universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Second-order stochastic non-convex optimization“

verfasst von / submitted by

Valentyn Boreiko BSc

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

Master of Science (MSc)

Wien, Oktober 2020 / Vienna, October 2020

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 821
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Mathematik / Mathematics
Betreut von / Supervisor:	Univ.-Prof. Dr. Radu Ioan Bot
Co-betreut von / Co-supervisor:	Univ.-Ass. Prof. Dr. Dan Alistarh

Abstract

The original Newton method in deterministic optimization has been known for decades and has since motivated many other more efficient algorithms. In this thesis, we focus on one of such improvements - cubic regularization of the Newton method (CR). In particular - on its extension to a non-convex fully stochastic setting - stochastic cubic regularization (SCR). In the theoretical part, we describe the mathematical foundation of the CR and the main results behind SCR. In the experimental part, we introduce some novel improvements for the SCR algorithm and show their performance on three well-known stochastic optimization problems of Deep Learning, one Linear regression problem, and one synthetic problem.

Zusammenfassung

Das ursprüngliche Newtonverfahren für die deterministische Optimierung ist seit Jahrzehnten bekannt und hat seitdem viele andere effizientere Algorithmen motiviert. In dieser Arbeit konzentrieren wir uns auf eine dieser Verbesserungen - die kubische Regularisierung des Newtonverfahrens (CR). Insbesondere - auf seiner Ausweitung auf eine nicht-konvexe vollständig stochastische Einstellung - stochastische kubische Regularisierung (SCR). Im theoretischen Teil beschreiben wir die mathematischen Grundlagen der CR und die Hauptergebnisse hinter SCR. Im experimentellen Teil stellen wir einige neuartige Verbesserungen für den SCR-Algorithmus vor und zeigen ihre Leistung bei drei bekannten stochastischen Optimierungsproblemen von Deep Learning, einem linearen Regressionsproblem und einem synthetischen Problem.

Contents

1 Introduction

This thesis is based on the paper [1], where the stochastic modification of the cubic regularized Newton method [2] for non-convex functions (SRC) is proposed. Therefore after the section with motivation we will start the detailed overview of the second paper.
After this we will discuss the first paper ([1]) and give the intuition behind its main results. In the end, we will introduce the novel improvements to the SCR and compare their performance on several datasets against popular stochastic optimizers.

2 Notations and basic definitions

For a symmetric real-valued $n \times n$ matrix H, its spectrum is denoted by $\{\lambda_i(H)\}_{i=1}^n$. We assume that the eigenvalues are ordered decreasingly:

$$\lambda_{max}(H) := \lambda_1(H) \geq ... \geq \lambda_n(H) =: \lambda_{min}(H).$$

Therefore, H symmetric is positive semi-definite if and only if $\lambda_n(H) \geq 0$. Furthermore, we use the standard spectral matrix norm for a symmetric positive definite matrix A:

$$\|A\| = \sqrt{\lambda_1(A^T A)},$$

and the standard Euclidean norm $\|\cdot\|$ for a vector. A generalized A-weighted vector norm for a symmetric positive semi-definite matrix A and a vector x we will denote by $\|x\|_A := \sqrt{x^T A x}$.
The interior and the boundary of a set Q are denoted respectively as $\operatorname{int} Q$ and ∂Q. Furthermore, we denote $(x)_+ := \max\{0, x\}$.
For the analysis of the stochastic case, we denote $f(m) \in \widetilde{\mathcal{O}}(g(m))$ as a shorthand for $f(m) \in \mathcal{O}(g(m) log^l g(m))$ for some l.

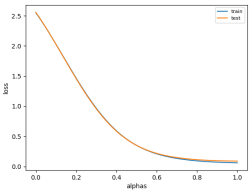
Definition 2.1. *For an objective function $J : \mathbb{R}^n \to \mathbb{R}$, we will call its evaluation at points $(1-\alpha)x_0+\alpha x_1$ for $\alpha \in [0,1], x_0, x_1 \in \mathbb{R}^n$ as its cross-section at $(1-\alpha)x_0+\alpha x_1$.*

Definition 2.2 (ϵ-second-order stationary point)**.** *Let f be twice differentiable with L-Lipschitz Hessian. We will call x as an ϵ-second-order stationary point (or an ϵ-approximate local minimum) of f if for $\epsilon > 0$ we have*

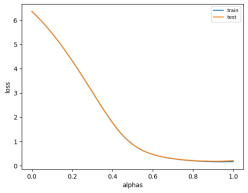
$$\|f'(x)\| \leq \epsilon \quad \textit{and} \quad \lambda_{min}(f''(x)) \geq -\sqrt{L\epsilon}.$$

3 Motivation

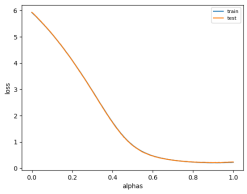
As motivation, based on [4], we look at the cross-section of the objective function J at $(1-\alpha)x_0 + \alpha x_1$, where x_0, x_1 are respectively starting and ending values of the set of parameters of a neural network (NN) at some time interval. This shows that, by considering NNs that perform well, this cross-section looks well-behaved with monotonically decreasing derivative.
This result motivates us to develop better optimization techniques for enhanced architectures, which would find the right direction - and the optimization problem would be solved with a coarse line search.
Below we show the results for several architectures and two datasets - MNIST ([53]) and CIFAR-10 ([54]) (we will call it CIFAR further for simplicity):



