python 2.7, NumPy 1.8.2 and Cython versions 0.27.3, 0.21.1 and 0.18 (for improved backward compatibility). Substantial performance improvements were realized when `cKDTree` was fully Cythonized and again when it was rewritten in C++.

Project organization and community

Governance. SciPy adopted an official governance document (<https://docs.scipy.org/doc/scipy/reference/dev/governance/governance.html>) on August 3, 2017. A steering council, currently composed of 18 members, oversees daily development of the project by contributing code and reviewing contributions from the community. Council members have commit rights to the project repository, but they are expected to merge changes only when there are no substantive community objections. The chair of the steering council, Ralf Gommers, is responsible for initiating biannual technical reviews of project direction and summarizing any private council activities to the broader community. The project's benevolent dictator for life, Pauli Virtanen, has overruling authority on any matter, but is expected to act in good faith and only exercise this authority when the steering council cannot reach agreement.

SciPy's official code of conduct was approved on October 24, 2017. In summary, there are five specific guidelines: be open to everyone participating in our community; be empathetic and patient in resolving conflicts; be collaborative, as we depend on each other to build the library; be inquisitive, as early identification of issues can prevent serious consequences; and be careful with wording. The code of conduct specifies how breaches can be reported to

a code of conduct committee and outlines procedures for the committee's response. Our diversity statement "welcomes and encourages participation by everyone."

Maintainers and contributors. The SciPy project has ~100 unique contributors for every 6-month release cycle. Anyone with the interest and skills can become a contributor; the SciPy contributor guide (https://scipy.github.io/devdocs/dev/contributor/contributor_toc.html) provides guidance on how to do that. In addition, the project currently has 15 active (volunteer) maintainers: people who review the contributions of others and do everything else needed to ensure that the software and the project move forward. Maintainers are critical to the health of the project⁹³; their skills and efforts largely determine how fast the project progresses, and they enable input from the much larger group of contributors. Anyone can become a maintainer, too, as they are selected on a rolling basis from contributors with a substantial history of high-quality contributions.

Funding. The development cost of SciPy is estimated in excess of 10 million dollars by Open Hub (https://www.openhub.net/p/scipy/estimated_cost). Yet the project is largely unfunded, having been developed predominantly by graduate students, faculty and members of industry in their free time. Small amounts of funding have been applied with success: some meetings were sponsored by universities and industry, Google's Summer of Code program supported infrastructure and algorithm work, and teaching grant funds were used early on to develop documentation. However, funding from national agencies, foundations and industry has not been commensurate with the enormous stack of important software that relies on SciPy. More diverse spending to support planning, development, management and infrastructure would help SciPy remain a healthy underpinning of international scientific and industrial endeavors.

Downstream projects. The scientific Python ecosystem includes many examples of domain-specific software libraries building on top of SciPy features and then returning to the base SciPy library to suggest and even implement improvements. For example, there are common contributors to the SciPy and Astropy core libraries⁹⁴, and what works well for one of the codebases, infrastructures or communities is often transferred in some form to the other. At the codebase level, the `binned_statistic` functionality is one such cross-project contribution: it was initially developed in

Table 2 | Summary of SciPy Roadmap items following 1.0 release

SciPy subpackage	Summary of change
optimize	A few more high-quality global optimizers
fftpack	Reduce overlap with NumPy equivalent
linalg	Reduce overlap with NumPy equivalent
interpolate	New spline fitting and arithmetic routines
interpolate	New transparent tensor-product splines
interpolate	New non-uniform rational B-splines
interpolate	Mesh refinement and coarsening of B-splines and tensor products
signal	Migrate spline functionality to <code>interpolate</code>
signal	Second order sections update to match capabilities in other routines
linalg	Support a more recent version of LAPACK
ndimage	Clarify usage of the 'data point' coordinate model, and add additional wrapping modes
sparse	Incorporate sparse arrays from Sparse package ¹¹⁸
sparse.linalg	Add PROPACK wrappers for faster SVD
spatial	Add support for (quaternion) rotation matrices
special	Precision improvements for hypergeometric, parabolic cylinder and spheroidal wave functions

an Astropy-affiliated package and then placed in SciPy afterward. In this perspective, SciPy serves as a catalyst for cross-fertilization throughout the Python scientific computing community.

Discussion

SciPy has a strong developer community and a massive user base. GitHub traffic metrics report roughly 20,000 unique visitors to the source website between 14 May 2018 and 27 May 2018 (near the time of writing), with 721 unique copies ('clones') of the codebase over that time period. The developer community at that time consisted of 610 unique contributors of source code, with more than 19,000 commits accepted into the codebase (GitHub page data).

From the user side, there were 13,096,468 downloads of SciPy from the Python Packaging Index (PyPI)⁹⁵ and 5,776,017 via the default channel of the `conda` (<https://github.com/ContinuumIO/anaconda-package-data>) package manager during the year 2017. These numbers establish a lower bound on the total number of downloads by users given that PyPI and `conda` are only two of several popular methods for installing SciPy. The SciPy website (<http://www.scipy.org/>), which has been the default citation in the absence of a peer-reviewed paper, has been cited over 3,000 times (<https://scholar.google.com/scholar?cites=2086009121748039507>). Some of the most prominent uses or demonstrations of credibility for SciPy include the LIGO-Virgo scientific collaboration that lead to the observation of gravitational waves⁹⁶, the fact that SciPy is shipped directly with macOS and in the Intel distribution for Python⁹⁷, and that SciPy is used by 47% of all machine learning projects on GitHub (<https://github.blog/2019-01-24-the-state-of-the-octoverse-machine-learning/>).

Nevertheless, SciPy continually strives to improve. The SciPy Roadmap (<https://docs.scipy.org/doc/scipy-1.0.0/reference/roadmap.html>, <https://scipy.github.io/devdocs/roadmap.html>), summarized in Table 2, is a continually updated document maintained by the community that describes some of the major directions for improvement for the project, as well as specific limitations and matters that require assistance moving forward. In addition to the items on the roadmap, we are still working to increase the number

of SciPy usage tutorials beyond our current 15 section offering. Also, the low-level Cython code in our library (which interacts with C-level code and exposes it for Python usage) could use some measure of modernization, including migration to typed memoryviews to handle NumPy arrays.

A problem faced by many open-source projects is attracting and retaining developers. Although it is normal for some individuals to contribute to a project for a while and then move on, too much turnover can result in the loss of institutional memory, leading to mistakes of the past being repeated, APIs of new code becoming inconsistent with the old code and a drifting project scope. We are fortunate that the SciPy project continues to attract enthusiastic and competent new developers while maintaining the involvement of a small but dedicated old guard. There are contributors who were present in the early years of the project who still contribute to discussions of bug reports and reviews of new code contributions. Our benevolent dictator for life has been with the project for more than 10 years and is still actively contributing code, and the head of our steering council, who also acts as a general manager, is approaching his eleventh anniversary. An additional half dozen or so active developers have been contributing steadily for five or more years. The combination of a committed old guard and a host of new contributors ensures that SciPy will continue to grow while maintaining a high level of quality.

A final important challenge to address is the accommodation of GPU and distributed computing without disrupting our conventional and heavily used algorithm/API infrastructure. Although the exact approach we will adopt across the entire library to leverage these emerging technologies remains unclear, and was not a priority at the 1.0 release point, we now have a concrete implementation of a subpackage that allows for the experimental use of multiple backends, such as GPU-tractable data structures, in the new `scipy.fft`. This will be described in detail in a future report.

Reporting Summary. Further information on research design is available in the Nature Research Reporting Summary linked to this article.

Data availability

Raw data for Fig. 2 are available at <https://github.com/tylerjereddy/scipy-cov-track>, and raw data for Fig. 3 are available at https://github.com/scipy/scipy-articles/tree/master/scipy-1.0/supporting_info/asv_bench/cKDTTree.

Code availability

All SciPy library source code is available in the SciPy GitHub repository, <https://github.com/scipy/scipy>.

Received: 28 July 2019; Accepted: 14 November 2019;

Published online: 3 February 2020

References

1. Oliphant, T.E. *Guide to NumPy* 1st edn (Trelgol Publishing USA, 2006).
2. van derWalt, S., Colbert, S. C. & Varoquaux, G. The NumPy array: a structure for efficient numerical computation. *Comput. Sci. Eng.* **13**, 22–30 (2011).
3. Pedregosa, F. et al. Scikit-learn: machine learning in Python. *J. Mach. Learn. Res.* **12**, 2825–2830 (2011).
4. van derWalt, S. et al. scikit-image: image processing in Python. *Peer J.* **2**, e453 (2014).
5. Nitz, A. et al. gwastro/pycbc: PyCBC v1.13.2 release, <https://doi.org/10.5281/zenodo.1596771> (27 November 2018).
6. Vallisneri, M., Kanner, J., Williams, R., Weinstein, A. & Stephens, B. The LIGO Open Science Center. *J. Phys. Conf. Ser.* **610**, 012021 (2015).
7. Abbott, B. P. et al. GW150914: First results from the search for binary black hole coalescence with Advanced LIGO. *Phys. Rev. D.* **93**, 122003 (2016).
8. Abbott, B. P. et al. GW170817: observation of gravitational waves from a binary neutron star inspiral. *Phys. Rev. Lett.* **119**, 161101 (2017).

PERSPECTIVE

NATURE METHODS

9. The Event Horizon Telescope Collaboration et al. First M87 event horizon telescope results. III. Data processing and calibration. *Astrophys. J. Lett.* **875**, L3 (2019).
10. Blanton, K. At Mathworks, support + fun = success: CEO Jack Little believes in power of his workers—and their ideas. *The Boston Globe*, J5 (20 April 1997).
11. Howell, D. Jack Dangermond's digital mapping lays it all out. *Investor's Business Daily* (14 August 2009).
12. Port, O. Simple solutions. *BusinessWeek*, 24–24 (3 October 2005).
13. van Rossum, G. *Python/C API Reference Manual*, <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.211.6702&rep=rep1&type=pdf> (2001).
14. Hugunin, J. The matrix object proposal (very long). <https://mail.python.org/pipermail/matrix-sig/1995-August/000002.html> (18 August 1995).
15. Hugunin, J. Extending Python for numerical computation, <http://hugunin.net/papers/hugunin95numpy.html> (1995).
16. Oliphant, T. E. Moving forward from the last decade of SciPy. Presentation slides, https://conference.scipy.org/scipy2010/slides/travis_oliphant_keynote.pdf (1 July 2010).
17. Oliphant, T. E. Some Python modules. *Web Archive*, <https://web.archive.org/web/19990125091242/http://oliphant.netpedia.net:80/> (25 January 1999).
18. Oliphant, T. E. Modules to enhance Numerical Python. *Web Archive*, <https://web.archive.org/web/20001206213500/http://oliphant.netpedia.net:80/> (6 December 2000).
19. Peterson, P.F.2PY: a tool for connecting Fortran and Python programs. *Int. J. Comput. Sci. Eng.* **4**, 296–305 (2009).
20. Strangman, G. Python modules. *Web Archive*, https://web.archive.org/web/20001022231108/http://www.nmr.mgh.harvard.edu/Neural_Systems_Group/gary/python.html (2000).
21. SciPy Developers. SciPy.org. *Web Archive*, <https://web.archive.org/web/20010309040805/http://scipy.org:80/> (2001).
22. Vaught, T. N. SciPy Developer mailing list now online, <https://mail.python.org/pipermail/scipy-dev/2001-June/000000.html> (2001).
23. Jones, E. ANN: SciPy 0.10—scientific computing with Python, <https://mail.python.org/pipermail/python-list/2001-August/106419.html> (2001).
24. Vaught, T. N. Reference documentation and Tutorial documentation are now available for download as tarballs. *Web Archive*, https://web.archive.org/web/20021013204556/http://www.scipy.org:80/scipy/site_content/site_news/docs_released1 (2002).
25. Vaught, T. N. [ANN] SciPy '02 - Python for Scientific Computing Workshop, <https://mail.python.org/pipermail/numpy-discussion/2002-June/001511.html> (2002).
26. Ascher, D., Dubois, P. F., Hinsen, K., Hugunin, J., & Oliphant, T. E. An open source project: Numerical Python, <https://doi.org/10.5281/zenodo.3599566> (2001).
27. Greenfield, P. How Python slithered into astronomy. Presentation, https://conference.scipy.org/scipy2011/slides/greenfield_keynote_astronomy.pdf (2011).
28. Greenfield, P., Miller, J. T., Hsu, J. T. & White, R. L. numarray: a new scientific array package for Python. *PyCon DC* (2003).
29. NumPy Developers. v1.0, <https://github.com/numpy/numpy/releases/tag/v1.0> (25 October 2006).
30. Millman, K. J. & Pérez, F. Developing open-source scientific practice. in *Implementing Reproducible Research* (CRC Press) 149–183 (2014).
31. Brandl, G. & the Sphinx team. Sphinx - Python Documentation Generator, <http://www.sphinx-doc.org/en/master/> (2007).
32. Virtanen, P. et al. pydocweb: a tool for collaboratively documenting Python modules via the web. *Web Archive*, <https://code.google.com/archive/p/pydocweb/> (2008).
33. Harrington, J. The SciPy documentation project. In *Proceedings of the 7th Python in Science Conference* (eds G. Varoquaux, G., Vaught, T. & Millman, K. J.) 33–35 (2008).
34. van der Walt, S. The SciPy documentation project (technical overview). In *Proceedings of the 7th Python in Science Conference* (eds G. Varoquaux, G., Vaught, T. & Millman, K. J.) 27–28 (2008).
35. Harrington, J. & Goldsmith, D. Progress report: NumPy and SciPy documentation in 2009. In *Proceedings of the 8th Python in Science Conference* (eds Varoquaux, G., van der Walt, S. & Millman, K. J.) 84–87 (2009).
36. Pérez, F., Langtangen, H. P. & LeVeque, R. Python for scientific computing. In *SIAM Conference on Computational Science and Engineering* **42** (5) (2009).
37. Dubois, P. EPython: batteries included. *Comput. Sci. Eng.* **9**, 7–9 (2007).
38. Millman, K. J. & Aivazis, M. Python for scientists and engineers. *Comput. Sci. Eng.* **13**, 9–12 (2011).
39. Pérez, F., Granger, B. E. & Hunter, J. D. Python: an ecosystem for scientific computing. *Comput. Sci. Eng.* **13**, 13–21 (2011).
40. Behnel, S. et al. Cython: the best of both worlds. *Comput. Sci. Eng.* **13**, 31–39 (2011).
41. Ramachandran, P. & Varoquaux, G. Mayavi: 3D visualization of scientific data. *Comput. Sci. Eng.* **13**, 40–51 (2011).
42. Müller, E. et al. Python in neuroscience. *Front. Neuroinform.* **9**, 11 (2015).
43. GitHub. Network dependents - scipy/scipy, <https://github.com/scipy/scipy/network/dependents> (2019).
44. Boisvert, R. F., Howe, S. E. & Kahaner, D. K. The guide to available mathematical software problem classification system. *Commun. Stat. Simul. Comput.* **20**, 811–842 (1991).
45. Seabold, S. & Perktold, J. Statsmodels: econometric and statistical modeling with Python. In *Proceedings of the 9th Python in Science Conference* 57–61 (2010).
46. Salvatier, J., Wiecki, T. V. & Fonnesbeck, C. Probabilistic programming in Python using PyMC3. *PeerJ Comput. Sci.* **2**, e55 (2016).
47. Foreman-Mackey, D., Hogg, D. W., Lang, D. & Goodman, J. emcee: the MCMC hammer. *Publ. Astron. Soc. Pac.* **125**, 306–312 (2013).
48. Meurer, A. et al. SymPy: symbolic computing in Python. *PeerJ Comput. Sci.* **3**, e103 (2017).
49. Hagberg, A. A., Schult, D. A. & Swart, P. J. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference*. (eds G. Varoquaux, G., Vaught, T. & Millman, K. J.) 11–15 (2008).
50. Koebel, C. H. & Zosel, M. E. *The High Performance FORTRAN Handbook* (MIT Press, 1993).
51. Piessens, R., de Doncker-Kapenga, E., Ueberhuber, C. W. & Kahaner, D. K. *QUADPACK: A Subroutine Package for Automatic Integration* (Springer, 1983).
52. Hindmarsh, A. C. ODEPACK, a systematized collection of ODE solvers. *Scientific Computing* 55–64 (1983).
53. Dierckx, P. *Curve and Surface Fitting with Splines* (Oxford Univ. Press, 1993).
54. Boggs, P. T., Byrd, R. H., Rogers, J. E. & Schnabel, R. B. *User's Reference Guide for ODRPACK Version 2.01: Software for Weight Orthogonal Distance Regression* (U.S. Department of Commerce, National Institute of Standards and Technology, 1992).
55. Moré, J. J., Garbow, B. S. & Hillstom, K. E. User guide for MINPACK-1. Report ANL-80-74 (Argonne National Laboratory, 1980).
56. Swartztrauber, P. N. Vectorizing the FFTs. In *Parallel Computations* (ed. Rodrigue, G.) 51–83 (Academic, 1982).
57. Swartztrauber, P. N. FFT algorithms for vector computers. *Parallel Comput.* **1**, 45–63 (1984).
58. Lehoucq, R. B., Sorensen, D. C. & Yang, C. ARPACK users' guide: solution of large scale eigenvalue problems with implicitly restarted Arnoldi methods. (Rice University, 1997).
59. Amos, D. E. Algorithm 644: A portable package for Bessel functions of a complex argument and nonnegative order. *ACM Trans. Math. Softw.* **12**, 265–273 (1986).
60. Brown, B., Lovato, J. & Russell, K. CDFLIB, https://people.sc.fsu.edu/~jburkardt/f_src/cdf/lib/cdflib.html (accessed 6 July 2018).
61. Kernighan, B. W. & Ritchie, D. M. *The C Programming Language* 2nd edn (Prentice Hall Professional Technical Reference, 1988).
62. Lenders, F., Kirches, C. & Potschka, A. trilb: a vector-free implementation of the GLTR method for iterative solution of the trust region problem. *Optim. Methods Softw.* **33**, 420–449 (2018).
63. Li, X. S. et al. SuperLU Users' Guide. Report LBNL-44289 (Lawrence Berkeley National Laboratory, 1999).
64. Li, X. S. An overview of SuperLU: algorithms, implementation, and user interface. *ACM Trans. Math. Softw.* **31**, 302–325 (2005).
65. Barber, C. B., Dobkin, D. P. & Huhdanpaa, H. The Quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.* **22**, 469–483 (1996).
66. Lam, S. K., Pitrou, A. & Seibert, S. Numba: A LLVM-based Python JIT compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC* 7:1–7:6 (ACM, 2015).
67. Bolz, C. F., Cuni, A., Fijalkowski, M. & Rigo, A. Tracing the meta-level: PyPy's tracing JIT compiler. In *Proceedings of the 4th Workshop on the Implementation, Compilation, Optimization of Object-Oriented Languages and Programming Systems* 18–25 (ACM, 2009).
68. VanderPlas, J. Benchmarking nearest neighbor searches in Python, <https://jakevdp.github.io/blog/2013/04/29/benchmarking-nearest-neighbor-searches-in-python/> (19 April 2013).
69. Maneewongvatana, S. & Mount, D. M. Analysis of approximate nearest neighbor searching with clustered point sets. Preprint at <https://arxiv.org/pdf/cs/9901013.pdf> (1999).
70. Molden, S. ENH: Enhancements to spatial.cKDTree, <https://github.com/scipy/scipy/pull/4374/> (7 January 2015).
71. Aspnas, M., Signell, A. & Westerholm, J. Efficient assembly of sparse matrices using hashing. In *Applied Parallel Computing. State of the Art in Scientific Computing* (eds Kågström, B. et al.) 900–907 (Springer, 2007).
72. Cormen, T. H., Stein, C., Rivest, R. L. & Leiserson, C. E. *Introduction to Algorithms* 2nd edn (McGraw-Hill Higher Education, 2001).
73. Moore, A. W. et al. Fast algorithms and efficient statistics: N-point correlation functions. In *Mining the Sky. ESO Astrophysics Symposia (European Southern Observatory)* (eds Banday, A. J., Zaroubi, S. & Bartelmann, M.) 71–82 (Springer, 2001).