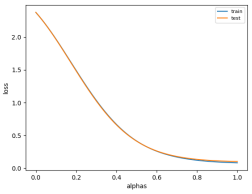
(a) MNIST with maxout network and cross-entropy loss



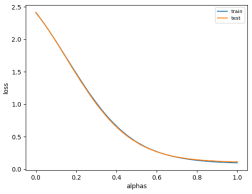
(b) MNIST with maxout network and adversarial cross-entropy loss



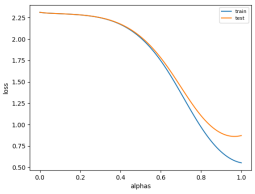
(c) MNIST with maxout network with dropout and adversarial cross-entropy loss



(d) MNIST with rectified linear (RELU) network and cross-entropy loss



(e) MNIST with RELU network and adversarial cross-entropy loss



(f) CIFAR with convolutional neural network (CNN) and cross-entropy loss

Figure 1: Cross-section of J at $(1-\alpha)x_0 + \alpha x_1$.

The model specifications are the same as in [4].
Even though noisy SGD methods already allow to avoid saddle points ([25], [26], [27],

[28]), we want to investigate techniques that escape saddle points and also converge faster and therefore use second-order information explicitly (and not implicitly like in [28]). We are aware however that a recent SGD5 algorithm ([29]) using many tricks has achieved SOA run-time complexity on par with the best second-order methods ([1], [5]).

4 Cubic regularization (CR) of Newton method and its global performance

4.1 Introduction

We will look at the following aspects in this section:

- analysis of the cubic regularization for unconstrained optimization,
- proof of the general local convergence.

Which issues can a normal Newton scheme have:

- Hessian may be degenerate at a point - the method is not well-defined then,
- it can diverge or converge to a saddle point, or to a local maximum.

To solve it, many approaches have been proposed ([10]).
Most of them combine the following ideas:

- *Levenberg-Marquardt regularization* ([9]) - if $f''(x)$ is not positive definite, regularize it with a multiple of unit matrix with $\gamma \geq 0$:

$$x_{k+1} = x_k - \big[f''(x_k) + \gamma I\big]^{-1} f'(x_k);$$

- *line search - damped* Newton method ([11]) with the step size $h_k > 0$ to form a monotone sequence ($f(x_{k+1}) \leq f(x_k)$):

$$x_{k+1} = x_k - h_k\big[f''(x_k)\big]^{-1} f'(x_k); \tag{1}$$

- *trust-region approach* ([10]) - select the next point x_{k+1} from the trust region of x_k (e.g., $\Delta(x_k) = B_\epsilon(x_k)$ - a ball centered at x_k of radius $\epsilon > 0$):

$$x_{k+1} = \underset{x \in \Delta(x_k)}{\arg\min} \big[\langle f'(x_k), x - x_k\rangle + \frac{1}{2}\langle f''(x_k)(x - x_k), x - x_k\rangle\big].$$

We focus on the questions related to the worst-case guarantees for global behavior of the second-order schemes and show that theoretically, a correctly chosen Newton-type scheme outperforms the gradient scheme (considering only the number of iterations).

4.2 Cubic regularization

The idea of the cubic regularization is to modify Newton method using technique similar to the *gradient mapping* ([12]).
The gradient mapping "replaces" the gradient in case of constrained minimization.

Definition 4.1 (Gradient mapping)**.** *Let $\gamma > 0$. Denote*

$$x_Q(\overline{x};\gamma) = \arg\min_{x\in Q}\left[f(\overline{x}) + \langle f'(\overline{x}), x-\overline{x}\rangle + \frac{\gamma}{2}\|x-\overline{x}\|^2\right],$$
$$g_Q(\overline{x};\gamma) = \gamma(\overline{x} - x_Q(\overline{x};\gamma)).$$

*Then $g_Q(\overline{x};\gamma)$ is the **gradient mapping** of f on Q. It is well defined for Q closed convex, as the argument of* arg min *above is a strongly convex function.*
For $Q := \mathbb{R}^n$, $\frac{1}{\gamma}$ can be seen as the "'gradient" step size:

$$x_Q(\overline{x};\gamma) = \overline{x} - \frac{1}{\gamma}f'(\overline{x}), \quad g_Q(\overline{x};\gamma) = f'(\overline{x}).$$

We can do so for the second-order approximation as well. First, we need the following assumption and result.

Assumption 4.1. *Let from now on the Hessian of the twice differentiable objective function f be L-Lipschitz for some $L > 0$ and for $Q \subseteq \mathbb{R}^n$ closed convex set with non-empty interior with $\mathcal{L}(f(x_0)) := \{x \in \mathbb{R}^n \mid f(x) \le f(x_0)\} \subseteq int\, Q$:*

$$\|f''(x) - f''(y)\| \le L\|x-y\|, \quad \forall x, y \in Q,$$

where $x_0 \in int\, Q$ is a starting point of the iterative scheme that will be introduced in this section.

Lemma 4.1. *Given Assumption 4.1, we have that for all $x, y \in Q$ the inequalities*

$$\|f'(y) - f'(x) - f''(x)(y-x)\| \le \frac{L}{2}\|y-x\|^2,$$

$$|f(y) - f(x) - \langle f'(x), y-x\rangle - \frac{1}{2}\langle f''(x)(y-x), y-x\rangle| \le \frac{L}{6}\|y-x\|^3$$

hold.

Proof. Regarding the first inequality, we have

$$\begin{aligned}\|f'(y) - f'(x) - f''(x)(y-x)\| &= \left\| \int_0^1 [f''(x+\alpha(y-x)) - f''(x)](y-x)\, d\alpha \right\| \\ &\leq L\|y-x\|^2 \int_0^1 \alpha\, d\alpha = \frac{L}{2}\|y-x\|^2.\end{aligned}$$

Regarding the second inequality, we have

$$f(y) = f(x) + \langle f'(x), y-x\rangle + \int_0^1 \int_0^\tau \langle f''(x+\alpha(y-x))(y-x), y-x\rangle\, d\alpha\, d\tau.$$

Since f'' is $L-$Lipschitz, it follows from the equation above that

$$\begin{aligned}|f(y) - f(x) - \langle f'(x), y-x\rangle - \frac{1}{2}\langle f''(x)(y-x), y-x\rangle| \\ \leq \Big| \int_0^1 \int_0^\tau \langle f''(x+\alpha(y-x))(y-x), y-x\rangle - \langle f''(x)(y-x), y-x\rangle\, d\alpha\, d\tau \Big| \\ \leq \frac{L}{6}\|y-x\|^3.\end{aligned}$$

□

One important consequence of this proposition is that we can approximate our objective function from above using the cubic approximation of the function.

Corollary 4.1. *The auxiliary function (or cubic approximation of the function)*

$$\xi_{2,x,M}(y) = f(x) + \langle f'(x), y-x\rangle + \frac{1}{2}\langle f''(x)(y-x), y-x\rangle + \frac{M}{6}\|y-x\|^3$$

for some $M > 0$ will be an upper second-order approximation for the objective function (for $Q := \mathbb{R}^n$):

$$f(y) \leq \xi_{2,x,L}(y) \quad \forall y \in \mathbb{R}^n.$$

Note that L indicates again the Hessian Lipschitz constant.

Definition 4.2 (CR of Newton method)**.** *The modified Newton scheme consists in iteratively solving the following auxiliary global minimization problem:*

$$T_M(x) \in \arg\min_{y\in\mathbb{R}^n} \xi_{2,x,M}(y). \tag{2}$$

It is called ***CR of Newton method.***

Remark 4.1. *Note that, according to the definition of T_M above, the point $y = T_M(x)$ satisfies*

$$f'(x) + f''(x)(y - x) + \frac{1}{2}M \|y - x\| (y - x) = 0$$

for $y \neq x$. Multiplying it by $T_M(x) - x$ we get

$$\langle f'(x), T_M(x) - x\rangle + \langle f''(x)(T_M(x) - x), T_M(x) - x\rangle + \frac{1}{2}Mr_M^3(x) = 0,$$

where we denote $r_M(x) := \|T_M(x) - x\|$.

Remark 4.2. *Note that the problem (2) is non-convex and it can have many local minima. Therefore it is important to simplify it.*

We will now describe a possible simplification of (2) using a duality argument.

4.3 Weak duality argument

We can simplify the auxiliary minimization problem (2) by solving its one-dimensional dual problem as it is shown in theorem below.
First, let us introduce the following definitions.

Definition 4.3. *Let $g \in \mathbb{R}^n$ and $H \in \mathbb{R}^{n\times n}$ be a real symmetric matrix.*

$$v_{u,M}(h) := \langle g, h\rangle + \frac{1}{2}\langle Hh, h\rangle + \frac{M}{6}\|h\|^3 \quad \forall h \in \mathbb{R}^n,$$
$$\mathcal{D} := \{r \in \mathbb{R} \,:\, H + \frac{M}{2}rI \succ 0, r \geq 0\},$$
$$v_{l,M}(r) := -\frac{1}{2}\langle (H + \frac{Mr}{2}I)^{-1}g, g\rangle - \frac{M}{12}r^3 \quad \forall r \in \mathcal{D}.$$

Here we let $r \geq 0$ as it is done in Levenberg-Marquardt method.

Theorem 4.1. *For any $M > 0$ we have the following weak duality result:*

$$\inf_{h\in\mathbb{R}^n} v_{u,M}(h) \geq \sup_{r\in\mathcal{D}} v_{l,M}(r). \tag{3}$$

Moreover, for any $r \in \mathcal{D}$, direction $h(r) = -(H + \frac{Mr}{2}I)^{-1}g$ satisfies the inequality

$$0 \leq v_{u,M}(h(r)) - v_{l,M}(r) = \frac{M}{12}(r + 2\|h(r)\|)(\|h(r)\| - r)^2 = \frac{4}{3M}\frac{r + 2\|h(r)\|}{(r + \|h(r)\|)^2}v_l'(r)^2.$$

Proof.