NATURE METHODS

PERSPECTIVE

74. Feng, Y. ENH: Faster count_neighbour in cKDTree / + weighted input data <https://github.com/scipy/scipy/pull/5647> (2015).
75. Martin, A. M., Giovanelli, R., Haynes, M. P. & Guzzo, L. The clustering characteristics of HI-selected galaxies from the 40% ALFALFA survey. *Astrophys. J.* **750**, 38 (2012).
76. Anderson, E. et al. *LAPACK Users' Guide* 3rd edn (Society for Industrial and Applied Mathematics, 1999).
77. Henriksen, I. Circumventing the linker: using SciPy's BLAS and LAPACK within Cython. In *Proceedings of the 14th Python in Science Conference (SciPy 2015)* (eds Huff, K. & Bergstra, J.) 49–52 (2015).
78. Andersen, E. D. & Andersen, K. D. (2000) The Mosek interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High Performance Optimization* 197–232 (Springer, 2000).
79. The NETLIB LP test problem set, <http://www.numerical.rl.ac.uk/cute/netlib.html> (2019).
80. Andersen, E. D. & Andersen, K. D. Presolving in linear programming. *Math. Program.* **71**, 221–245 (1995).
81. Wormington, M., Panaccione, C., Matney Kevin, M. & Bowen, D. K. Characterization of structures from X-ray scattering data using genetic algorithms. *Philos. Trans. R. Soc. Lond. A* **357**, 2827–2848 (1999).
82. Storn, R. & Price, K. Differential evolution — a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **11**, 341–359 (1997).
83. Griffiths, T. L. & Steyvers, M. Finding scientific topics. *Proc. Natl Acad. Sci. USA* **101**(Suppl. 1), 5228–5235 (2004).
84. Dierckx, P. *Curve and Surface Fitting with Splines* (Oxford Univ. Press, 1993).
85. Virtanen, P. ENH: interpolate: rewrite pform evaluation in Cython, <https://github.com/scipy/scipy/pull/2885> (2013).
86. Burovski, E. add b-splines, <https://github.com/scipy/scipy/pull/3174> (27 December 2013).
87. de Boor, C. *A Practical Guide to Splines* (Springer, 1978).
88. Mayorov, N. ENH: CubicSpline interpolator, <https://github.com/scipy/scipy/pull/5653> (2 January 2016).
89. Fritsch, F. N. & Carlson, R. E. Monotone piecewise cubic interpolation. *SIAM J. Numer. Anal.* **17**, 238–246 (1980).
90. Akima, H. A new method of interpolation and smooth curve fitting based on local procedures. *J. Assoc. Comput. Mach.* **17**, 589–602 (1970).
91. Beck, K. *Test-driven Development: By Example* (Addison-Wesley, 2003).
92. Silver, A. Collaborative software development made easy. *Nature* **550**, 143–144 (2017).
93. Eghbal, N. *Roads and Bridges: The Unseen Labor Behind Our Digital Infrastructure* (Ford Foundation, 2016).
94. The Astropy Collaboration et al. The Astropy Project: building an open-science project and status of the v2.0 core package. *Astron. J.* **156**, 123 (2018).
95. Lev, O., Dufresne, J., Kasim, R., Skinn, B. & Wilk, J. pypinfo: view PyPI download statistics with ease, <https://github.com/ofek/pypinfo> (2018).
96. Abbott, B. P. et al. Observation of gravitational waves from a binary black hole merger. *Phys. Rev. Lett.* **116**, 061102 (2016).
97. David Liu. The Intel distribution for Python, <https://software.intel.com/en-us/articles/intel-optimized-packages-for-the-intel-distribution-for-python> (25 August 2017, updated 30 October 2017, accessed 25 July 2018).
98. Nelder, J. A. & Mead, R. A simplex method for function minimization. *Comput. J.* **7**, 308–313 (1965).
99. Wright, M. H. Direct search methods: once scorned, now respectable. *Pitman Research Notes in Mathematics Series* 191–208 (1996).
100. Powell, M. J. D. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *Comput. J.* **7**, 155–162 (1964).
101. Powell, M. J. D. A direct search optimization method that models the objective and constraint functions by linear interpolation. In *Advances in Optimization and Numerical Analysis* (eds Gomez, S. & Hennart, J. P.) 51–67 (Springer, 1994).
102. Powell, M. J. D. Direct search algorithms for optimization calculations. *Acta Numerica* **7**, 287–336 (1998).
103. Powell, M. J. D. A view of algorithms for optimization without derivatives. *Math. Today Bull. Inst. Math. Appl.* **43**, 170–174 (2007).
104. Polak, E. & Ribiere, G. Note sur la convergence de methodes de directions conjuguées. *Rev. française d'informatique et. de. Rech. opérationnelle* **3**, 35–43 (1969).
105. Nocedal, J. & Wright, S. *Numerical Optimization* 2nd edn (Springer Science & Business Media, 2006).
106. Byrd, R. H., Lu, P., Nocedal, J. & Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.* **16**, 1190–1208 (1995).
107. Zhu, C., Byrd, R. H., Lu, P. & Nocedal, J. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.* **23**, 550–560 (1997).
108. Schittkowski, K. On the convergence of a sequential quadratic programming method with an augmented Lagrangian line search function. *Mathematische Operationsforschung und Statistik. Ser. Optim.* **14**, 197–216 (1983).
109. Schittkowski, K. The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function. Part 2: an efficient implementation with linear least squares subproblems. *Numer. Math.* **38**, 115–127 (1982).
110. Schittkowski, K. The nonlinear programming method of Wilson, Han, and Powell with an augmented Lagrangian type line search function. Part 1: convergence analysis. *Numer. Math.* **38**, 83–114 (1982).
111. Kraft, D. A software package for sequential quadratic programming. Report DFVLR-FR 88–28 (Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, 1988).
112. Nash, S. G. Newton-type minimization via the Lanczos method. *SIAM J. Numer. Anal.* **21**, 770–788 (1984).
113. Powell, M. J. D. A new algorithm for unconstrained optimization. *Nonlinear Programming* 31–65 (1970).
114. Steihaug, T. The conjugate gradient method and trust regions in large scale optimization. *SIAM J. Numer. Anal.* **20**, 626–637 (1983).
115. Conn, A. R., Gould, N. I. M. & Toint, P. L. *Trust Region Methods* (SIAM, 2000).
116. Moré, J. J. & Sorensen, D. C. Computing a trust region step. *SIAM J. Sci. Statist. Comput.* **4**, 553–572 (1983).
117. Gould, N. I. M., Lucidi, S., Roma, M. & Toint, P. L. Solving the trust-region subproblem using the Lanczos method. *SIAM J. Optim.* **9**, 504–525 (1999).
118. Abbasi, H. Sparse: a more modern sparse array library. In *Proceedings of the 17th Python in Science Conference* (eds Akici, F. et al.) 27–30 (2018).
119. Mohr, P. J., Newell, D. B. & Taylor, B. N. CODATA recommended values of the fundamental physical constants: 2014. *J. Phys. Chem. Ref. Data* **45**, 043102 (2016).
120. Boisvert, R. F., Pozo, R., Remington, K., Barrett, R. F. & Dongarra, J. J. Matrix Market: a web resource for test matrix collections. In *Quality of Numerical Software* 125–137 (Springer, 1997).
121. Rew, R. & Davis, G. NetCDF: an interface for scientific data access. *IEEE Comput. Graph. Appl.* **10**, 76–82 (1990).
122. Duff, I. S., Grimes, R. G. & Lewis, J. G. Users' guide for the Harwell-Boeing sparse matrix collection (release 1), <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.41.8922> (1992).

Acknowledgements

We thank everyone who has contributed to SciPy 1.0, from those who have posted one comment about an issue through those who have made several small patches and beyond.

Author contributions

P.V. and R.G. led the development of SciPy for over 10 years. T.E.O., E.J. and P.P. created SciPy. T.R. and M.H. composed the manuscript with input from others. Other named authors are SciPy core developers. All authors have contributed significant code, documentation and/or expertise to the SciPy project. All authors reviewed the manuscript.

Competing interests

The following statements indicate industry affiliations for authors in the main author list, but not for authors in the SciPy 1.0 Contributor group beyond that. These affiliations may have since changed. R.G. was employed by Quansight LLC. T.E.O., E.J. and R.K. were employed by Enthought, Inc. T.E.O. and I.H. were employed by Anaconda Inc. N.M. was employed by WayRay LLC. E.W.M. was employed by Bruker Biospin Corp. F.P. and P.v.M. were employed by Google LLC.

Additional information

Supplementary information is available for this paper at <https://doi.org/10.1038/s41592-019-0686-2>.

Correspondence should be addressed to R.G., M.H. or T.R.

Peer review information Rita Strack was the primary editor on this article and managed its editorial process and peer review in collaboration with the rest of the editorial team.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Open Access This article is licensed under a Creative Commons

Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.
© The Author(s) 2020

PERSPECTIVE

NATURE METHODS

SciPy 1.0 Contributors

Aditya Vijaykumar^{40,41}, Alessandro Pietro Bardelli⁴², Alex Rothberg¹⁵, Andreas Hilboll⁴³,
 Andreas Kloeckner⁴⁴, Anthony Scopatz², Antony Lee⁴⁵, Ariel Rokem⁴⁶, C. Nathan Woods⁹,
 Chad Fulton⁴⁷, Charles Masson⁴⁸, Christian Häggström⁴⁹, Clark Fitzgerald⁵⁰, David A. Nicholson⁵¹,
 David R. Hagen⁵², Dmitrii V. Pasechnik⁵³, Emanuele Olivetti⁵⁴, Eric Martin⁵⁵, Eric Wieser⁵⁶,
 Fabrice Silva⁵⁷, Felix Lenders^{58,59,60}, Florian Wilhelm⁶¹, G. Young¹⁷, Gavin A. Price⁶², Gert-Ludwig Ingold⁶³,
 Gregory E. Allen⁶⁴, Gregory R. Lee^{65,66}, Hervé Audren⁶⁷, Irvin Probst⁶⁸, Jörg P. Dietrich^{69,70},
 Jacob Silterra⁷¹, James T Webber⁷², Janko Slavič⁷³, Joel Nothman⁷⁴, Johannes Buchner^{75,76},
 Johannes Kulick⁷⁷, Johannes L. Schönberger¹⁵, José Vinícius de Miranda Cardoso⁷⁸, Joscha Reimer⁷⁹,
 Joseph Harrington⁸⁰, Juan Luis Cano Rodríguez⁸¹, Juan Nunez-Iglesias⁸², Justin Kuczynski⁸³,
 Kevin Tritz⁸⁴, Martin Thoma⁸⁵, Matthew Newville⁸⁶, Matthias Kümmerer⁸⁷, Maximilian Bolingbroke⁸⁸,
 Michael Tartre⁸⁹, Mikhail Pak⁹⁰, Nathaniel J. Smith⁹¹, Nikolai Nowaczyk⁹², Nikolay Shebanov⁹³,
 Oleksandr Pavlyk⁹⁴, Per A. Brodtkorb⁹⁵, Perry Lee⁹⁶, Robert T. McGibbon⁹⁷, Roman Feldbauer⁹⁸,
 Sam Lewis⁹⁹, Sam Tygier¹⁰⁰, Scott Sievert¹⁰¹, Sebastiano Vigna¹⁰², Stefan Peterson¹⁵, Surhud More^{103,104},
 Tadeusz Pudlik¹⁰⁵, Takuya Oshima¹⁰⁶, Thomas J. Pingel¹⁰⁷, Thomas P. Robitaille¹⁰⁸, Thomas Spura¹⁰⁹,
 Thouis R. Jones¹¹⁰, Tim Cera¹¹¹, Tim Leslie¹⁵, Tiziano Zito¹¹², Tom Krauss¹¹³, Utkarsh Upadhyay¹¹⁴,
 Yaroslav O. Halchenko¹¹⁵ and Yoshiki Vázquez-Baeza¹¹⁶

⁴⁰International Centre for Theoretical Sciences, Tata Institute of Fundamental Research, Bengaluru, India. ⁴¹Department of Physics, Birla Institute of Technology and Science, Pilani, India. ⁴²Independent researcher, Milan, Italy. ⁴³Institute of Environmental Physics, University of Bremen, Bremen, Germany. ⁴⁴Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, USA. ⁴⁵Laboratoire Photonique, Numérique et Nanosciences UMR 5298, Université de Bordeaux, Institut d'Optique Graduate School, CNRS, Talence, France. ⁴⁶The University of Washington eScience Institute, The University of Washington, Seattle, WA, USA. ⁴⁷Federal Reserve Board of Governors, Washington, DC, USA. ⁴⁸Datadog Inc., New York, NY, USA. ⁴⁹HQ, Orexplora, Stockholm, Sweden. ⁵⁰Statistics Department, University of California - Davis, Davis, CA, USA. ⁵¹Emory University, Atlanta, GA, USA. ⁵²Applied BioMath, Concord, MA, USA. ⁵³Department of Computer Science, University of Oxford, Oxford, UK. ⁵⁴Neuroinformatics Laboratory, Bruno Kessler Foundation, Trento, Italy. ⁵⁵Independent researcher, Chicago, IL, USA. ⁵⁶Department of Engineering, University of Cambridge, Cambridge, UK. ⁵⁷Aix Marseille Univ, CNRS, Centrale Marseille, LMA, Marseille, France. ⁵⁸Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University, Heidelberg, Germany. ⁵⁹ABB Corporate Research, ABB AG, Ladenburg, Germany. ⁶⁰Institut für Mathematische Optimierung, Technische Universität Carolo-Wilhelmina zu Braunschweig, Braunschweig, Germany. ⁶¹Independent researcher, Cologne, Germany. ⁶²Lawrence Berkeley National Laboratory, Berkeley, CA, USA. ⁶³Institut für Physik, Universität Augsburg, Augsburg, Germany. ⁶⁴Applied Research Laboratories, The University of Texas at Austin, Austin, TX, USA. ⁶⁵Department of Radiology, School of Medicine, University of Cincinnati, Cincinnati, OH, USA. ⁶⁶Department of Radiology, Cincinnati Children's Hospital Medical Center, Cincinnati, OH, USA. ⁶⁷Ascent Robotics Inc., Tokyo, Japan. ⁶⁸ENSTA Bretagne, Brest, France. ⁶⁹Faculty of Physics, Ludwig-Maximilians-Universität, München, Germany. ⁷⁰Excellence Cluster Universe, München, Germany. ⁷¹Independent researcher, Malden, Massachusetts, USA. ⁷²Data Sciences, Chan Zuckerberg Biohub, San Francisco, CA, USA. ⁷³Faculty of Mechanical Engineering, University of Ljubljana, Ljubljana, Slovenia. ⁷⁴Sydney Informatics Hub, The University of Sydney, Camperdown, NSW, Australia. ⁷⁵Instituto de Astrofísica, Pontificia Universidad Católica de Chile, Santiago, Chile. ⁷⁶Max Planck Institute for Extraterrestrial Physics, Garching, Germany. ⁷⁷University of Stuttgart, Machine Learning and Robotics Lab, Stuttgart, Germany. ⁷⁸Department of Electrical Engineering, Universidade Federal de Campina Grande, Campina Grande, Brazil. ⁷⁹Department of Computer Science, Kiel University, Kiel, Germany. ⁸⁰Planetary Sciences Group and Florida Space Institute and Department of Physics, University of Central Florida, Orlando, FL, USA. ⁸¹Independent researcher, Madrid, Spain. ⁸²Monash Micro Imaging, Monash University, Clayton, VIC, Australia. ⁸³Department of Molecular, Cellular, and Developmental Biology, University of Colorado, Boulder, Boulder, CO, USA. ⁸⁴Department of Physics and Astronomy, Johns Hopkins University, Baltimore, MD, USA. ⁸⁵Independent researcher, Munich, Germany. ⁸⁶Center for Advanced Radiation Sources, The University of Chicago, Chicago, IL, USA. ⁸⁷University of Tübingen, Tübingen, Germany. ⁸⁸Independent researcher, Rugby, UK. ⁸⁹Two Sigma Investments, New York, NY, USA. ⁹⁰Department of Mechanical Engineering, Technical University of Munich, Garching, Germany. ⁹¹Independent researcher, Berkeley, CA, USA. ⁹²Independent researcher, London, UK. ⁹³Independent researcher, Berlin, Germany. ⁹⁴Intel Corp., Austin, TX, USA. ⁹⁵Independent Researcher, Horten, Norway. ⁹⁶Independent researcher, Daly City, CA, USA. ⁹⁷D. E. Shaw Research, New York, NY, USA. ⁹⁸Division of Computational Systems Biology, Department of Microbiology and Ecosystem Science, University of Vienna, Vienna, Austria. ⁹⁹Independent researcher, Melbourne, Australia. ¹⁰⁰School of Physics and Astronomy, University of Manchester, Manchester, UK. ¹⁰¹Electrical and Computer Engineering, University of Wisconsin-Madison, Madison, WI, USA. ¹⁰²Dipartimento di Informatica, Università degli Studi di Milano, Milan, Italy. ¹⁰³Inter-University Centre for Astronomy and Astrophysics, Ganeshkhind, Pune, India. ¹⁰⁴Kavli Institute for the Physics and Mathematics of the Universe, Kashiwa-shi, Japan. ¹⁰⁵Waymo LLC, Mountain View, CA, USA. ¹⁰⁶Faculty of Engineering, Niigata University, Nishi-ku, Niigata, Japan. ¹⁰⁷Virginia Polytechnic Institute and State University, Blacksburg, VA, USA. ¹⁰⁸Aperio Software, Headingley Enterprise and Arts Centre, Leeds, UK. ¹⁰⁹Independent researcher, Duisburg, Germany. ¹¹⁰Broad Institute, Cambridge, MA, USA. ¹¹¹Independent researcher, Gainesville, FL, USA. ¹¹²Department of Psychology, Humboldt University of Berlin, Berlin, Germany. ¹¹³Epiq Solutions, Schaumburg, IL, USA. ¹¹⁴Max Planck Institute for Software Systems, Kaiserslautern, Germany. ¹¹⁵Department of Psychology and Brain Sciences, Dartmouth College, Hanover, NH, USA. ¹¹⁶Jacobs School of Engineering, University of California San Diego, La Jolla, CA, USA.