$$\begin{aligned}
\inf_{h\in\mathbb{R}^n} v_{u,M}(h) &= \inf_{h\in\mathbb{R}^n}\Big[\langle g,h\rangle + \frac{1}{2}\langle Hh,h\rangle + \frac{M}{6}\|h\|^3\Big] \\
&= \inf_{\substack{h\in\mathbb{R}^n,\\ \tau=\|h\|^2}}\Big[\langle g,h\rangle + \frac{1}{2}\langle Hh,h\rangle + \frac{M}{6}(\tau)_+^{3/2}\Big] \\
&= \inf_{\substack{h\in\mathbb{R}^n,\\ \tau\in\mathbb{R}}}\sup_{r\in\mathbb{R}}\Big[\langle g,h\rangle + \frac{1}{2}\langle Hh,h\rangle + \frac{M}{6}(\tau)_+^{3/2} + \frac{M}{4}r\Big(\|h\|^2-\tau\Big)\Big] \\
&\geq \sup_{r\in\mathcal{D}}\inf_{\substack{h\in\mathbb{R}^n,\\ \tau\in\mathbb{R}}}\Big[\langle g,h\rangle + \frac{1}{2}\langle Hh,h\rangle + \frac{M}{6}(\tau)_+^{3/2} + \frac{M}{4}r\Big(\|h\|^2-\tau\Big)\Big] = \sup_{r\in\mathcal{D}} v_{l,M}(r).
\end{aligned}$$

We have as well that

$$\begin{aligned}
v_{u,M}(h(r)) &= \langle g,h(r)\rangle + \frac{1}{2}\langle Hh(r),h(r)\rangle + \frac{M}{6}\|h(r)\|^3 \\
&= -\frac{1}{2}\langle Hh(r),h(r)\rangle - \frac{M}{2}r\|h(r)\|^2 + \frac{M}{6}\|h(r)\|^3 \\
&= -\frac{1}{2}\langle\Big(H+\frac{Mr}{2}I\Big)h(r),h(r)\rangle - \frac{M}{4}r\|h(r)\|^2 + \frac{M}{6}\|h(r)\|^3 \\
&= v_{l,M}(r) + \frac{M}{12}r^3 - \frac{M}{4}r\|h(r)\|^2 + \frac{M}{6}\|h(r)\|^3 \\
&= v_{l,M}(r) + \frac{M}{12}(r+2\|h(r)\|)(\|h(r)\|-r)^2,
\end{aligned}$$

which proves the desired statement. From it we see that the equality in (3) is attained for $r=\|h(r)\|$. Furthermore, we can solve in this case the dual problem (right-hand side of (3)) by solving the following system of one equation and one inequality

$$r = \left\|\Big(H+\frac{Mr}{2}I\Big)^{-1}g\right\|, \quad r\geq\frac{2}{M}(-\lambda_n(H))_+, \tag{4}$$

which is derived from the definition of the set $\mathcal{D}$ and necessary first-order local optimality condition for the dual problem over the set $\mathcal{D}$:

$$\begin{aligned}
0 = v'_{l,M}(r) &= -\frac{M}{4}r^2 - Hh(r)h'(r) - \frac{M}{4}(2h(r)h'(r)r + \|h\|^2) \\
&= -\frac{M}{4}r^2 + \frac{M}{2}(H+\frac{Mr}{2}I)(H+\frac{Mr}{2}I)^{-1}\|h(r)\|^2 - \frac{M}{4}\|h(r)\|^2 \\
&= \frac{M}{4}(\|h(r)\|^2 - r^2),
\end{aligned} \tag{5}$$

where $r \in \mathcal{D}$ and $r > 0$. This follows from the basic first-order optimality condition for our dual constrained problem (see e.g. [58], Theorem 6.12), which states that in order for r^* to be a locally optimal point of our dual problem, it has to satisfy

$$0 \in -\nabla v_{l,M}(r^*) + N_{\mathcal{D}}(r^*),$$

where $N_{\mathcal{D}}(r^*)$ is the normal cone of $\mathcal{D}$ at r^*. Moreover, for $r^* \in \mathcal{D}$ and $r^* > 0$ we have that $N_{\mathcal{D}}(r^*) = 0$ as in that case $r^* \in \operatorname{int} \mathcal{D}$.

□

Remark 4.3. *Note that there is indeed such an r that solves the system (4). For this, it is enough to write this system in the basis of the eigenvectors of matrix H and solve depending on the last coordinate of g in the new basis being equal to zero or no. More details regarding it can be found in Section 5 of [2].*

So the original problem is reduced to one-dimensional equation and inequality. Techniques from trust region methods can be used to solve them ([10], Chapter 7).

Remark 4.4. *Another approach to solving the minimization problem (2) can be also expressed as an equivalent two-dimensional constrained convex minimization problem:*

$$\min_{h \in \mathbb{R}^n} \left[\langle g, h \rangle + \frac{1}{2} \langle Hh, h \rangle + \frac{M}{6} \|h\|^3 \right] \equiv \min_{h \in \mathbb{R}^n} \left[\xi_1(h) + \frac{M}{6} \xi_2^{3/2}(h) \right] \equiv \min_{\xi \in K} \phi(\xi),$$

where

$$\begin{aligned} \xi_1(h) = \langle g, h \rangle + \frac{1}{2} \langle Hh, h \rangle, \quad \xi_2(h) = \|h\|^2, \\ K = \{ \xi = (\xi_1(h), \xi_2(h))^T : h \in \mathbb{R}^n \} \subset \mathbb{R}^2, \\ \phi(\xi) = \xi_1(h) + \frac{M}{6} (\xi_2(h))_+^{3/2}, \end{aligned}$$

and for $n \geq 2$ the set K is convex and closed (see [33], Theorem 2.2).

4.4 Algorithm for CR of Newton method

Let us describe the respective algorithm for the problem of interest in this section:

$$\min_{y \in \mathbb{R}^n} f(y),$$

where the objective function f satisfies the Assumption 4.1. Further, we can consider the criterion of local optimality μ_M for $M > 0$:

$$\mu_M(x) = \max\left\{\sqrt{\frac{2}{L+M}\|f'(x)\|}, -\frac{2}{2L+M}\lambda_n(f''(x))\right\}. \tag{6}$$

Let us now show the results that will help to explain the analytical form of $\mu_M(x)$:

Proposition 4.1. *Denote $r_M(x) = \|T_M(x) - x\|$. Then for any $x \in Q$ (from Assumption 4.1) we have*

$$f''(x) + \frac{1}{2}Mr_M(x)I \succcurlyeq 0.$$

Proof. Following the notation of [17], define $s := T_M(x) - x$ and the matrix

$$A_s := f''(x) + \frac{1}{2}M\|s\| I.$$

Because, according to Definition (2), s is the global minimizer of the auxiliary function $\xi_{2,x,M}(\overline{s}) = f(x) + \langle f'(x), \overline{s}\rangle + \frac{1}{2}\langle f''(x)\overline{s}, \overline{s}\rangle + \frac{M}{6}\|\overline{s}\|^3$, the necessary first-order optimality condition holds:

$$\nabla\xi_{2,x,M}(s) = f'(x) + A_s s = 0.$$

This allows us to express the auxiliary function $\xi_{2,x,M}$ at s as

$$\xi_{2,x,M}(s) = f(x) - \frac{1}{2}\langle A_s s, s\rangle - \frac{M}{12}\|s\|^3$$

and at any $y \in \mathbb{R}^n$ alternatively as:

$$\begin{aligned}
\xi_{2,x,M}(s) + \frac{1}{2}\langle A_s(y-s), (y-s)\rangle + \frac{M}{12}(\|s\| - \|y\|)^2(\|s\| + 2\|y\|) \\
&= f(x) - \frac{1}{2}\langle A_s s, s\rangle + \frac{1}{2}\langle A_s s, s\rangle + \frac{1}{2}\langle A_s y, y\rangle - \frac{1}{2}\langle A_s s, y\rangle \\
&- \frac{1}{2}\langle A_s y, s\rangle - \frac{M}{12}\|s\|^3 + \frac{M}{12}\|s\|^3 + \frac{M}{6}\|y\|^3 + \frac{M}{6}\|s\|^2\|y\| \\
&+ \frac{M}{12}\|y\|^2\|s\| - \frac{M}{6}\|s\|^2\|y\| - \frac{M}{3}\|s\|\|y\|^2 \\
&= \xi_{2,x,M}(y) + \frac{1}{2}\langle A_s s, y\rangle - \frac{1}{2}\langle A_s y, s\rangle = \xi_{2,x,M}(y),
\end{aligned}$$

where the last equality follows, because A_s represents a self-adjoint operator, as it is symmetric. Let $d \in \mathbb{R}^n \setminus \mathbf{0}$. Then, because s being a global minimizer of $\xi_{2,x,M}$

implies $\xi_{2,x,M}(y) - \xi_{2,x,M}(s) \geq 0$ for any $y \in \mathbb{R}^n$ and thus also for $y = s + td$ for all $t \in \mathbb{R}$, we have

$$\begin{aligned}
&\frac{t^2}{2}\langle A_s d, d\rangle + \frac{M}{12}(\|s\| - \|s+td\|)^2(\|s\| + 2\|s+td\|) \\
&\qquad = \frac{t^2}{2}\langle A_s d, d\rangle + \frac{M}{12}(t^2\|d\|^2 + 2t\langle s, d\rangle)(\|s\| + 2\|s+td\|) \\
&\qquad \geq 0,
\end{aligned}$$

which implies that

$$\langle A_s d, d\rangle + \frac{M}{6t}(t\|d\|^2 + 2\langle s, d\rangle)(\|s\| + 2\|s+td\|) \geq 0 \quad \forall t \in \mathbb{R} \setminus \mathbf{0}.$$

By letting $t \to -\frac{2\langle s,d\rangle}{\|d\|^2}$ we have that

$$\langle A_s d, d\rangle \geq 0,$$

which gives the desired statement.

□

Lemma 4.2. *If $T_M(x) \in Q$ (from the Assumption 4.1), then*

$$\|f'(T_M(x))\| \leq \frac{1}{2}(L+M)r_M^2(x).$$

Proof. From Remark 4.1, we get

$$\|f'(x) + f''(x)(T_M(x) - x)\| = \frac{1}{2}Mr_M^2(x).$$

Using the first inequality in Lemma 4.1, we obtain

$$\|f'(T_M(x)) - f'(x) - f''(x)(T_M(x) - x)\| \leq \frac{1}{2}Lr_M^2(x).$$

Combining the above relations and the triangle inequality gives us the desired result.

□

An analytical form of $\mu_M(x)$ is derived then from the following result:

Lemma 4.3. *For any $x \in Q$ we have $\mu_M(T_M(x)) \leq r_M(x)$.*

Proof. First, we obtain the following inequalities (using the Löwner partial order):

$$f''(T_M(x)) \geq f''(x) - Lr_M(x)I \geq -(\frac{1}{2}M + L)r_M(x)I, \tag{7}$$

where the first inequality follows from the Assumption 4.1, as

$$\|f''(x) - f''(T_M(x))\| \leq L\,\|T_M(x) - x\| = Lr_M(x).$$

The second one follows directly from the Proposition 4.1. The statement of the lemma follows then directly by Lemma 4.2, which bounds the first term in (6) and by the inequality above, which bounds the second term in (6).