Reporting Summary

Nature Research wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Research policies, see [Authors & Referees](#) and the [Editorial Policy Checklist](#).

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a Confirmed

- ☒ ☐ The exact sample size (n) for each experimental group/condition, given as a discrete number and unit of measurement
- ☒ ☐ A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
- ☒ ☐ The statistical test(s) used AND whether they are one- or two-sided
Only common tests should be described solely by name; describe more complex techniques in the Methods section.
- ☒ ☐ A description of all covariates tested
- ☒ ☐ A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
- ☒ ☐ A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
- ☒ ☐ For null hypothesis testing, the test statistic (e.g. F , t , r) with confidence intervals, effect sizes, degrees of freedom and P value noted
Give P values as exact values whenever suitable.
- ☒ ☐ For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
- ☒ ☐ For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
- ☒ ☐ Estimates of effect sizes (e.g. Cohen's d , Pearson's r), indicating how they were calculated

Our web collection on [statistics for biologists](#) contains articles on many of the points above.

Software and code

Policy information about [availability of computer code](#)

Data collection	Language choice data was collected using the linguist library, and in particular using a special feature branch of this project that is openly available: https://github.com/github/linguist/pull/4181 Data for Figure 2 was collected using the pytest-cov and gcov packages; Docker was used to manage containers & the exact source code used is openly available: https://github.com/tylerjereddy/scipy-cov-track Data for figure 3 was collected using the Airspeed Velocity (asv version 0.4) package--see their openly available repository: https://github.com/airspeed-velocity/asv
Data analysis	NA

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors/reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Research [guidelines for submitting code & software](#) for further information.

Data

Policy information about [availability of data](#)

All manuscripts must include a [data availability statement](#). This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A list of figures that have associated raw data
- A description of any restrictions on data availability

All SciPy library source code and most data generated for the current study are available in the SciPy GitHub repository, <https://github.com/scipy>. Some supporting code and data have also been stored in other public repositories cited by this manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☐ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see [nature.com/documents/nr-reporting-summary-flat.pdf](https://www.nature.com/documents/nr-reporting-summary-flat.pdf)

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	Describe how sample size was determined, detailing any statistical methods used to predetermine sample size OR if no sample-size calculation was performed, describe how sample sizes were chosen and provide a rationale for why these sample sizes are sufficient.
Data exclusions	Describe any data exclusions. If no data were excluded from the analyses, state so OR if data were excluded, describe the exclusions and the rationale behind them, indicating whether exclusion criteria were pre-established.
Replication	Describe the measures taken to verify the reproducibility of the experimental findings. If all attempts at replication were successful, confirm this OR if there are any findings that were not replicated or cannot be reproduced, note this and describe why.
Randomization	Describe how samples/organisms/participants were allocated into experimental groups. If allocation was not random, describe how covariates were controlled OR if this is not relevant to your study, explain why.
Blinding	Describe whether the investigators were blinded to group allocation during data collection and/or analysis. If blinding was not possible, describe why OR explain why blinding was not relevant to your study.

Behavioural & social sciences study design

All studies must disclose on these points even when the disclosure is negative.

Study description	Briefly describe the study type including whether data are quantitative, qualitative, or mixed-methods (e.g. qualitative cross-sectional, quantitative experimental, mixed-methods case study).
Research sample	State the research sample (e.g. Harvard university undergraduates, villagers in rural India) and provide relevant demographic information (e.g. age, sex) and indicate whether the sample is representative. Provide a rationale for the study sample chosen. For studies involving existing datasets, please describe the dataset and source.
Sampling strategy	Describe the sampling procedure (e.g. random, snowball, stratified, convenience). Describe the statistical methods that were used to predetermine sample size OR if no sample-size calculation was performed, describe how sample sizes were chosen and provide a rationale for why these sample sizes are sufficient. For qualitative data, please indicate whether data saturation was considered, and what criteria were used to decide that no further sampling was needed.
Data collection	Provide details about the data collection procedure, including the instruments or devices used to record the data (e.g. pen and paper, computer, eye tracker, video or audio equipment) whether anyone was present besides the participant(s) and the researcher, and whether the researcher was blind to experimental condition and/or the study hypothesis during data collection.
Timing	Indicate the start and stop dates of data collection. If there is a gap between collection periods, state the dates for each sample cohort.
Data exclusions	If no data were excluded from the analyses, state so OR if data were excluded, provide the exact number of exclusions and the rationale behind them, indicating whether exclusion criteria were pre-established.
Non-participation	State how many participants dropped out/declined participation and the reason(s) given OR provide response rate OR state that no participants dropped out/declined participation.
Randomization	If participants were not allocated into experimental groups, state so OR describe how participants were allocated to groups, and if allocation was not random, describe how covariates were controlled.

Ecological, evolutionary & environmental sciences study design

All studies must disclose on these points even when the disclosure is negative.

Study description	Briefly describe the study. For quantitative data include treatment factors and interactions, design structure (e.g. factorial, nested, hierarchical), nature and number of experimental units and replicates.
-------------------	--

Research sample	Describe the research sample (e.g. a group of tagged <i>Passer domesticus</i> , all <i>Stenocereus thurberi</i> within Organ Pipe Cactus National Monument), and provide a rationale for the sample choice. When relevant, describe the organism taxa, source, sex, age range and any manipulations. State what population the sample is meant to represent when applicable. For studies involving existing datasets, describe the data and its source.
Sampling strategy	Note the sampling procedure. Describe the statistical methods that were used to predetermine sample size OR if no sample-size calculation was performed, describe how sample sizes were chosen and provide a rationale for why these sample sizes are sufficient.
Data collection	Describe the data collection procedure, including who recorded the data and how.
Timing and spatial scale	Indicate the start and stop dates of data collection, noting the frequency and periodicity of sampling and providing a rationale for these choices. If there is a gap between collection periods, state the dates for each sample cohort. Specify the spatial scale from which the data are taken
Data exclusions	If no data were excluded from the analyses, state so OR if data were excluded, describe the exclusions and the rationale behind them, indicating whether exclusion criteria were pre-established.
Reproducibility	Describe the measures taken to verify the reproducibility of experimental findings. For each experiment, note whether any attempts to repeat the experiment failed OR state that all attempts to repeat the experiment were successful.
Randomization	Describe how samples/organisms/participants were allocated into groups. If allocation was not random, describe how covariates were controlled. If this is not relevant to your study, explain why.
Blinding	Describe the extent of blinding used during data acquisition and analysis. If blinding was not possible, describe why OR explain why blinding was not relevant to your study.

Did the study involve field work? ☐ Yes ☒ No

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your research. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Included in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology
☒	☐ Animals and other organisms
☒	☐ Human research participants
☒	☐ Clinical data

Methods

n/a	Included in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging

A.2. Unpublished work

The author contributed patches to several open source software projects. Among these are many small, hardly noteworthy fixes, for example to eggNOG-mapper [262], BinSanity [263], Selene [167], or pyfaidx [264] (for details, see the author’s Github profile²). Contributions of higher effort are outlined in the sections below. Some of these tools had already been published in scientific journals prior to the author’s contribution, while others have not been formally published as of yet.

A.2.1. scikit-learn: machine learning in Python

The `scikit-learn` package is the defacto standard for machine learning in Python. At the time of writing (2020/03) it is used by nearly 100 000 public projects on GitHub.³ All work packages described in this thesis make heavy use of functionality provided by scikit-learn. On several occasions, limitations or bugs on certain corner cases were detected. Six pull requests (PRs) stemming from these observations have been merged into upstream scikit-learn, with additional ones likely to follow.⁴ Most changes are a direct consequence of the close investigation of the `sklearn.neighbors` package during the development of scikit-hubness. Five PRs improve the test suite of scikit-learn. For example, several test assertions were erroneously never executed, which could have led to false positive test results. These changes make scikit-learn more robust to future changes on the code base.

A.2.2. thunderSVM: fast support vector machines on CPUs and GPUs

ThunderSVM provides highly efficient SVM implementations for CPUs and GPUs. While evaluating GPU support in `phenotrex` (see Appendix A.2.3) through ThunderSVM, certain bottlenecks in the library’s Python bindings were identified. The ThunderSVM core C++ and CUDA routines provide fast computation, especially for larger datasets. In case of replicated nested cross-validation on smaller datasets, most processor time was spent in Python code accessing data structures. Code analysis and profiling revealed that overhead by redundant type checks in list comprehensions slowed down the Python bindings. For example, instead of retrieving the support vectors in Python via list comprehensions (lists of support vectors), the complete matrix can be read from a buffer and accessed as a NumPy array. This avoids most type checks and improves performance on small datasets dramatically.

A.2.3. phenotrex: microbial phenotypic trait prediction

The multitude of complete genomes in public databases allows for machine learning-based microbial phenotypic trait prediction. Early attempts focused on class association rule mining (ARM) approaches, including the netcar [265] and PICA [266]

²Github profile of RF: <https://github.com/VarIr>

³scikit-learn dependents: <https://github.com/scikit-learn/scikit-learn/network/dependents>

⁴Contributions by RF: <https://github.com/scikit-learn/scikit-learn/pulls?q=author%3AVarIr>, all accessed 2020/05/28

packages. The author investigated these approaches in his Master thesis, and improved the support vector machine functionality of PICA [267]. Development had subsequently been continued by other members of the Division of Computational Systems Biology (CUBE), at the University of Vienna. PICA is the work horse of the PhenDB pipeline [217], which currently provides predictions for more than 70 different microbial traits.

In 2019, a PICA re-implementation project was started, due to obsolete functionality, difficult-to-maintain code, and outdated dependencies. The new software package **phenotrex**⁵ is being developed with best practices in mind. It features many improvements over the previous solution:

- Builds upon the battle-tested SciPy stack (**scikit-learn**, **matplotlib**, etc.)
- More easily extensible due to careful software architecture design decisions
- New powerful classifiers (e.g., XGboost [268])
- Dropped obsolete functions (e.g., ARM) for a lean, easy-to-maintain code base
- Inbuilt performance estimation for incomplete and/or contaminated (metagenomic assembled) genomes
- Model interpretability: Shapley additive explanations (SHAP, [269]) reveal contributions of individual genes towards specific trait predictions at inference time
- Improved computational efficiency (e.g., faster linear support vector machines with LIBLINEAR [270])
- Test-driven development with high code coverage and continuous integration on all major operating systems
- Code review procedures for all proposed changes
- Easy-to-use stand-alone program
 - Inbuilt gene calling with **prodigal** [271]
 - Integrates **deepnog** (cf. Chapter 6) for protein orthology inference
 - Trait prediction directly from FASTA files, without external requirements
- Supports Python 3.7 and higher
- Available from PyPI: `$ pip install phenotrex`

Development is led by Lukas L uftinger. The author contributes to the development, manages DevOps procedures, and performs thorough code review.

phenotrex is currently evaluated comprehensively, and will subsequently replace PICA in the PhenDB pipeline. An additional manuscript reporting on **phenotrex** and PhenDB is currently in preparation at the time of writing (June 2020).

⁵phenotrex: <https://github.com/univieCUBE/phenotrex>, accessed 2020/05/28

B. Additional projects

In the course of this research project stretching over more than four years, a number of opportunities for additional side projects, additional research ideas, and additional collaborations opened up.

This chapter features a non-exhaustive list of side-projects that have so far not been disseminated in terms of publications, software, or otherwise.

B.1. Protein unfolding

The Marlovits group at the Centre for Structural Systems Biology (CSSB) Hamburg studies the structural basis for assembly, regulation, and function of transmembrane molecular machines.

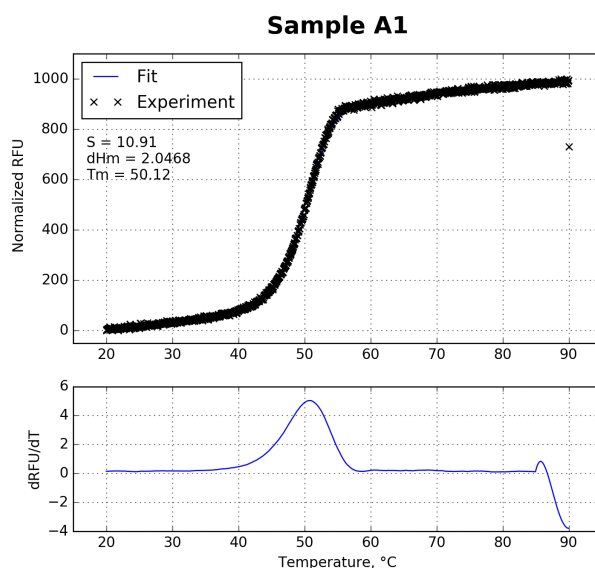


Figure B.1.: Prototypical protein unfolding Gibbs free energy curve. The top plot represents the experimental data and the sigmoidal fit (hidden by the data points). The lower plot shows the first derivative. The most informative part is the transition region, a steep rise of the signal in range 40-60°C. Two other components of the curve are called pre- and post-transition baselines.

The group investigates protein unfolding, which is required for example by the type III secretion system to transport proteins across membranes. Protein unfolding is also relevant for characterizing therapeutic monoclonal antibodies [272].

Unfolding experiments yield large numbers of Gibbs free energy time series [273]. Currently, manual inspection is required in order to filter out successful experiments as indicated by certain sigmoidal-like curves (see Figure B.1). This marks a limiting factor in the experimental setup.

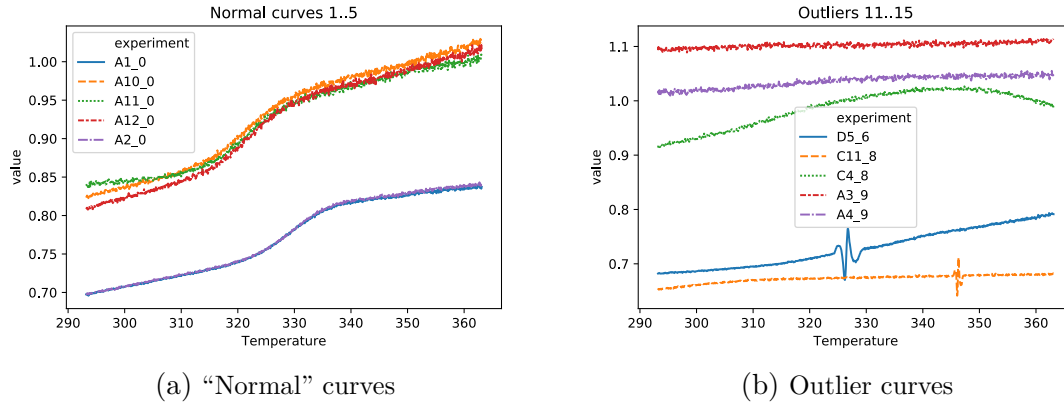


Figure B.2.: Several curves randomly picked from the test set. The “normal” curves clearly exhibit sigmoidal shape, while outliers often show linear patterns. Intermittent peaks indicate contamination with dust particles.

In order to increase experimental throughput, automatic machine learning-based pipelines for estimating result quality are investigated. Initial experiments considered fully unsupervised methods of outlier detection. Given that most curves in the training data set indeed follow the desired sigmoidal pattern, low quality results should be identifiable as outliers with respect to some measure of (dis-)similarity between the curves.

An exploratory pipeline of outlier detection was set up:

1. Measurements are interpolated in order to obtain a common, regular temperature grid for all unfolding experiments.
2. Curves are trimmed slightly on the left and the right to remove fluctuation artifacts impairing the following steps.
3. Dynamic time warping between all pairs of curves provides a dissimilarity matrix (similar to Smith-Waterman alignments of biological sequences).
4. Local outlier factors (LOF) are computed for the final outlier detection, that is, detection of low-quality curves.

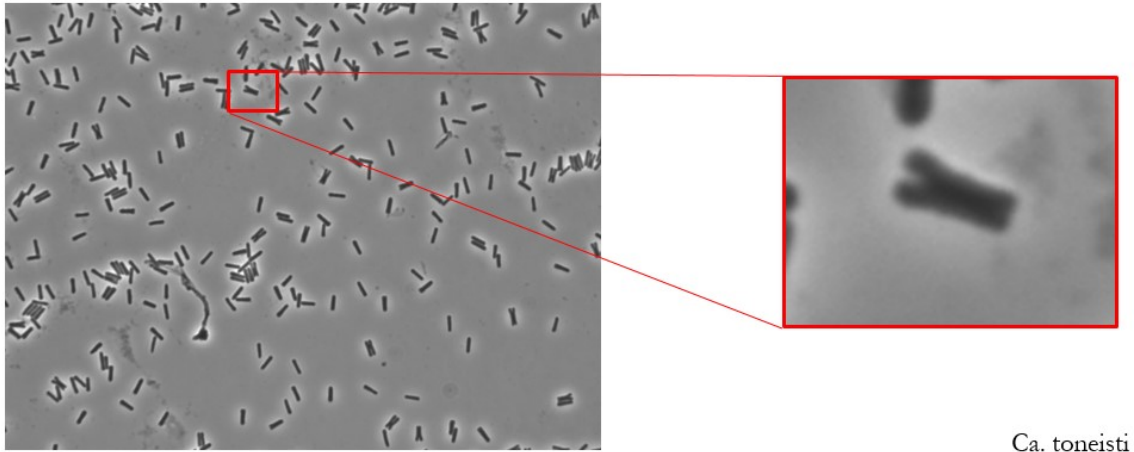
The resulting partition into high-quality and low-quality curves/experiments was qualitatively examined by the collaboration partner, and deemed highly promising (see Figure B.2 for exemplary curves). Further improvements of the pipeline are expected when labeled training data become available, and supervised algorithms can be applied to estimate curve quality. These approaches will be evaluated quantitatively as well. The project will continue when the collaboration partner provides the labeled training data.