□

This criterion vanishes when x is a local minimum.
Denote further

$$\overline{f}_M(x) := \min_{y\in\mathbb{R}^n}\left[f(x) + \langle f'(x), y - x\rangle + \frac{1}{2}\langle f''(x)(y-x), y-x\rangle + \frac{M}{6}\|y-x\|^3\right].$$

Algorithm is then as follows:

Algorithm 4.1: CR of Newton method

Input : f satisfying the Assumption 4.1, local optimality condition threshold ϵ and maximal allowed number of iterations K

Output: A sequence (x_i)

$(x_0, k, L_0) \leftarrow (\in \mathbb{R}^n, 0, \in (0, L])$;

do

  Find $M_k \in [L_0, 2L]$ such that $f(T_{M_k}(x_k)) \leq \overline{f}_{M_k}(x_k)$;

  $x_{k+1} \leftarrow T_{M_k}(x_k)$;

  $k \leftarrow k + 1$;

while $\mu_{M_k}(x_{k+1}) > \epsilon$ ***AND*** $k < K$;

return (x_i);

Moreover, as we will see, with the help of the following results, Algorithm 4.1 generates a monotone sequence.

Lemma 4.4. *For any $x \in Q$, f satisfying the Assumption 4.1, and such that $f(x) \leq f(x_0)$ for the starting point $x_0 \in \mathrm{int}\,Q$ of Algorithm 4.1, we have the following result:*

$$\langle f'(x), x - T_M(x)\rangle \geq 0. \tag{8}$$

Furthermore, if $M \geq \frac{2}{3}L$ and $x \in \mathrm{int}\,Q$, then $T_M(x) \in \mathcal{L}(f(x)) \subseteq \mathrm{int}\,Q$.

Proof. First, multiplication of the result from the Proposition 4.1 by $x - T_M(x)$ twice, gives us

$$\langle f''(x)(T_M(x) - x), T_M(x) - x\rangle + \frac{1}{2}Mr_M^3(x) \geq 0.$$

Therefore the desired inequality (8) follows from Remark 4.1.
Regarding the second statement, let $M \geq \frac{2}{3}L$ together with $x \in \operatorname{int} Q$ and assume by contradiction that $T_M(x) \notin Q$, and thus also $T_M(x) \notin \mathcal{L}(f(x))$. I.e., $f(T_M(x)) > f(x)$ and thus $r_M(x) = \|x - T_M(x)\| > 0$ as f is a function, and thus is well-defined. Consider now the convex hull of two points - x and $T_M(x)$:

$$y_\alpha = (1 - \alpha)x + \alpha T_M(x), \quad \alpha \in [0, 1].$$

Since, by assumption, $y_0 = x \in \operatorname{int} Q$, the mapping of some $\overline{\alpha} \in (0, 1)$ to the boundary:

$$\overline{\alpha} \mapsto \ y_{\overline{\alpha}} \in \partial Q$$

is well-defined as y_α is an interval that connects two points, one of which, x, is inside of $\operatorname{int} Q$ and another one, $T_M(x)$, is outside of Q (and thus is in the exterior of Q as Q is closed), by the assumption.
According to the assumption, $\overline{\alpha} < 1$ and $y_\alpha \in Q$ for all $\alpha \in [0, \overline{\alpha}]$.
Lemma 4.1 gives us for all $\alpha \in [0, 1]$

$$|f(y_\alpha) - f(x) - \langle f'(x), y_\alpha - x\rangle - \frac{1}{2}\langle f''(x)(y_\alpha - x), y_\alpha - x\rangle| \leq \frac{L}{6}\|y_\alpha - x\|^3 = \frac{\alpha^3 L}{6}r_M(x)^3.$$

Furthermore, using Remark 4.1, and the proven inequality (8) with the given assumption ($L \leq \frac{3}{2}M$), we have for all $\alpha \in [0, 1]$:

$$\begin{aligned} f(y_\alpha) &\leq f(x) + \langle f'(x), y_\alpha - x\rangle + \frac{1}{2}\langle f''(x)(y_\alpha - x), y_\alpha - x\rangle + \frac{\alpha^3 L}{6}r_M(x)^3 \\ &\leq f(x) + (\alpha - \frac{\alpha^2}{2})\langle f'(x), T_M(x) - x\rangle - \frac{\alpha^2(1-\alpha)}{4}Mr_M(x)^3 \\ &\leq f(x) - \frac{\alpha^2(1-\alpha)}{4}Mr_M(x)^3. \end{aligned}$$

Therefore, $f(y_{\overline{\alpha}}) < f(x)$, and thus $y_{\overline{\alpha}} \in \operatorname{int}\mathcal{L}(f(x)) \subseteq \operatorname{int} Q$, which is a contradiction to $y_{\overline{\alpha}} \in \partial Q$ being well-defined. Thus, $T_M(x) \in Q$. This allows us to use 4.1 for $T_M(x), x \in Q$:

$$\begin{aligned} |f(T_M(x)) - f(x) - \langle f'(x), T_M(x) - x\rangle - \frac{1}{2}\langle f''(x)(T_M(x) - x), T_M(x) - x\rangle| \\ \leq \frac{L}{6}\|T_M(x) - x\|^3 = \frac{L}{6}r_M(x)^3. \end{aligned}$$

Using again Remark 4.1, and the proven inequality (8) with the given assumption ($L \leq \frac{3}{2}M$), we have

$$\begin{aligned} f(T_M(x)) &\leq f(x) + \langle f'(x), T_M(x) - x\rangle + \frac{1}{2}\langle f''(x)(T_M(x) - x), T_M(x) - x\rangle + \frac{L}{6}r_M(x)^3 \\ &\leq f(x) + \frac{1}{2}\langle f'(x), T_M(x) - x\rangle \\ &\leq f(x). \end{aligned}$$

This implies that $T_M(x) \in \mathcal{L}(f(x)) \subseteq \operatorname{int} Q$ according to Assumption 4.1.

□

Lemma 4.5. *For any $x \in Q$ we have*

$$\overline{f}_M(x) \leq \min_{y \in Q}[f(y) + \frac{L+M}{6}\|y - x\|^3],$$

$$f(x) - \overline{f}_M(x) \geq \frac{M}{12}r_M^3(x).$$

Moreover, if $M \geq L$, and $x \in \operatorname{int} Q$ then $T_M(x) \in \operatorname{int} Q$ and

$$f(T_M(x)) \leq \overline{f}_M(x).$$

Proof. Using Lemma 4.1 and the definition of $\overline{f}_M(x)$ we get the first inequality. Further, applying the definition of $T_M(x)$, Remark 4.1, and the inequality from Lemma 4.4, we have

$$\begin{aligned} f(x) - \overline{f}_M(x) &= \langle f'(x), x - T_M(x)\rangle - \frac{1}{2}\langle f''(x)(T_M(x) - x), T_M(x) - x\rangle - \frac{M}{6}r_M^3(x) \\ &= \frac{1}{2}\langle f'(x), x - T_M(x)\rangle + \frac{M}{12}r_M^3(x) \\ &\geq \frac{M}{12}r_M^3(x). \end{aligned}$$

Finally, if $M \geq L \geq \frac{2}{3}L$, then, using Lemma 4.4, we have that $T_M(x) \in \operatorname{int} Q$. This allows us to apply Lemma 4.1 to the definition of $\overline{f}_M$ to get the last inequality:

$$f(T_M(x)) \leq \overline{f}_M(x).$$

□

Remark 4.5. *Since, according to Lemma 4.5, $f(x_k) - \overline{f}_{M_k}(x_k) \geq \frac{M_k}{12} r^3_{M_k}(x_k)$, and thus $f(T_{M_i}(x_k)) \leq \overline{f}_{M_k}(x_k) \leq f(x_k)$ in Q for $M_k = L$, the sequence of the function f evaluations at the sequence generated by Algorithm 4.1 is monotone in Q:*

$$f(x_{k+1}) \leq f(x_k) \quad \forall k \geq 0$$

for $x_k \in intQ$ and $x_{k+1} = T_{M_k}(x_k) \in intQ$. Note that the important here part of Assumption 4.1 was that $\mathcal{L}(f(x_0)) := \{x \in \mathbb{R}^n \mid f(x) \leq f(x_0)\} \subseteq intQ$, as it allows all consequent iterates to stay in the same set $intQ$ and thus allows to use for them Lemma 4.4.

4.5 General convergence results

Now let us prove the theorem that helps to get further convergence results. Then we present one of such situations.

Theorem 4.2. *Let the sequence (x_k) be generated by Algorithm 4.1 with $K := \infty$. Further, assume that the objective function is bounded below:*

$$f(x) \geq f^* \quad \forall x \in Q.$$

Then $\sum_{k=0}^{\infty} r^3_{M_k}(x_k) \leq \frac{12}{L_0}(f(x_0) - f^*)$, for L_0 as in Algorithm 4.1, $\lim_{k\to\infty} \mu_L(x_k) = 0$ and for any $m \geq 1$ we get

$$\min_{1\leq k\leq m} \mu_L(x_k) \leq \frac{8}{3}\Big(\frac{3(f(x_0) - f^*)}{2mL_0}\Big)^{1/3}.$$

Proof. Lemma 4.5 together with the Algorithm 4.1 gives us for all $k \geq 0$:

$$f(x_k) - f(x_{k+1}) = f(x_k) - f(T_{M_k}(x)) \geq f(x) - \overline{f}_{M_k}(x) \geq \frac{M_k}{12} r^3_{M_k}(x).$$

Therefore, using telescoping sum, we get for any $m \geq 1$:

$$f(x_0) - f^* \geq f(x_0) - f(x_m) = \sum_{k=0}^{m-1} (f(x_k) - f(x_{k+1})) \geq \sum_{k=0}^{m-1} \frac{M_k}{12} r^3_{M_k}(x_k) \geq \frac{L_0}{12} \sum_{k=0}^{m-1} r^3_{M_k}(x_k)$$

due to the bounds in Algorithm 4.1 on M_k for all $k \geq 0$:

$$L_0 \leq M_k \leq 2L.$$

Letting $m \to \infty$, we get the first inequality.
Further, note that the for $1 \leq k \leq m$ the following holds:

$$\begin{aligned}
\mu_{M_k}(x_k) &= \max\left\{\sqrt{\frac{2}{L+M_k}\,\|f'(x_k)\|}, -\frac{2}{2L+M_k}\lambda_n(f''(x_k))\right\} \\
&\geq \max\left\{\sqrt{\frac{2}{3L}\,\|f'(x_k)\|}, -\frac{1}{L}\lambda_n(f''(x_k))\right\} \\
&\geq \min\left\{\sqrt{\frac{2}{3}}, \frac{3}{2}\right\} \max\left\{\sqrt{\frac{1}{L}\,\|f'(x_k)\|}, -\frac{2}{3L}\lambda_n(f''(x_k))\right\} \\
&= \sqrt{\frac{2}{3}}\mu_L(x_k) \\
&\geq \frac{3}{4}\mu_L(x_k).
\end{aligned}$$