Collaboration partners:

- Vadim Kotov, Thomas Marlovits (CSSB Hamburg, Germany).

B.2. Microbial longitudinal division

Prokaryotic species exhibit a wide range of morphological features, including spherical appearance, or star, rod, or spiral shapes. Cell growth and division patterns also



Ca. toneisti

Figure B.3.: Longitudinally dividing bacteria

differ between organisms: A vast majority of rod-shaped bacteria are known to grow in length and divide perpendicularly to growth direction, for example, Gammaproteobacteria such as *Escherichia coli*. Other bacterial species, however, have been identified to divide longitudinally, including the symbiotic *Thiosymbion* [274].

The Bulgheresi group at the Faculty of Life Sciences, University of Vienna, researches properties and life style of longitudinally dividing microbes. Currently only a few species have been recognized to divide longitudinally. There might be, however, a number of additional known species, which have not yet been identified as users of this particular mode of division, as this phenomenon has only been first described in 2012 [275].

We hypothesize, that additional longitudinally dividing species have already been described in literature. In order to identify these microbes, we develop a method for the automatic screening of microscopical images, filtering out images of microbes potentially dividing longitudinally. This method can then be used as a high-throughput hypothesis generator, screening through a vast number of publicly available microscopic images of prokaryotes (for an example, see Figure B.3), and providing candidates for detailed wet-lab characterization.

State-of-the-art computer vision has to a large extent relied on deep learning since the ImageNet Large Scale Visual Recognition Challenge 2012 and the introduction of AlexNet [276]. We finetune a deep residual convolutional network (ResNet) [156] pretrained on the ImageNet classification task as a binary classifier: “Longitudinal division” versus “other division”. The training data set (provided by the Ecogenomics group, see below) must remain confidential until the corresponding manuscript has been released to the public. Images are first segmented to cut patches of single (dividing) bacteria (performed by Helmholtz group). Segments are augmented by various strategies (black/white colors, random resizing, cropping, rotation, mirroring, and others) before being processed by the deep network. Classification performance is estimated in a leave-one-species-out scheme, that is, a model is evaluated on a test set comprising all images of species A, after being trained on the images of all species except A. This procedure is repeated, so that each species constitutes the test set once, and results are averaged over folds. Primary interest is on precision, as to not flood downstream analysis with a high number of false positives. Models are inspected by class activation maps (CAM) [277], which employ global average pool-

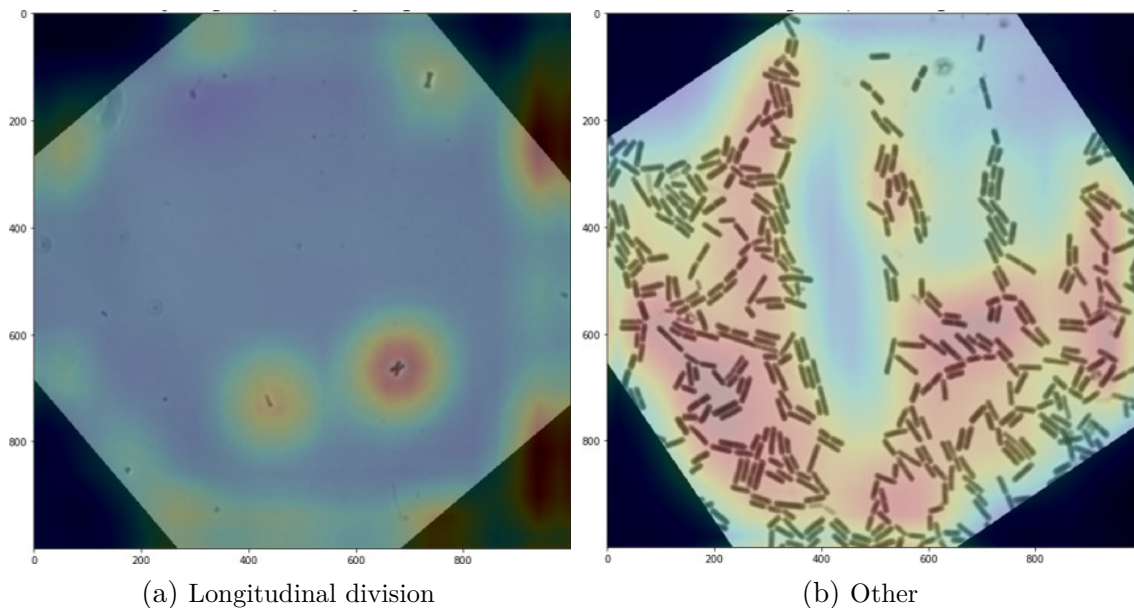


Figure B.4.: True positives from the test set. Red patches indicate discriminatory regions. While the network has learned to focus on microbes, it clearly also picks up confounding cues at image borders, possibly hinting at a “clever Hans” [278].

ing in order to locate discriminatory regions for particular categories (for instance, “longitudinal division”). Figure B.4 shows correctly classified test images overlaid with the corresponding CAMs. This particular model was trained on full images and apparently picks up spurious correlations in addition to reasonable cues. These observations among others motivated the additional segmentation step before training the classifier. Nevertheless, the pipeline achieves precision values > 0.9 even without segmentation preprocessing. (Deep learning related work performed by the author).

This project is continuing at the time of writing. Integration of subroutines is pending, after which a final model will be trained. An inference module will be set up, so that image databases can be screened. In parallel, a manuscript is currently being drafted.

Collaboration partners:

- Philipp Weber, Gabriela Paredes, Silvia Bulgheresi (Archaea Biology and Ecogenomics Division, University of Vienna).
- Javier Fernandez Betel Geijo (Division of Computational Systems Biology, University of Vienna).
- Keichi Ito, Carlos García, Wolfgang zu Castell (Scientific Computing Research Unit, Helmholtz Zentrum München, Germany).
- Students of “SE Applied machine learning for biological problems”(W2019, see also Appendix C) evaluated various augmentation strategies.

C. Academic teaching

An academic career does not only entail research and knowledge creation, but also comprises proliferating what has been learned. There are numerous ways to achieve this goal, from informal coffee-break talks with colleagues (which probably constitute a significant fraction of all exchanged ideas in academia), to answering questions on platforms like StackExchange¹, and engaging with the society, thus spreading the word more widely than in the scientific community to embiggen impact.

Higher educational teaching is arguably the most important aspect of academic knowledge sharing. That is, training the next generation of scientists, as well as highly qualified employees, and curious citizens in general.

To that end, the author engaged early in the ‘active’ part of higher education, while still being a member of the ‘passive’ part at the same time as a PhD student.

- Spring 2019: Supervision of Axinya Tokareva’s Bachelor thesis in the context of the *frenetiq* project about protein sequence embedding with deep learning.
- Summer 2019: Supervision of Lukas Gosch’s software development project about protein orthologous group prediction in the *DeepNOG* project. Both students were eventually graded as excellent (“Sehr gut”).
- Fall 2019/Winter 2020: Teaching the seminar *Applied machine learning for biological problems* at the University of Vienna, a non-compulsory course for Master students of various backgrounds, overbooked by 200%, and well-received by the students, according to the course evaluation. It is not without pride, when the author presents the following exemplary statement:

“Fazit: Danke! Einer der besten Kurse, an denen ich bislang an Unis teilgenommen habe (und das waren nicht wenige, weil 3 abgeschlossene Studien) oder die ich im privatwirtschaftlichen Umfeld absolviert habe. Großes Lob von meiner Seite!”

— Anonymous student, *Course evaluation*^a

^aTranslation by <https://www.deepl.com>: “Conclusion: Thank you! One of the best courses I have attended at universities (and they were not few, because 3 completed studies) or which I have completed in a private sector environment. Great praise from my side!”

¹The author’s stackoverflow contributions: <https://stackoverflow.com/users/6555620/rvf>, accessed 2020/05/28

Bibliography

Academic journals and conference proceedings

- [1] A. J. Antczak, T. Tsubota, P. D. Kaufman, and J. M. Berger, “Structure of the yeast histone H3-ASF1 interaction: Implications for chaperone mechanism, species-specific interactions, and epigenetics.”, *BMC structural biology*, vol. 6, p. 26, Dec. 2006, ISSN: 1472-6807. DOI: 10.1186/1472-6807-6-26, ppublish.
- [2] J. Xu, “Distance-based protein folding powered by deep learning”, *bioRxiv*, 2018. DOI: 10.1101/465955.
- [3] W. Humphrey, A. Dalke, and K. Schulten, “VMD: Visual molecular dynamics.”, *Journal of molecular graphics*, vol. 14, pp. 33–8, 27–8, 1 Feb. 1996, ISSN: 0263-7855. DOI: 10.1016/0263-7855(96)00018-5, ppublish.
- [6] C. Mathé, M.-F. Sagot, T. Schiex, and P. Rouzé, “Current methods of gene prediction, their strengths and weaknesses.”, *Nucleic acids research*, vol. 30, pp. 4103–4117, 19 Oct. 2002, ISSN: 1362-4962. DOI: 10.1093/nar/gkf543, ppublish.
- [7] N. Scalzitti, A. Jeannin-Girardon, P. Collet, O. Poch, and J. D. Thompson, “A benchmark study of ab initio gene prediction methods in diverse eukaryotic organisms.”, *BMC genomics*, vol. 21, p. 293, 1 Apr. 2020, ISSN: 1471-2164. DOI: 10.1186/s12864-020-6707-9, ppublish.
- [8] Y. Yang, J. Gao, J. Wang, R. Heffernan, J. Hanson, K. K. Paliwal, and Y. Zhou, “Sixty-five years of the long march in protein secondary structure prediction: The final stretch?”, *Briefings in Bioinformatics*, vol. 19, no. 3, pp. 482–494, 2018. DOI: 10.1093/bib/bbw129.
- [9] A. Kryshchuk, T. Schwede, M. Topf, K. Fidelis, and J. Moult, “Critical assessment of methods of protein structure prediction (CASP)-Round XIII.”, *Proteins*, vol. 87, pp. 1011–1020, 12 Dec. 2019, ISSN: 1097-0134. DOI: 10.1002/prot.25823, ppublish.
- [10] P. Larrañaga, B. Calvo, R. Santana, C. Bielza, J. Galdiano, I. Inza, J. A. Lozano, R. Armañanzas, G. Santafé, A. P. Martínez, and V. Robles, “Machine learning in bioinformatics”, *Briefings in Bioinformatics*, vol. 7, no. 1, pp. 86–112, 2006. DOI: 10.1093/bib/bbk007.
- [11] F. Recknagel, “Applications of machine learning to ecological modelling”, *Ecological modelling*, vol. 146, no. 1-3, pp. 303–310, 2001.
- [12] F. Crick, “Central dogma of molecular biology.”, *Nature*, vol. 227, pp. 561–563, 5258 Aug. 1970, ISSN: 0028-0836. DOI: 10.1038/227561a0, ppublish.
- [14] C. M. Kaiser, D. H. Goldman, J. D. Chodera, I. Tinoco, and C. Bustamante, “The ribosome modulates nascent protein folding”, *Science*, vol. 334, pp. 1723–1727, 6063 Dec. 2011, ISSN: 1095-9203. DOI: 10.1126/science.1209740, ppublish.
- [15] S. C. Schuster, “Next-generation sequencing transforms today’s biology”, *Nature methods*, vol. 5, pp. 16–18, 1 Jan. 2008, ISSN: 1548-7105. DOI: 10.1038/nmeth1156, ppublish.
- [16] J. M. Schmiedel and B. Lehner, “Determining protein structures using deep mutagenesis.”, *Nature genetics*, vol. 51, pp. 1177–1186, 7 Jul. 2019, ISSN: 1546-1718. DOI: 10.1038/s41588-019-0431-x, ppublish.
- [17] I. Ladunga, “Phylogenetic continuum indicates “galaxies” in the protein universe: Preliminary results on the natural group structures of proteins.”, *Journal of molecular evolution*, vol. 34, pp. 358–375, 4 Apr. 1992, ISSN: 0022-2844. DOI: 10.1007/BF00160244, ppublish.

- [18] E. V. Koonin, Y. I. Wolf, and G. P. Karev, “The structure of the protein universe and genome evolution.”, *Nature*, vol. 420, pp. 218–223, 6912 Nov. 2002, ISSN: 0028-0836. DOI: 10.1038/nature01256, ppublish.
- [19] M. Levitt, “Nature of the protein universe.”, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, pp. 11 079–11 084, 27 Jul. 2009, ISSN: 1091-6490. DOI: 10.1073/pnas.0905029106.
- [20] T. Brunette, F. Parmeggiani, P.-S. Huang, G. Bhabha, D. C. Ekiert, S. E. Tsutakawa, G. L. Hura, J. A. Tainer, and D. Baker, “Exploring the repeat protein universe through computational protein design.”, *Nature*, vol. 528, no. 7583, pp. 580–584, Dec. 2015, ISSN: 1476-4687. DOI: 10.1038/nature16162.
- [21] J. Hou, G. E. Sims, C. Zhang, and S.-H. Kim, “A global representation of the protein fold space.”, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, pp. 2386–2390, 5 Mar. 2003, ISSN: 0027-8424. DOI: 10.1073/pnas.2628030100, ppublish.
- [22] J. Skolnick, A. K. Arakaki, S. Y. Lee, and M. Brylinski, “The continuity of protein structure space is an intrinsic property of proteins.”, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, pp. 15 690–15 695, 37 Sep. 2009, ISSN: 1091-6490. DOI: 10.1073/pnas.0907683106, ppublish.
- [23] S. Nepomnyachiy, N. Ben-Tal, and R. Kolodny, “Global view of the protein universe.”, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 111, pp. 11 691–11 696, 32 Aug. 2014, ISSN: 1091-6490. DOI: 10.1073/pnas.1403395111, ppublish.
- [24] E. V. Koonin, “Orthologs, paralogs, and evolutionary genomics.”, *Annual review of genetics*, vol. 39, pp. 309–338, 2005, ISSN: 0066-4197. DOI: 10.1146/annurev.genet.39.073003.114725, ppublish.
- [25] A. M. Altenhoff, R. A. Studer, M. Robinson-Rechavi, and C. Dessimoz, “Resolving the ortholog conjecture: Orthologs tend to be weakly, but significantly, more similar in function than paralogs”, *PLoS computational biology*, vol. 8, no. 5, 2012.
- [26] S. F. Altschul, R. Bundschuh, R. Olsen, and T. Hwa, “The estimation of statistical parameters for local alignment score distributions.”, *Nucleic acids research*, vol. 29, pp. 351–361, 2 Jan. 2001, ISSN: 1362-4962. DOI: 10.1093/nar/29.2.351, ppublish.
- [27] R. Van Noorden, B. Maher, and R. Nuzzo, “The top 100 papers.”, *Nature*, vol. 514, pp. 550–553, 7524 Oct. 2014, ISSN: 1476-4687. DOI: 10.1038/514550a, ppublish.
- [29] S. B. Needleman and C. D. Wunsch, “A general method applicable to the search for similarities in the amino acid sequence of two proteins.”, *Journal of molecular biology*, vol. 48, pp. 443–453, 3 Mar. 1970, ISSN: 0022-2836. DOI: 10.1016/0022-2836(70)90057-4, ppublish.
- [30] T. F. Smith and M. S. Waterman, “Identification of common molecular subsequences.”, *Journal of molecular biology*, vol. 147, pp. 195–197, 1 Mar. 1981, ISSN: 0022-2836. DOI: 10.1016/0022-2836(81)90087-5, ppublish.
- [31] M. Dayhoff, R. Schwartz, and B. Orcutt, “A model of evolutionary change in proteins”, in *Atlas of protein sequence and structure*, vol. 5, National Biomedical Research Foundation Silver Spring MD, 1978, pp. 345–352.
- [32] S. Henikoff and J. G. Henikoff, “Amino acid substitution matrices from protein blocks.”, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 89, pp. 10 915–10 919, 22 Nov. 1992, ISSN: 0027-8424. DOI: 10.1073/pnas.89.22.10915, ppublish.
- [33] O. Gotoh, “An improved algorithm for matching biological sequences.”, *Journal of molecular biology*, vol. 162, pp. 705–708, 3 Dec. 1982, ISSN: 0022-2836. DOI: 10.1016/0022-2836(82)90398-9, ppublish.
- [34] S. F. Altschul and B. W. Erickson, “Optimal sequence alignment using affine gap costs.”, *Bulletin of mathematical biology*, vol. 48, pp. 603–616, 5-6 1986, ISSN: 0092-8240. DOI: 10.1007/BF02462326, ppublish.

-
- [35] E. W. Myers and W. Miller, “Optimal alignments in linear space.”, *Computer applications in the biosciences : CABIOS*, vol. 4, pp. 11–17, 1 Mar. 1988, ISSN: 0266-7061. DOI: 10.1093/bioinformatics/4.1.11, ppublish.
 - [36] S. A. Manavski and G. Valle, “CUDA compatible GPU cards as efficient hardware accelerators for Smith-Waterman sequence alignment.”, *BMC bioinformatics*, vol. 9 Suppl 2, S10, Mar. 2008, ISSN: 1471-2105. DOI: 10.1186/1471-2105-9-S2-S10, epublish.
 - [37] M. S. Farrar, “Optimizing Smith-Waterman for the cell broadband engine”, *Bioinformatics*, vol. 23, pp. 156–161, 2008.
 - [38] T. Rognes, “Faster Smith-Waterman database searches with inter-sequence SIMD parallelisation”, *BMC bioinformatics*, vol. 12, p. 221, Jun. 2011, ISSN: 1471-2105. DOI: 10.1186/1471-2105-12-221, epublish.
 - [39] M. Zhao, W.-P. Lee, E. P. Garrison, and G. T. Marth, “SSW library: an SIMD Smith-Waterman C/C++ library for use in genomic applications”, *PloS one*, vol. 8, e82138, 12 2013, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0082138, epublish.
 - [40] R. Rahn, S. Budach, P. Costanza, M. Ehrhardt, J. Hancox, and K. Reinert, “Generic accelerated sequence alignment in seqan using vectorization and multi-threading.”, *Bioinformatics (Oxford, England)*, vol. 34, pp. 3437–3445, 20 Oct. 2018, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/bty380, ppublish.
 - [41] I. Elias, “Settling the intractability of multiple alignment.”, *Journal of computational biology : a journal of computational molecular cell biology*, vol. 13, pp. 1323–1339, 7 Sep. 2006, ISSN: 1066-5277. DOI: 10.1089/cmb.2006.13.1323, ppublish.
 - [42] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, “Basic local alignment search tool.”, *Journal of molecular biology*, vol. 215, pp. 403–410, 3 Oct. 1990, ISSN: 0022-2836. DOI: 10.1016/S0022-2836(05)80360-2, ppublish.
 - [43] R. C. Edgar, “Search and clustering orders of magnitude faster than BLAST”, *Bioinformatics (Oxford, England)*, vol. 26, pp. 2460–2461, 19 Oct. 2010, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btq461, ppublish.
 - [44] S. M. Kielbasa, R. Wan, K. Sato, P. Horton, and M. C. Frith, “Adaptive seeds tame genomic sequence comparison”, *Genome research*, vol. 21, no. 3, pp. 487–493, 2011.
 - [45] Y. Zhao, H. Tang, and Y. Ye, “RAPSearch2: A fast and memory-efficient protein similarity search tool for next-generation sequencing data.”, *Bioinformatics (Oxford, England)*, vol. 28, pp. 125–126, 1 Jan. 2012, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btr595, ppublish.
 - [46] M. Steinegger and J. Söding, “MMseqs2 enables sensitive protein sequence searching for the analysis of massive data sets”, *Nature biotechnology*, vol. 35, pp. 1026–1028, 11 Nov. 2017, ISSN: 1546-1696. DOI: 10.1038/nbt.3988, ppublish.
 - [47] S. Karlin and C. Burge, “Dinucleotide relative abundance extremes: A genomic signature.”, *Trends in genetics : TIG*, vol. 11, pp. 283–290, 7 Jul. 1995, ISSN: 0168-9525. DOI: 10.1016/S0168-9525(00)89076-9, ppublish.
 - [48] G. E. Sims, S.-R. Jun, G. A. Wu, and S.-H. Kim, “Alignment-free genome comparison with feature frequency profiles (ffp) and optimal resolutions.”, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 106, pp. 2677–2682, 8 Feb. 2009, ISSN: 1091-6490. DOI: 10.1073/pnas.0813249106, ppublish.
 - [49] C.-A. Leimeister and B. Morgenstern, “Kmacs: The k-mismatch average common substring approach to alignment-free sequence comparison.”, *Bioinformatics (Oxford, England)*, vol. 30, pp. 2000–2008, 14 Jul. 2014, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btu331, ppublish.
 - [50] W. R. Taylor, T. P. Flores, and C. A. Orengo, “Multiple protein structure alignment.”, *Protein science : a publication of the Protein Society*, vol. 3, pp. 1858–1870, 10 Oct. 1994, ISSN: 0961-8368. DOI: 10.1002/pro.5560031025, ppublish.

- [51] L. Holm and C. Sander, “Mapping the protein universe.”, *Science (New York, N.Y.)*, vol. 273, pp. 595–603, 5275 Aug. 1996, ISSN: 0036-8075. DOI: 10.1126/science.273.5275.595, ppublish.
- [52] R. Arnold, F. Goldenberg, H.-W. Mewes, and T. Rattei, “SIMAP—the database of all-against-all protein sequence similarities and annotations with new interfaces and increased coverage”, *Nucleic acids research*, vol. 42, pp. D279–D284, Database issue Jan. 2014, ISSN: 1362-4962. DOI: 10.1093/nar/gkt970, ppublish.
- [54] A. Pascual-García, M. Arenas, and U. Bastolla, “The molecular clock in the evolution of protein structures.”, *Systematic biology*, vol. 68, pp. 987–1002, 6 Nov. 2019, ISSN: 1076-836X. DOI: 10.1093/sysbio/syz022, ppublish.
- [55] N. Rappoport, S. Karsenty, A. Stern, N. Linial, and M. Linial, “Protonet 6.0: Organizing 10 million protein sequences in a compact hierarchical family tree”, *Nucleic acids research*, vol. 40, no. D1, pp. D313–D320, 2012.
- [56] R. Sasidharan, T. Nepusz, D. Swarbreck, E. Huala, and A. Paccanaro, “Gfam: A platform for automatic annotation of gene families.”, *Nucleic acids research*, vol. 40, e152, 19 Oct. 2012, ISSN: 1362-4962. DOI: 10.1093/nar/gks631, ppublish.
- [57] P. Jiang and M. Singh, “SPICi: A fast clustering algorithm for large biological networks.”, *Bioinformatics (Oxford, England)*, vol. 26, pp. 1105–1111, 8 Apr. 2010, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btq078, ppublish.
- [58] B. E. Suzek, Y. Wang, H. Huang, P. B. McGarvey, C. H. Wu, and U. Consortium, “Uniref clusters: A comprehensive and scalable alternative for improving sequence similarity searches.”, *Bioinformatics (Oxford, England)*, vol. 31, pp. 926–932, 6 Mar. 2015, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btu739, ppublish.
- [59] M. Mirdita, L. von den Driesch, C. Galiez, M. J. Martin, J. Söding, and M. Steinegger, “Uniclust databases of clustered and deeply annotated protein sequences and alignments.”, *Nucleic acids research*, vol. 45, pp. D170–D176, D1 Jan. 2017, ISSN: 1362-4962. DOI: 10.1093/nar/gkw1081, ppublish.
- [60] M. Y. Galperin, K. S. Makarova, Y. I. Wolf, and E. V. Koonin, “Expanded microbial genome coverage and improved protein family annotation in the COG database.”, *Nucleic acids research*, vol. 43, pp. D261–D269, Database issue Jan. 2015, ISSN: 1362-4962. DOI: 10.1093/nar/gku1223, ppublish.
- [61] M. Kanehisa and S. Goto, “KEGG: Kyoto encyclopedia of genes and genomes.”, *Nucleic acids research*, vol. 28, pp. 27–30, 1 Jan. 2000, ISSN: 0305-1048. DOI: 10.1093/nar/28.1.27, ppublish.
- [62] L. Li, C. J. Stoeckert, and D. S. Roos, “OrthoMCL: Identification of ortholog groups for eukaryotic genomes.”, *Genome research*, vol. 13, pp. 2178–2189, 9 Sep. 2003, ISSN: 1088-9051. DOI: 10.1101/gr.1224503, ppublish.
- [63] A. M. Altenhoff, N. M. Glover, C.-M. Train, K. Kaleb, A. Warwick Vesztrocy, D. Dylus, T. M. de Farias, K. Zile, C. Stevenson, J. Long, H. Redestig, G. H. Gonnet, and C. Dessimoz, “The OMA orthology database in 2018: Retrieving evolutionary relationships among all domains of life through richer web and programmatic interfaces.”, *Nucleic acids research*, vol. 46, pp. D477–D485, D1 Jan. 2018, ISSN: 1362-4962. DOI: 10.1093/nar/gkx1019, ppublish.
- [64] L. J. Jensen, P. Julien, M. Kuhn, C. von Mering, J. Muller, T. Doerks, and P. Bork, “eggNOG: Automated construction and annotation of orthologous groups of genes.”, *Nucleic acids research*, vol. 36, pp. D250–D254, Database issue Jan. 2008, ISSN: 1362-4962. DOI: 10.1093/nar/gkm796, ppublish.
- [65] D. M. Emms and S. Kelly, “OrthoFinder: Phylogenetic orthology inference for comparative genomics.”, *Genome biology*, vol. 20, p. 238, 1 Nov. 2019, ISSN: 1474-760X. DOI: 10.1186/s13059-019-1832-y, epubli.

- [67] A. Kolesnikov, X. Zhai, and L. Beyer, “Revisiting self-supervised visual representation learning”, in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 1920–1929. DOI: 10.1109/CVPR.2019.00202. [Online]. Available: http://openaccess.thecvf.com/content_CVPR_2019/html/Kolesnikov_Revisiting_Self-Supervised_Visual_Representation_Learning_CVPR_2019_paper.html.
- [68] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding”, in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Association for Computational Linguistics, 2019, pp. 4171–4186. DOI: 10.18653/v1/n19-1423.
- [69] T. Chen, S. Kornblith, M. Norouzi, and G. E. Hinton, “A simple framework for contrastive learning of visual representations”, *CoRR*, 2020. [Online]. Available: <https://arxiv.org/abs/2002.05709>.
- [71] J. L. Bentley, “Multidimensional binary search trees used for associative searching”, *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975. DOI: 10.1145/361002.361007.
- [72] S. M. Omohundro, “Five balltree construction algorithms”, (*Technical report*), 1989. [Online]. Available: <ftp://ftp.icsi.berkeley.edu/pub/techreports/1989/tr-89-063.pdf>.
- [73] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger, “The R*-Tree: An efficient and robust access method for points and rectangles”, in *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, USA, May 23-25, 1990*, H. Garcia-Molina and H. V. Jagadish, Eds., ACM Press, 1990, pp. 322–331. DOI: 10.1145/93597.98741.
- [74] A. Andoni, P. Indyk, T. Laarhoven, I. P. Razenshteyn, and L. Schmidt, “Practical and optimal LSH for angular distance”, in *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 1225–1233. [Online]. Available: <http://papers.nips.cc/paper/5893-practical-and-optimal-lsh-for-angular-distance>.
- [75] Y. A. Malkov and D. A. Yashunin, “Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, pp. 824–836, 4 Apr. 2020, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2018.2889473.
- [76] R. Agrawal and R. Srikant, “Privacy-preserving data mining”, in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, W. Chen, J. F. Naughton, and P. A. Bernstein, Eds., ACM, 2000, pp. 439–450. DOI: 10.1145/342009.335438.
- [77] X. Xi, E. J. Keogh, C. R. Shelton, L. Wei, and C. A. Ratanamahatana, “Fast time series classification using numerosity reduction”, in *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, W. W. Cohen and A. W. Moore, Eds., ser. ACM International Conference Proceeding Series, vol. 148, ACM, 2006, pp. 1033–1040. DOI: 10.1145/1143844.1143974.
- [78] A. J. Bagnall, A. Bostrom, J. Large, and J. Lines, “The great time series classification bake off: An experimental evaluation of recently proposed algorithms. extended version”, *CoRR*, vol. abs/1602.01711, 2016. arXiv: 1602.01711. [Online]. Available: <http://arxiv.org/abs/1602.01711>.
- [79] W. Jiang, “Time series classification: Nearest neighbor versus deep learning models”, *SN Applied Sciences*, vol. 2, pp. 721–738, 2020, ISSN: 2523-3963. DOI: 10.1007/s42452-020-2506-9.

- [82] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, “On the surprising behavior of distance metrics in high dimensional spaces”, in *Database Theory - ICDT 2001, 8th International Conference, London, UK, January 4-6, 2001, Proceedings*, J. V. den Bussche and V. Vianu, Eds., ser. Lecture Notes in Computer Science, vol. 1973, Springer, 2001, pp. 420–434. DOI: 10.1007/3-540-44503-X_27. [Online]. Available: https://doi.org/10.1007/3-540-44503-X_27.
- [83] K. S. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft, “When is “nearest neighbor” meaningful?”, in *Database Theory - ICDT '99, 7th International Conference, Jerusalem, Israel, January 10-12, 1999, Proceedings*, C. Beeri and P. Buneman, Eds., ser. Lecture Notes in Computer Science, vol. 1540, Springer, 1999, pp. 217–235. DOI: 10.1007/3-540-49257-7_15. [Online]. Available: https://doi.org/10.1007/3-540-49257-7_15.
- [84] M. Radovanović, A. Nanopoulos, and M. Ivanović, “Time-series classification in many intrinsic dimensions”, in *Proceedings of the SIAM International Conference on Data Mining, SDM 2010, April 29 - May 1, 2010, Columbus, Ohio, USA*, SIAM, 2010, pp. 677–688. DOI: 10.1137/1.9781611972801.59.
- [85] T. K. Vintsyuk, “Speech discrimination by dynamic programming”, *Cybernetics*, vol. 4, no. 1, pp. 52–57, 1968.
- [86] J. Aucouturier and F. Pachet, “A scale-free distribution of false positives for a large class of audio similarity measures”, *Pattern Recognit.*, vol. 41, no. 1, pp. 272–284, 2008. DOI: 10.1016/j.patcog.2007.04.012.
- [87] M. Radovanović, A. Nanopoulos, and M. Ivanović, “Hubs in space: Popular nearest neighbors in high-dimensional data”, *J. Mach. Learn. Res.*, vol. 11, pp. 2487–2531, 2010. [Online]. Available: <http://portal.acm.org/citation.cfm?id=1953015>.
- [88] T. Low, C. Borgelt, S. Stober, and A. Nürnberger, “The hubness phenomenon: Fact or artifact?”, in *Towards Advanced Data Analysis by Combining Soft Computing and Statistics*, C. Borgelt, M. Gil, J. Sousa, and M. Verleysen, Eds., Studies in Fuzziness and Soft Computing, vol. 285, Springer, 2013, pp. 267–278.
- [89] M. E. Houle, “Dimensionality, discriminability, density and distance distributions”, in *13th IEEE International Conference on Data Mining Workshops, ICDM Workshops, TX, USA, December 7-10, 2013*, W. Ding, T. Washio, H. Xiong, G. Karypis, B. M. Thuraisingham, D. J. Cook, and X. Wu, Eds., IEEE Computer Society, 2013, pp. 468–473. DOI: 10.1109/ICDMW.2013.139.
- [91] F. Angiulli, “On the behavior of intrinsically high-dimensional spaces: Distances, direct and reverse nearest neighbors, and hubness”, *J. Mach. Learn. Res.*, vol. 18, pp. 170:1–170:60, 2017. [Online]. Available: <http://jmlr.org/papers/v18/17-151.html>.
- [92] D. Schnitzer, A. Flexer, M. Schedl, and G. Widmer, “Local and global scaling reduce hubs in space”, *J. Mach. Learn. Res.*, vol. 13, pp. 2871–2902, 2012. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2503333>.
- [93] L. Zelnik-Manor and P. Perona, “Self-tuning spectral clustering”, in *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British Columbia, Canada]*, 2004, pp. 1601–1608. [Online]. Available: <http://papers.nips.cc/paper/2619-self-tuning-spectral-clustering>.
- [94] H. Jégou, H. Harzallah, and C. Schmid, “A contextual dissimilarity measure for accurate and efficient image search”, in *2007 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007), 18-23 June 2007, Minneapolis, Minnesota, USA*, IEEE Computer Society, 2007. DOI: 10.1109/CVPR.2007.382970.
- [95] R. A. Jarvis and E. A. Patrick, “Clustering using a similarity measure based on shared near neighbors”, *IEEE Trans. Computers*, vol. 22, no. 11, pp. 1025–1034, 1973. DOI: 10.1109/T-C.1973.223640.

- [96] M. E. Houle, H. Kriegel, P. Kröger, E. Schubert, and A. Zimek, “Can shared-neighbor distances defeat the curse of dimensionality?”, in *Scientific and Statistical Database Management, 22nd International Conference, SSDBM 2010, Heidelberg, Germany, June 30 - July 2, 2010. Proceedings*, M. Gertz and B. Ludäscher, Eds., ser. Lecture Notes in Computer Science, vol. 6187, Springer, 2010, pp. 482–500. DOI: 10.1007/978-3-642-13818-8_34.
- [97] A. Flexer and D. Schnitzer, “Can shared nearest neighbors reduce hubness in high-dimensional spaces?”, in *13th IEEE International Conference on Data Mining Workshops, ICDM Workshops, TX, USA, December 7-10, 2013*, W. Ding, T. Washio, H. Xiong, G. Karypis, B. M. Thuraisingham, D. J. Cook, and X. Wu, Eds., IEEE Computer Society, 2013, pp. 460–467. DOI: 10.1109/ICDMW.2013.101.
- [98] N. Tomašev and D. Mladenović, “Hubness-aware shared neighbor distances for high-dimensional k-nearest neighbor classification”, *Knowl. Inf. Syst.*, vol. 39, no. 1, pp. 89–122, 2014. DOI: 10.1007/s10115-012-0607-5.
- [99] N. Tomašev, “Taming the empirical hubness risk in many dimensions”, in *Proceedings of the 2015 SIAM International Conference on Data Mining, Vancouver, BC, Canada, April 30 - May 2, 2015*, S. Venkatasubramanian and J. Ye, Eds., SIAM, 2015, pp. 891–899. DOI: 10.1137/1.9781611974010.100.
- [100] I. Suzuki, K. Hara, M. Shimbo, Y. Matsumoto, and M. Saerens, “Investigating the Effectiveness of Laplacian-Based Kernels in Hub Reduction”, in *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*, J. Hoffmann and B. Selman, Eds., AAAI Press, 2012. [Online]. Available: <http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/4984>.
- [101] I. Suzuki, K. Hara, M. Shimbo, M. Saerens, and K. Fukumizu, “Centering similarity measures to reduce hubs”, in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, ACL, 2013, pp. 613–623. [Online]. Available: <https://www.aclweb.org/anthology/D13-1058/>.
- [102] K. Hara, I. Suzuki, M. Shimbo, K. Kobayashi, K. Fukumizu, and M. Radovanović, “Localized centering: Reducing hubness in large-sample data”, in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, B. Bonet and S. Koenig, Eds., AAAI Press, 2015, pp. 2645–2651. [Online]. Available: <http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9898>.
- [103] K. Hara, I. Suzuki, K. Kobayashi, K. Fukumizu, and M. Radovanović, “Flattening the density gradient for eliminating spatial centrality to reduce hubness”, in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, D. Schuurmans and M. P. Wellman, Eds., AAAI Press, 2016, pp. 1659–1665. [Online]. Available: <http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12055>.
- [104] A. Flexer and D. Schnitzer, “Choosing ℓ^p norms in high-dimensional spaces based on hub analysis”, *Neurocomputing*, vol. 169, pp. 281–287, 2015. DOI: 10.1016/j.neucom.2014.11.084.
- [105] E. M. Mirkes, J. Allohibi, and A. N. Gorban, “Fractional norms and quasinorms do not help to overcome the curse of dimensionality”, *CoRR*, vol. abs/2004.14230, Apr. 29, 2020. arXiv: 2004.14230 [cs.LG]. [Online]. Available: <https://arxiv.org/abs/2004.14230>.
- [106] S. Aryal, K. M. Ting, G. Haffari, and T. Washio, “Mp-dissimilarity: A data dependent dissimilarity measure”, in *2014 IEEE International Conference on Data Mining, ICDM 2014, Shenzhen, China, December 14-17, 2014*, R. Kumar, H. Toivonen, J. Pei, J. Z. Huang, and X. Wu, Eds., IEEE Computer Society, 2014, pp. 707–712. DOI: 10.1109/ICDM.2014.33.

- [107] N. Tomašev, M. Radovanović, D. Mladenović, and M. Ivanović, “A probabilistic approach to nearest-neighbor classification: naive hubness bayesian kNN”, in *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, C. Macdonald, I. Ounis, and I. Ruthven, Eds., ACM, 2011, pp. 2173–2176. DOI: 10.1145/2063576.2063919.
- [108] —, “Hubness-based fuzzy measures for high-dimensional k-nearest neighbor classification”, *Int. J. Machine Learning & Cybernetics*, vol. 5, no. 3, pp. 445–458, 2014. DOI: 10.1007/s13042-012-0137-1.
- [109] N. Tomašev and D. Mladenović, “Class imbalance and the curse of minority hubs”, *Knowl. Based Syst.*, vol. 53, pp. 157–172, 2013. DOI: 10.1016/j.knosys.2013.08.031.
- [110] N. Tomašev and K. Buza, “Hubness-aware kNN classification of high-dimensional data in presence of label noise”, *Neurocomputing*, vol. 160, pp. 157–172, 2015. DOI: 10.1016/j.neucom.2014.10.084.
- [111] N. Tomašev, K. Buza, K. Marussy, and P. B. Kis, “Hubness-aware classification, instance selection and feature construction: Survey and extensions to time-series”, in *Feature Selection for Data and Pattern Recognition*, ser. Studies in Computational Intelligence, U. Stanczyk and L. C. Jain, Eds., vol. 584, Springer, 2015, pp. 231–262. DOI: 10.1007/978-3-662-45620-0_11.
- [112] K. Hara, I. Suzuki, K. Kobayashi, K. Fukumizu, and M. Radovanović, “Reducing hubness for kernel regression”, in *Similarity Search and Applications - 8th International Conference, SISAP 2015, Glasgow, UK, October 12-14, 2015, Proceedings*, G. Amato, R. C. H. Connor, F. Falchi, and C. Gennaro, Eds., ser. Lecture Notes in Computer Science, vol. 9371, Springer, 2015, pp. 339–344. DOI: 10.1007/978-3-319-25087-8_33.
- [113] K. Buza and L. Peška, “Drug–target interaction prediction with bipartite local models and hubness-aware regression”, vol. 260, pp. 284–293, 2017, ISSN: 0925-2312. DOI: 10.1016/j.neucom.2017.04.055.
- [114] D. Schnitzer and A. Flexer, “The unbalancing effect of hubs on k-medoids clustering in high-dimensional spaces”, in *2015 International Joint Conference on Neural Networks, IJCNN 2015, Killarney, Ireland, July 12-17, 2015*, IEEE, 2015, pp. 1–8. DOI: 10.1109/IJCNN.2015.7280303.
- [115] N. Tomašev, M. Radovanović, D. Mladenović, and M. Ivanović, “The role of hubness in clustering high-dimensional data”, *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 3, pp. 739–751, 2014. DOI: 10.1109/TKDE.2013.25.
- [117] A. Flexer, “An empirical analysis of hubness in unsupervised distance-based outlier detection”, in *IEEE International Conference on Data Mining Workshops, ICDM Workshops 2016, December 12-15, 2016, Barcelona, Spain*, C. Domeniconi, F. Gullo, F. Bonchi, J. Domingo-Ferrer, R. Baeza-Yates, Z. Zhou, and X. Wu, Eds., IEEE Computer Society, 2016, pp. 716–723. DOI: 10.1109/ICDMW.2016.0106.
- [118] J. Chen, H. Fang, and Y. Saad, “Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection”, *J. Mach. Learn. Res.*, vol. 10, pp. 1989–2012, 2009. [Online]. Available: <https://dl.acm.org/citation.cfm?id=1755852>.
- [119] B. Bratic, M. E. Houle, V. Kurbalija, V. Oria, and M. Radovanovic, “NN-Descent on high-dimensional data”, in *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics, WIMS 2018, Novi Sad, Serbia, June 25-27, 2018*, R. Akerkar, M. Ivanovic, S. Kim, Y. Manolopoulos, R. Rosati, M. Savic, C. Badica, and M. Radovanovic, Eds., ACM, 2018, 20:1–20:8. DOI: 10.1145/3227609.3227643.
- [120] B. Bratić, M. E. Houle, V. Kurbalija, V. Oria, and M. Radovanović, “The influence of hubness on NN-Descent”, *International Journal on Artificial Intelligence Tools*, vol. 28, no. 6, 1960002:1–1960002:23, 2019. DOI: 10.1142/S0218213019600029.

- [121] A. Flexer, D. Schnitzer, M. Gasser, and T. Pohle, “Combining features reduces hubness in audio similarity”, in *Proceedings of the 11th International Society for Music Information Retrieval Conference, ISMIR 2010, Utrecht, Netherlands, August 9-13, 2010*, J. S. Downie and R. C. Veltkamp, Eds., International Society for Music Information Retrieval, 2010, pp. 171–176. [Online]. Available: <http://ismir2010.ismir.net/proceedings/ismir2010-31.pdf>.
- [122] A. Flexer, D. Schnitzer, and J. Schlueter, “A MIREX meta-analysis of hubness in audio music similarity”, in *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012, Mosteiro S.Bento Da Vitória, Porto, Portugal, October 8-12, 2012*, F. Gouyon, P. Herrera, L. G. Martins, and M. Müller, Eds., FEUP Edições, 2012, pp. 175–180. [Online]. Available: <http://ismir2012.ismir.net/event/papers/175-ismir-2012.pdf>.
- [123] A. Flexer, “Improving visualization of high-dimensional music similarity spaces”, in *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, M. Müller and F. Wiering, Eds., 2015, pp. 547–553. [Online]. Available: http://ismir2015.uma.es/articles/35_Paper.pdf.
- [124] A. Flexer and J. Stevens, “Mutual proximity graphs for improved reachability in music recommendation”, *Journal of New Music Research*, vol. 47, no. 1, pp. 17–28, 2018, PMID: 29348779. DOI: 10.1080/09298215.2017.1354891.
- [125] D. Schnitzer, A. Flexer, and N. Tomašev, “A case for hubness removal in high-dimensional multimedia retrieval”, in *Advances in Information Retrieval*, M. de Rijke, T. Kenter, A. P. de Vries, C. Zhai, F. de Jong, K. Radinsky, and K. Hofmann, Eds., Cham: Springer International Publishing, 2014, pp. 687–692, ISBN: 978-3-319-06028-6.
- [126] P. Knees, D. Schnitzer, and A. Flexer, “Improving neighborhood-based collaborative filtering by reducing hubness”, in *International Conference on Multimedia Retrieval, ICMR ’14, Glasgow, United Kingdom - April 01 - 04, 2014*, M. S. Kankanhalli, S. Rueger, R. Manmatha, J. M. Jose, and K. van Rijsbergen, Eds., ACM, 2014, p. 161. DOI: 10.1145/2578726.2578747.
- [127] N. Tomašev, J. Rupnik, and D. Mladenović, “The role of hubs in cross-lingual supervised document retrieval”, in *Advances in Knowledge Discovery and Data Mining, 17th Pacific-Asia Conference, PAKDD 2013, Gold Coast, Australia, April 14-17, 2013, Proceedings, Part II*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds., ser. Lecture Notes in Computer Science, vol. 7819, Springer, 2013, pp. 185–196. DOI: 10.1007/978-3-642-37456-2_16. [Online]. Available: https://doi.org/10.1007/978-3-642-37456-2_16.
- [128] G. Dinu and M. Baroni, “Improving zero-shot learning by mitigating the hubness problem”, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: <http://arxiv.org/abs/1412.6568>.
- [129] Y. Shigeto, I. Suzuki, K. Hara, M. Shimbo, and Y. Matsumoto, “Ridge regression, hubness, and zero-shot learning”, in *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2015, Porto, Portugal, September 7-11, 2015, Proceedings, Part I*, A. Appice, P. P. Rodrigues, V. S. Costa, C. Soares, J. Gama, and A. Jorge, Eds., ser. Lecture Notes in Computer Science, vol. 9284, Springer, 2015, pp. 135–151. DOI: 10.1007/978-3-319-23528-8_9.
- [130] A. Lazaridou, G. Dinu, and M. Baroni, “Hubness and pollution: Delving into cross-space mapping for zero-shot learning”, in *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, The Association for Computer Linguistics, 2015, pp. 270–280. DOI: 10.3115/v1/p15-1027.
- [131] D. Zhang, M. Luo, and F. He, “Reconstructed similarity for faster GANs-based word translation to mitigate hubness”, *Neurocomputing*, vol. 362, pp. 83–93, 2019. DOI: 10.1016/j.neucom.2019.06.082.

- [132] N. Tomašev, R. Brehar, D. Mladenović, and S. Nedevski, “The influence of hubness on nearest-neighbor methods in object recognition”, in *IEEE International Conference on Intelligent Computer Communication and Processing, ICCP 2011, Cluj-Napoca, Romania, August 25-27, 2011*, IEEE, 2011, pp. 367–374. DOI: 10.1109/ICCP.2011.6047899.
- [133] L. Zhang, T. Xiang, and S. Gong, “Learning a deep embedding model for zero-shot learning”, in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 3010–3019. DOI: 10.1109/CVPR.2017.321.
- [134] D. Schnitzer, A. Flexer, and J. Schlueter, “The relation of hubs to the doddington zoo in speaker verification”, in *21st European Signal Processing Conference, EUSIPCO 2013, Marrakech, Morocco, September 9-13, 2013*, IEEE, 2013, pp. 1–5. [Online]. Available: <http://ieeexplore.ieee.org/document/6811521/>.
- [135] K. Hara, I. Suzuki, K. Kobayashi, and K. Fukumizu, “Reducing hubness: A cause of vulnerability in recommender systems”, in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015*, R. Baeza-Yates, M. Lalmas, A. Moffat, and B. A. Ribeiro-Neto, Eds., ACM, 2015, pp. 815–818. DOI: 10.1145/2766462.2767823.
- [136] R. Heylen, M. Parente, and P. Scheunders, “Estimation of the number of endmembers in a hyperspectral image via the hubness phenomenon”, *IEEE Trans. Geoscience and Remote Sensing*, vol. 55, no. 4, pp. 2191–2200, 2017. DOI: 10.1109/TGRS.2016.2638541.
- [137] —, “Estimation of the intrinsic dimensionality in hyperspectral imagery via the hubness phenomenon”, in *Latent Variable Analysis and Signal Separation - 13th International Conference, LVA/ICA 2017, Grenoble, France, February 21-23, 2017, Proceedings*, P. Tichavský, M. Babaie-Zadeh, O. J. J. Michel, and N. Thirion-Moreau, Eds., ser. Lecture Notes in Computer Science, vol. 10169, 2017, pp. 357–366. DOI: 10.1007/978-3-319-53547-0_34.
- [138] K. Buza, “Semi-supervised naive hubness bayesian k-nearest neighbor for gene expression data”, in *Proceedings of the 9th International Conference on Computer Recognition Systems CORES 2015, Wroclaw, Poland, 25-27 May 2015*, R. Burduk, K. Jackowski, M. Kurzynski, M. Wozniak, and A. Zolnierok, Eds., ser. Advances in Intelligent Systems and Computing, vol. 403, Springer, 2015, pp. 101–110. DOI: 10.1007/978-3-319-26227-7_10.
- [139] —, “Classification of gene expression data: A hubness-aware semi-supervised approach”, *Comput. Methods Programs Biomed.*, vol. 127, pp. 105–113, 2016. DOI: 10.1016/j.cmpb.2016.01.016.
- [140] K. Buza and N. Á. Varga, “Parkinsonet: Estimation of UPDRS score using hubness-aware feedforward neural networks”, *Applied Artificial Intelligence*, vol. 30, no. 6, pp. 541–555, 2016. DOI: 10.1080/08839514.2016.1193716.
- [141] X. He, G. E. Doucet, D. Pustina, M. R. Sperling, A. D. Sharan, and J. I. Tracy, “Presurgical thalamic “hubness” predicts surgical outcome in temporal lobe epilepsy.”, *Neurology*, vol. 88, pp. 2285–2293, 24 Jun. 2017. DOI: 10.1212/WNL.0000000000004035, ppublish.
- [142] S.-I. Lee, S. Celik, B. A. Logsdon, S. M. Lundberg, T. J. Martins, V. G. Oehler, E. H. Estey, C. P. Miller, S. Chien, J. Dai, A. Saxena, C. A. Blau, and P. S. Becker, “A machine learning approach to integrate big data for precision medicine in acute myeloid leukemia.”, *Nature communications*, vol. 9, p. 42, 1 Jan. 2018, ISSN: 2041-1723. DOI: 10.1038/s41467-017-02465-5, epublish.
- [143] W. S. McCulloch and W. Pitts, “A logical calculus of the ideas immanent in nervous activity”, *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943, ISSN: 0007-4985. DOI: 10.1007/bf02478259.
- [144] F. Rosenblatt, “The perceptron: A probabilistic model for information storage and organization in the brain.”, *Psychological review*, vol. 65, pp. 386–408, 6 Nov. 1958, ISSN: 0033-295X. DOI: 10.1037/h0042519, ppublish.
- [146] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors”, *Nature*, vol. 323, pp. 533–536, 1986, ISSN: 0028-0836. DOI: 10.1038/323533a0.

- [147] G. Cybenko, “Approximation by superpositions of a sigmoidal function”, *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, 1989, ISSN: 0932-4194. DOI: 10.1007/bf02551274.
- [148] K. Hornik, “Approximation capabilities of multilayer feedforward networks”, *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991, ISSN: 0893-6080. DOI: [https://doi.org/10.1016/0893-6080\(91\)90009-T](https://doi.org/10.1016/0893-6080(91)90009-T).
- [149] M. Leshno, V. Y. Lin, A. Pinkus, and S. Schocken, “Multilayer feedforward networks with a nonpolynomial activation function can approximate any function”, *Neural Networks*, vol. 6, no. 6, pp. 861–867, 1993. DOI: 10.1016/S0893-6080(05)80131-5.
- [150] Z. Lu, H. Pu, F. Wang, Z. Hu, and L. Wang, “The expressive power of neural networks: A view from the width”, in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 6231–6239. [Online]. Available: <http://papers.nips.cc/paper/7203-the-expressive-power-of-neural-networks-a-view-from-the-width>.
- [151] B. Hanin and M. Sellke, “Approximating continuous functions by ReLU nets of minimal width”, *CoRR*, 2017. arXiv: 1710.11278.
- [152] R. Raina, A. Madhavan, and A. Y. Ng, “Large-scale deep unsupervised learning using graphics processors”, in *Proceedings of the 26th Annual International Conference on Machine Learning, ICML 2009, Montreal, Quebec, Canada, June 14-18, 2009*, A. P. Danyluk, L. Bottou, and M. L. Littman, Eds., ser. ACM International Conference Proceeding Series, vol. 382, ACM, 2009, pp. 873–880. DOI: 10.1145/1553374.1553486.
- [153] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li, “Imagenet: A large-scale hierarchical image database”, in *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, IEEE Computer Society, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.
- [154] S. Hochreiter and J. Schmidhuber, “Long short-term memory”, *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997. DOI: 10.1162/neco.1997.9.8.1735.
- [155] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks”, in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*, G. J. Gordon, D. B. Dunson, and M. Dudík, Eds., ser. JMLR Proceedings, vol. 15, JMLR.org, 2011, pp. 315–323. [Online]. Available: <http://proceedings.mlr.press/v15/glorot11a/glorot11a.pdf>.
- [156] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.
- [157] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, “Self-normalizing neural networks”, in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 971–980. [Online]. Available: <http://papers.nips.cc/paper/6698-self-normalizing-neural-networks>.
- [158] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, Y. Bengio, A. Bergeron, J. Bergstra, V. Bisson, J. Blecher Snyder, N. Bouchard, N. Boulanger-Lewandowski, X. Bouthillier, A. de Brébisson, O. Breuleux, P.-L. Carrier, K. Cho, J. Chorowski, P. Christiano, T. Cooijmans, M.-A. Côté, M. Côté, A. Courville, Y. N. Dauphin, O. Delalleau, J. Demouth, G. Desjardins, S. Dieleman, L. Dinh, M. Ducoffe, V. Dumoulin, S. Ebrahimi Kahou, D. Erhan, Z. Fan, O. Firat, M. Germain, X. Glorot, I. Goodfellow, M. Graham, C. Gulcehre, P. Hamel, I. Harlouchet, J.-P. Heng, B. Hidasi, S. Honari, A. Jain, S. Jean, K. Jia, M. Korobov, V. Kulkarni, A. Lamb, P. Lamblin, E. Larsen, C. Laurent, S. Lee, S. Lefrancois, S. Lemieux, N. Léonard, Z. Lin, J. A. Livezey, C. Lorenz, J. Lowin, Q. Ma, P.-A. Manzagol, O. Mastropietro, R. T.

- McGibbon, R. Memisevic, B. van Merriënboer, V. Michalski, M. Mirza, A. Orlandi, C. Pal, R. Pascanu, M. Pezeshki, C. Raffel, D. Renshaw, M. Rocklin, A. Romero, M. Roth, P. Sadowski, J. Salvatier, F. Savard, J. Schlüter, J. Schulman, G. Schwartz, I. V. Serban, D. Serdyuk, S. Shabanian, É. Simon, S. Spieckermann, S. R. Subramanyam, J. Sygnowski, J. Tanguay, G. van Tulder, J. Turian, S. Urban, P. Vincent, F. Visin, H. de Vries, D. Warde-Farley, D. J. Webb, M. Willson, K. Xu, L. Xue, L. Yao, S. Zhang, and Y. Zhang, “Theano: A Python framework for fast computation of mathematical expressions”, *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: <http://arxiv.org/abs/1605.02688>.
- [159] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “TensorFlow: A system for large-scale machine learning”, in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283. [Online]. Available: <https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf>.
- [160] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, “PyTorch: An Imperative Style, High-Performance Deep Learning Library”, in *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. B. Fox, and R. Garnett, Eds., 2019, pp. 8024–8035. [Online]. Available: <http://papers.nips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library>.
- [161] R. H. R. Hahnloser and H. S. Seung, “Permitted and forbidden sets in symmetric threshold-linear networks”, in *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, T. K. Leen, T. G. Dietterich, and V. Tresp, Eds., MIT Press, 2000, pp. 217–223. [Online]. Available: <http://papers.nips.cc/paper/1793-permitted-and-forbidden-sets-in-symmetric-threshold-linear-networks>.
- [163] Y. LeCun, Y. Bengio, and G. E. Hinton, “Deep learning”, *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. DOI: 10.1038/nature14539.
- [164] S. Min, B. Lee, and S. Yoon, “Deep learning in bioinformatics”, *Briefings in Bioinformatics*, vol. 18, no. 5, pp. 851–869, Jul. 2016, ISSN: 1467-5463. DOI: 10.1093/bib/bbw068.
- [165] C. Angermueller, T. Pärnamaa, L. Parts, and O. Stegle, “Deep learning for computational biology”, *Molecular systems biology*, vol. 12, no. 7, p. 878, 2016. DOI: 10.15252/msb.20156651.
- [166] G. Eraslan, Ž. Avsec, J. Gagneur, and F. J. Theis, “Deep learning: New computational modelling techniques for genomics”, *Nature Reviews Genetics*, vol. 20, no. 7, pp. 389–403, 2019.
- [167] K. M. Chen, E. M. Cofer, J. Zhou, and O. G. Troyanskaya, “Selene: a PyTorch-based deep learning library for sequence data.”, *Nature methods*, vol. 16, pp. 315–318, 4 Apr. 2019, ISSN: 1548-7105. DOI: 10.1038/s41592-019-0360-8, ppublish.
- [168] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, “Dermatologist-level classification of skin cancer with deep neural networks.”, *Nature*, vol. 542, pp. 115–118, 7639 Feb. 2017, ISSN: 1476-4687. DOI: 10.1038/nature21056.
- [169] J. De Fauw, J. R. Ledsam, B. Romera-Paredes, S. Nikolov, N. Tomasev, S. Blackwell, H. Askham, X. Glorot, B. O’Donoghue, D. Visentin, G. van den Driessche, B. Lakshminarayanan, C. Meyer, F. Mackinder, S. Bouton, K. Ayoub, R. Chopra, D. King, A. Karthikesalingam, C. O. Hughes, R. Raine, J. Hughes, D. A. Sim, C. Egan, A. Tufail, H. Montgomery, D. Hassabis, G. Rees, T. Back, P. T. Khaw, M. Suleyman, J. Cornebise, P. A. Keane, and O. Ronneberger, “Clinically applicable deep learning for diagnosis and referral in retinal disease.”, *Nature medicine*, vol. 24, pp. 1342–1350, 9 Sep. 2018, ISSN: 1546-170X. DOI: 10.1038/s41591-018-0107-6, ppublish.

- [170] A. Y. Hannun, P. Rajpurkar, M. Haghpanahi, G. H. Tison, C. Bourn, M. P. Turakhia, and A. Y. Ng, “Cardiologist-level arrhythmia detection and classification in ambulatory electrocardiograms using a deep neural network.”, *Nature medicine*, vol. 25, pp. 65–69, 1 Jan. 2019, ISSN: 1546-170X. DOI: 10.1038/s41591-018-0268-3, ppublish.
- [171] A. W. Senior, R. Evans, J. Jumper, J. Kirkpatrick, L. Sifre, T. Green, C. Qin, A. Židek, A. W. R. Nelson, A. Bridgland, H. Penedones, S. Petersen, K. Simonyan, S. Crossan, P. Kohli, D. T. Jones, D. Silver, K. Kavukcuoglu, and D. Hassabis, “Improved protein structure prediction using potentials from deep learning.”, *Nature*, vol. 577, pp. 706–710, 7792 Jan. 2020, ISSN: 1476-4687. DOI: 10.1038/s41586-019-1923-7, ppublish.
- [173] D. Szklarczyk, A. L. Gable, D. Lyon, A. Junge, S. Wyder, J. Huerta-Cepas, M. Simonovic, N. T. Doncheva, J. H. Morris, P. Bork, L. J. Jensen, and C. v. Mering, “STRING v11: Protein-protein association networks with increased coverage, supporting functional discovery in genome-wide experimental datasets.”, *Nucleic acids research*, vol. 47, pp. D607–D613, D1 Jan. 2019, ISSN: 1362-4962. DOI: 10.1093/nar/gky1131, ppublish.
- [174] J. Huerta-Cepas, D. Szklarczyk, D. Heller, A. Hernández-Plaza, S. K. Forslund, H. Cook, D. R. Mende, I. Letunic, T. Rattei, L. J. Jensen, C. von Mering, and P. Bork, “eggNOG 5.0: A hierarchical, functionally and phylogenetically annotated orthology resource based on 5090 organisms and 2502 viruses.”, *Nucleic acids research*, vol. 47, pp. D309–D314, D1 Jan. 2019, ISSN: 1362-4962. DOI: 10.1093/nar/gky1085, ppublish.
- [175] U. Consortium, “Uniprot: A worldwide hub of protein knowledge.”, *Nucleic acids research*, vol. 47, pp. D506–D515, D1 Jan. 2019, ISSN: 1362-4962. DOI: 10.1093/nar/gky1049, ppublish.
- [176] J. Buhler, “Efficient large-scale sequence comparison by locality-sensitive hashing.”, *Bioinformatics (Oxford, England)*, vol. 17, pp. 419–428, 5 May 2001, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/17.5.419, ppublish.
- [177] K. Berlin, S. Koren, C.-S. Chin, J. P. Drake, J. M. Landolin, and A. M. Phillippy, “Assembling large genomes with single-molecule sequencing and locality-sensitive hashing.”, *Nature biotechnology*, vol. 33, pp. 623–630, 6 Jun. 2015, ISSN: 1546-1696. DOI: 10.1038/nbt.3238, ppublish.
- [178] G. Marçais, D. F. DeBlasio, P. Pandey, and C. Kingsford, “Locality-sensitive hashing for the edit distance”, *Bioinformatics*, vol. 35, no. 14, pp. i127–i135, 2019. DOI: 10.1093/bioinformatics/btz354.
- [179] M. Wainberg, D. Merico, A. Delong, and B. J. Frey, “Deep learning in biomedicine”, *Nature biotechnology*, vol. 36, no. 9, pp. 829–838, 2018.
- [180] P. Indyk and R. Motwani, “Approximate nearest neighbors: Towards removing the curse of dimensionality”, in *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998*, J. S. Vitter, Ed., ACM, 1998, pp. 604–613. DOI: 10.1145/276698.276876.
- [181] H. Jégou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search”, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2011. DOI: 10.1109/TPAMI.2010.57.
- [182] M. Iwasaki and D. Miyazaki, “Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data”, *CoRR*, vol. abs/1810.07355, 2018. arXiv: 1810.07355. [Online]. Available: <http://arxiv.org/abs/1810.07355>.
- [184] W. Zheng, L. Yang, R. J. Genco, J. Wactawski-Wende, M. Buck, and Y. Sun, “SENSE: Siamese neural network for sequence embedding and alignment-free comparison.”, *Bioinformatics (Oxford, England)*, vol. 35, pp. 1820–1828, 11 Jun. 2019, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/bty887, ppublish.
- [185] J. De Leeuw, “Applications of convex analysis to multidimensional scaling”, in *Recent developments in statistics*, J. R. Barra, F. Brodeau, G. Romier, and B. Van Cutsem, Eds., North Holland Publishing Company, 1977. DOI: 10.1.1.329.2432.
- [186] J. B. Kruskal, “Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis”, *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.

- [187] C. Faloutsos and K. Lin, “FastMap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets”, in *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data, San Jose, California, USA, May 22-25, 1995*, M. J. Carey and D. A. Schneider, Eds., ACM Press, 1995, pp. 163–174. DOI: 10.1145/223784.223812.
- [188] V. Athitsos, M. Potamias, P. Papapetrou, and G. Kollios, “Nearest neighbor retrieval using distance-based hashing”, in *Proceedings of the 24th International Conference on Data Engineering, ICDE 2008, April 7-12, 2008, Cancún, Mexico*, G. Alonso, J. A. Blakeley, and A. L. P. Chen, Eds., IEEE Computer Society, 2008, pp. 327–336. DOI: 10.1109/ICDE.2008.4497441.
- [189] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, “Generative adversarial networks”, in *Proc. 27th Int. Conf. Neural Information Processing Systems*, 2014, pp. 2672–2680.
- [190] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, “Signature verification using a siamese time delay neural network”, in *Advances in Neural Information Processing Systems 6, [7th NIPS Conference, Denver, Colorado, USA, 1993]*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., Morgan Kaufmann, 1993, pp. 737–744. [Online]. Available: <http://papers.nips.cc/paper/769-signature-verification-using-a-siamese-time-delay-neural-network>.
- [191] R. Hadsell, S. Chopra, and Y. LeCun, “Dimensionality reduction by learning an invariant mapping”, in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006), 17-22 June 2006, New York, NY, USA*, IEEE Computer Society, 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.
- [192] G. Chechik, V. Sharma, U. Shalit, and S. Bengio, “Large scale online learning of image similarity through ranking”, *J. Mach. Learn. Res.*, vol. 11, pp. 1109–1135, 2010. [Online]. Available: <https://dl.acm.org/citation.cfm?id=1756042>.
- [193] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering”, in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, IEEE Computer Society, 2015, pp. 815–823. DOI: 10.1109/CVPR.2015.7298682.
- [194] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need”, in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5998–6008. [Online]. Available: <http://papers.nips.cc/paper/7181-attention-is-all-you-need>.
- [195] N. Strodthoff, P. Wagner, M. Wenzel, and W. Samek, “UDSMProt: Universal deep sequence models for protein classification.”, *Bioinformatics (Oxford, England)*, Jan. 2020, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btaa003, aheadofprint.
- [196] A. Rives, S. Goyal, J. Meier, D. Guo, M. Ott, C. L. Zitnick, J. Ma, and R. Fergus, “Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences”, *bioRxiv*, p. 622 803, 2019. [Online]. Available: <https://www.biorxiv.org/content/10.1101/622803v1>.
- [197] P. J. A. Cock, T. Antao, J. T. Chang, B. A. Chapman, C. J. Cox, A. Dalke, I. Friedberg, T. Hamelryck, F. Kauff, B. Wilczynski, and M. J. L. de Hoon, “Biopython: Freely available python tools for computational molecular biology and bioinformatics.”, *Bioinformatics (Oxford, England)*, vol. 25, pp. 1422–1423, 11 Jun. 2009, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btp163, ppublish.
- [198] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition”, *Neural Computation*, vol. 1(4), pp. 541–551, 1989, ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541.
- [199] J. T. Barron, “Continuously differentiable exponential linear units”, *CoRR*, 2017. arXiv: 1704.07483. [Online]. Available: <http://arxiv.org/abs/1704.07483>.

- [200] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: <http://arxiv.org/abs/1412.6980>.
- [201] M. Aumüller, E. Bernhardsson, and A. Faithfull, “ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms”, *Information Systems*, vol. 87, p. 101 374, 2020, ISSN: 0306-4379. DOI: <https://doi.org/10.1016/j.is.2019.02.006>.
- [202] L. Boytsov and B. Naidan, “Engineering efficient and effective non-metric space library”, in *Similarity Search and Applications - 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings*, N. R. Brisaboa, O. Pedreira, and P. Zezula, Eds., ser. Lecture Notes in Computer Science, vol. 8199, Springer, 2013, pp. 280–293. DOI: 10.1007/978-3-642-41062-8_28.
- [203] R. B. Allan and R. C. Laskar, “On domination and independent domination numbers of a graph”, *Discret. Math.*, vol. 23, no. 2, pp. 73–76, 1978. DOI: 10.1016/0012-365X(78)90105-X.
- [204] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, “Densely connected convolutional networks”, in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 2261–2269. DOI: 10.1109/CVPR.2017.243.
- [205] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. W. Senior, and K. Kavukcuoglu, “WaveNet: A generative model for raw audio”, *CoRR*, vol. abs/1609.03499, 2016. arXiv: 1609.03499.
- [206] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”, *CoRR*, 2018. arXiv: 1803.01271.
- [207] K. Cho, B. van Merriënboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation”, in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds., ACL, 2014, pp. 1724–1734. DOI: 10.3115/v1/d14-1179.
- [208] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate”, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: <http://arxiv.org/abs/1409.0473>.
- [209] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew, “HuggingFace’s Transformers: State-of-the-art natural language processing”, *CoRR*, 2019. arXiv: 1910.03771.
- [210] N. Kitaev, L. Kaiser, and A. Levskaya, “Reformer: The efficient transformer”, in *International Conference on Learning Representations (ICLR)*, 2020, (accepted for publication). arXiv: 2001.04451.
- [211] R. Salakhutdinov and G. E. Hinton, “Semantic hashing”, *Int. J. Approx. Reason.*, vol. 50, no. 7, pp. 969–978, 2009. DOI: 10.1016/j.ijar.2008.11.006.
- [212] J. Schlüter, “Learning binary codes for efficient large-scale music similarity search”, in *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013, Curitiba, Brazil, November 4-8, 2013*, A. de Souza Britto Jr., F. Gouyon, and S. Dixon, Eds., 2013, pp. 581–586. [Online]. Available: http://www.ppgia.pucpr.br/ismir2013/wp-content/uploads/2013/09/189_Paper.pdf.
- [213] R. Manmatha, C. Wu, A. J. Smola, and P. Krähenbühl, “Sampling matters in deep embedding learning”, in *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, IEEE Computer Society, 2017, pp. 2859–2867. DOI: 10.1109/ICCV.2017.309.

- [214] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting”, *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014. [Online]. Available: <http://dl.acm.org/citation.cfm?id=2670313>.
- [215] T. Beppler and B. Berger, “Learning protein sequence embeddings using information from structure”, in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019. [Online]. Available: <https://openreview.net/forum?id=SygLehCqtm>.
- [216] S. El-Gebali, J. Mistry, A. Bateman, S. R. Eddy, A. Luciani, S. C. Potter, M. Qureshi, L. J. Richardson, G. A. Salazar, A. Smart, E. L. L. Sonnhammer, L. Hirsh, L. Paladin, D. Piovesan, S. C. E. Tosatto, and R. D. Finn, “The Pfam protein families database in 2019.”, *Nucleic acids research*, vol. 47, pp. D427–D432, D1 Jan. 2019, ISSN: 1362-4962. DOI: 10.1093/nar/gky995, ppublish.
- [219] B. Rekapalli, K. Wuichet, G. D. Peterson, and I. B. Zhulin, “Dynamics of domain coverage of the protein sequence universe.”, *BMC genomics*, vol. 13, p. 634, Nov. 2012, ISSN: 1471-2164. DOI: 10.1186/1471-2164-13-634.
- [220] K. S. Makarova, Y. I. Wolf, and E. V. Koonin, “Towards functional characterization of archaeal genomic dark matter.”, *Biochemical Society transactions*, vol. 47, pp. 389–398, 1 Feb. 2019, ISSN: 1470-8752. DOI: 10.1042/BST20180560, ppublish.
- [221] C. Wiwie, J. Baumbach, and R. Röttger, “Comparing the performance of biomedical clustering methods.”, *Nature methods*, vol. 12, pp. 1033–1038, 11 Nov. 2015, ISSN: 1548-7105. DOI: 10.1038/nmeth.3583, ppublish.
- [222] M. Hofmarcher, A. Mayr, E. Rumetshofer, P. Ruch, P. Renz, J. Schimunek, P. Seidl, A. Vall, M. Widrich, S. Hochreiter, and G. Klambauer, “Large-scale ligand-based virtual screening for SARS-CoV-2 inhibitors using deep neural networks”, *CoRR*, 2020. arXiv: 2004.00979.
- [223] B. Tilocca, A. Soggiu, M. Sanguinetti, V. Musella, D. Britti, L. Bonizzi, A. Urbani, and P. Roncada, “Comparative computational analysis of SARS-CoV-2 nucleocapsid protein epitopes in taxonomically related coronaviruses.”, *Microbes and infection*, Apr. 2020, ISSN: 1769-714X. DOI: 10.1016/j.micinf.2020.04.002, aheadofprint.
- [224] Y. Han and P. Král, “Computational design of ACE2-based peptide inhibitors of SARS-CoV-2”, *ACS nano*, 2020.
- [225] R. Cagliani, D. Forni, M. Clerici, and M. Sironi, “Computational inference of selection underlying the evolution of the novel coronavirus, SARS-CoV-2.”, *Journal of virology*, Apr. 2020, ISSN: 1098-5514. DOI: 10.1128/JVI.00411-20, aheadofprint.
- [226] K. G. Andersen, A. Rambaut, W. I. Lipkin, E. C. Holmes, and R. F. Garry, “The proximal origin of SARS-CoV-2.”, *Nature medicine*, vol. 26, pp. 450–452, 4 Apr. 2020, ISSN: 1546-170X. DOI: 10.1038/s41591-020-0820-9, ppublish.
- [227] X. Huang, R. Pearce, and Y. Zhang, “Computational design of peptides to block binding of the SARS-CoV-2 spike protein to human ACE2”, *bioRxiv*, 2020. DOI: 10.1101/2020.03.28.013607.
- [228] S. Liu, Q. Zheng, and Z. Wang, “Potential covalent drugs targeting the main protease of the SARS-CoV-2 coronavirus.”, *Bioinformatics (Oxford, England)*, Apr. 2020, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btaa224, aheadofprint.
- [229] S. Cleemput, W. Dumon, V. Fonseca, W. A. Karim, M. Giovanetti, L. C. Alcantara, K. Deforche, and T. de Oliveira, “Genome detective coronavirus typing tool for rapid identification and characterization of novel coronavirus genomes.”, *Bioinformatics (Oxford, England)*, Feb. 2020, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btaa145, aheadofprint.
- [230] K. Dick, K. K. Biggar, and J. R. Green, “Computational prediction of the comprehensive SARS-CoV-2 vs. human interactome to guide the design of therapeutics”, *bioRxiv*, 2020. DOI: 10.1101/2020.03.29.014381.

-
- [231] C. S. Lupala, X. Li, J. Lei, H. Chen, J. Qi, H. Liu, and X.-d. Su, “Computational simulations reveal the binding dynamics between human ACE2 and the receptor binding domain of SARS-CoV-2 spike protein”, *bioRxiv*, 2020. DOI: 10.1101/2020.03.24.005561.
 - [232] H. T. T. Lai, L. H. Nguyen, A. Kranjc, T. T. Nguyen, and D. Nguyen-Manh, “Elucidating the differences in the molecular mechanism of receptor binding between 2019-nCoV and the SARS-CoV viruses using computational tools”, *bioRxiv*, 2020. DOI: 10.1101/2020.04.21.053009.
 - [233] J. Lung, Y.-S. Lin, Y.-H. Yang, Y.-L. Chou, G.-H. Chang, M.-S. Tsai, C.-M. Hsu, R.-A. Yeh, L.-H. Shu, Y.-C. Cheng, H. T. Liu, and C.-Y. Wu, “The potential SARS-CoV-2 entry inhibitor”, *bioRxiv*, 2020. DOI: 10.1101/2020.03.26.009803.
 - [234] U. F. Chowdhury, M. U. S. Shohan, K. I. Hoque, M. A. Beg, M. K. S. Siam, and M. A. Moni, “A computational approach to design potential siRNA molecules as a prospective tool for silencing nucleocapsid phosphoprotein and surface glycoprotein gene of SARS-CoV-2”, *bioRxiv*, 2020. DOI: 10.1101/2020.04.10.036335.
 - [235] M. W. Libbrecht and W. S. Noble, “Machine learning applications in genetics and genomics.”, *Nature reviews. Genetics*, vol. 16, pp. 321–332, 6 Jun. 2015, ISSN: 1471-0064. DOI: 10.1038/nrg3920, ppublish.
 - [236] J. Zou, M. Huss, A. Abid, P. Mohammadi, A. Torkamani, and A. Telenti, “A primer on deep learning in genomics.”, *Nature genetics*, vol. 51, pp. 12–18, 1 Jan. 2019, ISSN: 1546-1718. DOI: 10.1038/s41588-018-0295-5, ppublish.
 - [237] C. Yu, M. Deng, S.-Y. Cheng, S.-C. Yau, R. L. He, and S. S.-T. Yau, “Protein space: A natural method for realizing the nature of protein universe.”, *Journal of theoretical biology*, vol. 318, pp. 197–204, Feb. 2013, ISSN: 1095-8541. DOI: 10.1016/j.jtbi.2012.11.005.
 - [238] S. van Dongen, “Graph clustering via a discrete uncoupling process”, *SIAM J. Matrix Analysis Applications*, vol. 30, no. 1, pp. 121–141, 2008. DOI: 10.1137/040608635.
 - [239] A. J. Enright, S. Van Dongen, and C. A. Ouzounis, “An efficient algorithm for large-scale detection of protein families.”, *Nucleic acids research*, vol. 30, pp. 1575–1584, 7 Apr. 2002, ISSN: 1362-4962. DOI: 10.1093/nar/30.7.1575, ppublish.
 - [240] D. Paez-Espino, E. A. Eloë-Fadrosh, G. A. Pavlopoulos, A. D. Thomas, M. Huntemann, N. Mikhailova, E. Rubin, N. N. Ivanova, and N. C. Kyrpides, “Uncovering Earth’s virome.”, *Nature*, vol. 536, pp. 425–430, 7617 Aug. 2016, ISSN: 1476-4687. DOI: 10.1038/nature19094, ppublish.
 - [241] E. L. Huttlin, R. J. Bruckner, J. A. Paulo, J. R. Cannon, L. Ting, K. Baltier, G. Colby, F. Gebreab, M. P. Gygi, H. Parzen, J. Szpyt, S. Tam, G. Zarraga, L. Pontano-Vaites, S. Swarup, A. E. White, D. K. Schweppe, R. Rad, B. K. Erickson, R. A. Obar, K. G. Guruharsha, K. Li, S. Artavanis-Tsakonas, S. P. Gygi, and J. W. Harper, “Architecture of the human interactome defines protein communities and disease networks.”, *Nature*, vol. 545, pp. 505–509, 7655 May 2017, ISSN: 1476-4687. DOI: 10.1038/nature22366, ppublish.
 - [242] B. Al-Shayeb, R. Sachdeva, L.-X. Chen, F. Ward, P. Munk, A. Devoto, C. J. Castelle, M. R. Olm, K. Bouma-Gregson, Y. Amano, C. He, R. Méheust, B. Brooks, A. Thomas, A. Lavy, P. Matheus-Carnevali, C. Sun, D. S. A. Goltsman, M. A. Borton, A. Sharrar, A. L. Jaffe, T. C. Nelson, R. Kantor, R. Keren, K. R. Lane, I. F. Farag, S. Lei, K. Finstad, R. Amundson, K. Anantharaman, J. Zhou, A. J. Probst, M. E. Power, S. G. Tringe, W.-J. Li, K. Wrighton, S. Harrison, M. Morowitz, D. A. Relman, J. A. Doudna, A.-C. Lehours, L. Warren, J. H. D. Cate, J. M. Santini, and J. F. Banfield, “Clades of huge phages from across Earth’s ecosystems.”, *Nature*, vol. 578, pp. 425–431, 7795 Feb. 2020, ISSN: 1476-4687. DOI: 10.1038/s41586-020-2007-4, ppublish.
 - [243] M. Ashburner, C. A. Ball, J. A. Blake, D. Botstein, H. Butler, J. M. Cherry, A. P. Davis, K. Dolinski, S. S. Dwight, J. T. Eppig, M. A. Harris, D. P. Hill, L. Issel-Tarver, A. Kasarskis, S. Lewis, J. C. Matese, J. E. Richardson, M. Ringwald, G. M. Rubin, and G. Sherlock, “Gene ontology: Tool for the unification of biology.”, *Nature genetics*, vol. 25, pp. 25–29, 1 May 2000, ISSN: 1061-4036. DOI: 10.1038/75556, ppublish.

- [244] J. Z. Wang, Z. Du, R. Payattakool, P. S. Yu, and C.-F. Chen, “A new method to measure the semantic similarity of GO terms.”, *Bioinformatics (Oxford, England)*, vol. 23, pp. 1274–1281, 10 May 2007, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btm087, ppublish.
- [245] P. H. Guzzi, M. Mina, C. Guerra, and M. Cannataro, “Semantic similarity analysis of protein data: Assessment with biological features and issues.”, *Briefings in bioinformatics*, vol. 13, pp. 569–585, 5 Sep. 2012, ISSN: 1477-4054. DOI: 10.1093/bib/bbr066, ppublish.
- [246] M. V. Omelchenko, M. Y. Galperin, Y. I. Wolf, and E. V. Koonin, “Non-homologous isofunctional enzymes: A systematic analysis of alternative solutions in enzyme evolution.”, *Biology direct*, vol. 5, p. 31, Apr. 2010, ISSN: 1745-6150. DOI: 10.1186/1745-6150-5-31, epubliish.
- [247] A. Bairoch and R. Apweiler, “The SWISS-PROT protein sequence database: Its relevance to human molecular medical research.”, *Journal of molecular medicine (Berlin, Germany)*, vol. 75, pp. 312–316, 5 May 1997, ISSN: 0946-2716, ppublish.
- [248] wwPDB consortium, “Protein Data Bank: the single global archive for 3D macromolecular structure data.”, *Nucleic acids research*, vol. 47, pp. D520–D528, D1 Jan. 2019, ISSN: 1362-4962. DOI: 10.1093/nar/gky949, ppublish.
- [249] T. J. Hubbard, A. G. Murzin, S. E. Brenner, and C. Chothia, “SCOP: A structural classification of proteins database.”, *Nucleic acids research*, vol. 25, pp. 236–239, 1 Jan. 1997, ISSN: 0305-1048. DOI: 10.1093/nar/25.1.236, ppublish.
- [250] N. K. Fox, S. E. Brenner, and J.-M. Chandonia, “SCOPE: Structural classification of proteins—extended, integrating SCOP and ASTRAL data and classification of new structures.”, *Nucleic acids research*, vol. 42, pp. D304–D309, Database issue Jan. 2014, ISSN: 1362-4962. DOI: 10.1093/nar/gkt1240, ppublish.
- [251] K. K. Yang, Z. Wu, C. N. Bedbrook, and F. H. Arnold, “Learned protein embeddings for machine learning.”, *Bioinformatics (Oxford, England)*, vol. 34, pp. 2642–2648, 15 Aug. 2018, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/bty178, ppublish.
- [252] E. Asgari and M. R. K. Mofrad, “Continuous distributed representation of biological sequences for deep proteomics and genomics.”, *PloS one*, vol. 10, e0141287, 11 2015, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0141287, epubliish.
- [253] Y. Li, S. Wang, R. Umarov, B. Xie, M. Fan, L. Li, and X. Gao, “DEEPre: sequence-based enzyme EC number prediction by deep learning.”, *Bioinformatics (Oxford, England)*, vol. 34, pp. 760–769, 5 Mar. 2018, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btx680, ppublish.
- [254] A. Y. Ng, M. I. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm”, in *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., MIT Press, 2001, pp. 849–856. [Online]. Available: <http://papers.nips.cc/paper/2092-on-spectral-clustering-analysis-and-an-algorithm>.
- [255] A. Paccanaro, J. A. Casbon, and M. A. S. Saqi, “Spectral clustering of protein sequences.”, *Nucleic acids research*, vol. 34, pp. 1571–1580, 5 2006, ISSN: 1362-4962. DOI: 10.1093/nar/gkj515, epubliish.
- [256] B. J. Frey and D. Dueck, “Clustering by passing messages between data points.”, *Science (New York, N.Y.)*, vol. 315, pp. 972–976, 5814 Feb. 2007, ISSN: 1095-9203. DOI: 10.1126/science.1136800, ppublish.
- [257] S. Karlin and S. F. Altschul, “Methods for assessing the statistical significance of molecular sequence features by using general scoring schemes.”, *Proceedings of the National Academy of Sciences of the United States of America*, vol. 87, pp. 2264–2268, 6 Mar. 1990, ISSN: 0027-8424. DOI: 10.1073/pnas.87.6.2264, ppublish.
- [258] S. Wolfsheimer, B. Burghardt, and A. K. Hartmann, “Local sequence alignments statistics: Deviations from Gumbel statistics in the rare-event tail.”, *Algorithms for molecular biology*, vol. 2, p. 9, Jul. 2007, ISSN: 1748-7188. DOI: 10.1186/1748-7188-2-9, epubliish.

- [259] P. Ortet and O. Bastien, “Where does the alignment score distribution shape come from?”, *Evolutionary bioinformatics online*, vol. 6, pp. 159–187, Dec. 2010, ISSN: 1176-9343. DOI: 10.4137/EB0.S5875, epubliish.
- [260] H. Pang, J. Tang, S.-S. Chen, and S. Tao, “Statistical distributions of optimal global alignment scores of random protein sequences.”, *BMC bioinformatics*, vol. 6, p. 257, Oct. 2005, ISSN: 1471-2105. DOI: 10.1186/1471-2105-6-257, epubliish.
- [261] C. Li, M. Zhang, D. G. Andersen, and Y. He, “Improving approximate nearest neighbor search through learned adaptive early termination”, in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD)*, vol. 3, 2020.
- [262] J. Huerta-Cepas, K. Forslund, L. P. Coelho, D. Szklarczyk, L. J. Jensen, C. von Mering, and P. Bork, “Fast genome-wide functional annotation through orthology assignment by eggno-mapper.”, *Molecular biology and evolution*, vol. 34, pp. 2115–2122, 8 Aug. 2017, ISSN: 1537-1719. DOI: 10.1093/molbev/msx148, ppublish.
- [263] E. D. Graham, J. F. Heidelberg, and B. J. Tully, “Binsanity: Unsupervised clustering of environmental microbial assemblies using coverage and affinity propagation.”, *PeerJ*, vol. 5, e3035, 2017, ISSN: 2167-8359. DOI: 10.7717/peerj.3035, epubliish.
- [264] M. D. Shirley, Z. Ma, B. S. Pedersen, and S. J. Wheelan, “Efficient ‘pythonic’ access to FASTA files using pyfaidx”, *PeerJ PrePrints*, vol. 3:e970v1, 2015. DOI: 10.7287/peerj.preprints.970.
- [265] M. Tamura and P. D’haeseleer, “Microbial genotype-phenotype mapping by class association rule mining.”, *Bioinformatics (Oxford, England)*, vol. 24, pp. 1523–1529, 13 Jul. 2008, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btn210, ppublish.
- [266] N. J. MacDonald and R. G. Beiko, “Efficient learning of microbial genotype-phenotype association rules.”, *Bioinformatics (Oxford, England)*, vol. 26, pp. 1834–1840, 15 Aug. 2010, ISSN: 1367-4811. DOI: 10.1093/bioinformatics/btq305, ppublish.
- [267] R. Feldbauer, F. Schulz, M. Horn, and T. Rattei, “Prediction of microbial phenotypes based on comparative genomics.”, *BMC bioinformatics*, vol. 16 Suppl 14, S1, 2015, ISSN: 1471-2105. DOI: 10.1186/1471-2105-16-S14-S1, ppublish.
- [268] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system”, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, and R. Rastogi, Eds., ACM, 2016, pp. 785–794. DOI: 10.1145/2939672.2939785.
- [269] S. M. Lundberg and S.-I. Lee, “A unified approach to interpreting model predictions”, in *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., Curran Associates, Inc., 2017, pp. 4765–4774. [Online]. Available: <http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf>.
- [270] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, “LIBLINEAR: A library for large linear classification”, *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, 2008. [Online]. Available: <https://dl.acm.org/citation.cfm?id=1442794>.
- [271] D. Hyatt, G.-L. Chen, P. F. Locascio, M. L. Land, F. W. Larimer, and L. J. Hauser, “Prodigal: Prokaryotic gene recognition and translation initiation site identification.”, *BMC bioinformatics*, vol. 11, p. 119, Mar. 2010, ISSN: 1471-2105. DOI: 10.1186/1471-2105-11-119, epubliish.
- [272] D. B. Temel, P. Landsman, and M. L. Brader, “Chapter fourteen - orthogonal methods for characterizing the unfolding of therapeutic monoclonal antibodies: Differential scanning calorimetry, isothermal chemical denaturation, and intrinsic fluorescence with concomitant static light scattering”, in *Calorimetry*, ser. Methods in Enzymology, A. L. Feig, Ed., vol. 567, Academic Press, 2016, pp. 359–389. DOI: <https://doi.org/10.1016/bs.mie.2015.08.029>.

- [273] V. Kotov, K. Bartels, K. Veith, I. Josts, U. k. Tiruttani subhramanyam, C. Günther, J. Labahn, T. Marlovits, I. Moraes, H. Tidow, C. Löw, and M. Garcia Alai, “High-throughput stability screening for detergent-solubilized membrane proteins”, *Scientific Reports*, vol. 9, Dec. 2019. DOI: 10.1038/s41598-019-46686-8.
- [274] T. den Blaauwen, “Is longitudinal division in rod-shaped bacteria a matter of swapping axis?”, *Frontiers in Microbiology*, vol. 9, p. 822, 2018, ISSN: 1664-302X. DOI: 10.3389/fmicb.2018.00822. [Online]. Available: <https://www.frontiersin.org/article/10.3389/fmicb.2018.00822>.
- [275] N. Leisch, J. Verheul, N. R. Heindl, H. R. Gruber-Vodicka, N. Pende, T. den Blaauwen, and S. Bulgheresi, “Growth in width and ftsz ring longitudinal positioning in a gammaproteobacterial symbiont”, *Current Biology*, vol. 22, R831–R832, 2012, ISSN: 0960-9822. DOI: 10.1016/j.cub.2012.08.033.
- [276] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks”, in *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, P. L. Bartlett, F. C. N. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., 2012, pp. 1106–1114. [Online]. Available: <http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks>.
- [277] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba, “Learning deep features for discriminative localization”, in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, pp. 2921–2929. DOI: 10.1109/CVPR.2016.319.
- [278] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, “Unmasking Clever Hans predictors and assessing what machines really learn.”, *Nature communications*, vol. 10, p. 1096, 1 Mar. 2019, ISSN: 2041-1723. DOI: 10.1038/s41467-019-08987-4, epublish.

Book sources

- [4] M. T. Madigan, J. M. Martinko, P. V. Dunlap, and D. P. Clark, *Brock Biology of Microorganisms*, Twelfth edition, ser. Pearson international edition. 1301 Sansome Street, San Francisco, CA 94111: Pearson Education/Benjamin Cummings, 2009, ISBN: 9780132324601. [Online]. Available: <https://books.google.at/books?id=mwUpAQAAAJ>.
- [5] C. M. Bishop, *Pattern recognition and machine learning*, 5th Edition, ser. Information science and statistics. Springer, 2007, ISBN: 9780387310732. [Online]. Available: <http://www.worldcat.org/oclc/71008143>.
- [13] J. Berg, J. Tymoczko, and L. Stryer, *Biochemistry*, Fifth edition. New York: W H Freeman, 2002. [Online]. Available: <https://www.ncbi.nlm.nih.gov/books/NBK22342/>.
- [28] D. W. Mount, *Bioinformatics: sequence and genome analysis*. Second edition. Cold Spring Harbor, NY.: Cold Spring Harbor Laboratory Press, 2004, ISBN: 978-0-87969-608-5.
- [66] S. Marsland, *Machine Learning: An Algorithmic Perspective*, second edition, T. F. Gonzalez, J. Diaz-Herrera, and A. Tucker, Eds., ser. Machine Learning & Pattern Recognition. Chapman & Hall/CRC, 2014, ISBN: 978-1466583283.
- [70] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd Edition, ser. Springer Series in Statistics. Springer, 2009, ISBN: 9780387848570. DOI: 10.1007/978-0-387-84858-7.
- [80] J. Watson, T. Baker, S. Bell, A. Gann, M. Levine, and R. Losick, *Molecular Biology of the Gene*, Sixth edition, B. Wilbur, G. Carlson, M. Early, J. Inglis, and A. Gann, Eds. 1301 Sansome Street, San Francisco, CA 94111: Pearson Education/Benjamin Cummings, 2008, 978-0-321-50781-5, ISBN: 978-0-321-50781-5. [Online]. Available: <https://books.google.at/books?id=MByWPwAACAAJ>.

- [81] R. Bellman, *Adaptive Control Processes - A Guided Tour (Reprint from 1961)*, ser. Princeton Legacy Library. Princeton University Press, 2015, vol. 2045, ISBN: 978-1-4008-7466-8. DOI: 10.1515/9781400874668.
- [145] M. Minsky and S. Papert, *Perceptrons - an introduction to computational geometry*, Second edition. MIT Press, 1972, ISBN: 0-262-63022-2.
- [162] I. J. Goodfellow, Y. Bengio, and A. C. Courville, *Deep Learning*, ser. Adaptive computation and machine learning. MIT Press, 2016, ISBN: 978-0-262-03561-3. [Online]. Available: <http://www.deeplearningbook.org/>.

Online resources

- [53] *SIMAP2: The similarity matrix of protein*, 2014. [Online]. Available: <https://cube.univie.ac.at/resources/simap/>.
- [172] *OFAI Hub-Toolbox for Python3: Hubness analysis and reduction*. [Online]. Available: <https://github.com/OFAI/hub-toolbox-python3>.
- [183] Q. Chen, H. Wang, M. Li, G. Ren, S. Li, J. Zhu, J. Li, C. Liu, L. Zhang, and J. Wang, *Sptag: A library for fast approximate nearest neighbor search*, 2018. [Online]. Available: <https://github.com/Microsoft/SPTAG>.
- [217] *PhenDB: Prediction of bacterial phenotypes*. [Online]. Available: <https://phendb.csb.univie.ac.at/>.
- [218] *VOGDB: Virus orthologous groups*. [Online]. Available: <https://vogdb.csb.univie.ac.at/>.