Now, use it and Lemma 4.3 to get the second desired inequality by

$$\frac{3}{4}\min_{1\leq k\leq m}\mu_L(x_k) \leq \min_{1\leq k\leq m}\mu_{M_k}(x_k) \leq \min_{1\leq k\leq m} r_{M_k}(x_{k-1}) \leq 2\Big(\frac{3(f(x_0)-f^*)}{2mL_0}\Big)^{1/3},$$

where the third inequality is due to $\mu_{M_k}(x_k) \leq r_{M_k}(x_{k-1})$ for $1 \leq k \leq m$ from Lemma 4.3. □

Remark 4.6. *From the second inequality of Theorem 4.2 and the definition of μ_M for $M > 0$ we get that*

$$\min_{1\leq k\leq m}\|f'(x_k)\| \leq \mathcal{O}(m^{-2/3}),$$

compared to gradient descent with the right-hand side (RHS) being $\mathcal{O}(m^{-1/2})$. (e.g., [12], inequality (1.2.14)). For further comparison of gradient descent and Algorithm 4.1 for non-convex optimization refer to [6].

This theorem gives us global efficiency estimate in the second inequality. Moreover, one global convergence result for Algorithm 4.1 follows directly from it.

Theorem 4.3. *Let the sequence (x_k) be generated by Algorithm 4.1 with $K := \infty$ and $M_k \geq L$. For some $k \geq 0$, assume $\mathcal{L}(f(x_k))$ to be bounded. Then there exists a limit*

$$\lim_{k\to\infty} f(x_k) = f^*.$$

The set X^* of the limit points of this sequence is non-empty. Moreover, it is a compact connected set, such that for any $x^* \in X^*$ we have

$$f(x^*) = f^*, \quad f'(x^*) = 0, \quad f''(x^*) \succcurlyeq 0.$$

Proof. Because the sequence $(f(x_k))$ of real numbers is monotone by Remark 4.5 and is bounded from below by f^*, then, by monotone convergence theorem, it is convergent as well to f^*, i.e. $\lim_{k\to\infty} f(x_k) = f^*$. Moreover, as twice differentiability at a point implies continuity at a point, we have that for an arbitrary limit point $x^* \in X^*$ of the sequence (x_k): $f(x^*) = \lim_{k\to\infty} f(x_k) = f^*$.
The set of limit points of the sequence is indeed non-empty as the sequence (x_k) is assumed to be bounded by the boundedness of the sublevel sets. It is moreover compact as it is equal to $\cap_{m\geq 0}\overline{\cup_{k\geq m}\{(x_k)\}}$, which is compact, as a countable intersection of bounded and closed sets. The proof that it is also a connected set is analogous to the proof of Lemma 5 in [46].
Finally, for any limit point x^* of the sequence (x_k) it holds that $f'(x^*) = 0, f''(x^*) \succcurlyeq 0$ by one of the result of Theorem 4.2: $\lim_{k\to\infty} \mu_L(x_k) = 0$.

Remark 4.7. *Note that theorem above just tells us that all limit points of the sequence (x_k) generated by Algorithm 4.1 are local minimizers of f. It does not tell us, however, anything about the guarantee or rate of convergence of (x_k).*

We will present two results for local behaviour. First one describes the behaviour of Algorithm 4.1 in a neighborhood of a non-degenerate saddle point or a point of local maximum.

Lemma 4.6. *Assume $\overline{x} \in$ int Q to be a non-degenerate saddle point or a point of local maximum of f :*

$$f'(\overline{x}) = 0, \quad \lambda_n(f''(\overline{x})) < 0.$$

Then there exist constants $\epsilon, \delta > 0$ such that if an iterate x_k is close to $\overline{x}$ and is also a non-degenerate saddle point or a point of local maximum, i.e., $x \in M$ for $M := \{x \ : \ \|x - \overline{x}\| \leq \epsilon, f(x) \geq f(\overline{x})\}$ (e.g., $x_k = \overline{x}$), then the next point x_{k+1} for $M_k \geq L$ leaves the set M and never comes there again (since the process defined in the Algorithm 4.1 is monotone w.r.t. objective function according to Remark 4.5):

$$f(x_{k+1}) \leq f(\overline{x}) - \delta.$$

Proof. Take a unit vector d and $\overline{s} > 0$ such that

$$\langle f''(\overline{x})d, d\rangle =: \sigma < 0, \text{ and } \overline{x} \pm sd \in Q.$$

Then, by applying Lemma 4.5 and upper bound on M_k from Algorithm 4.1 for $s \leq |\overline{s}|$ and $M_k \geq L$, we get:

$$f(x_{k+1}) = f(T_M(x_k)) \leq \min_{y\in Q}[f(y) + \frac{L+M}{6}\|y - x\|^3] \leq f(\overline{x}+sd) + \frac{L}{2}\|\overline{x} + sd - x_k\|^3.$$

Now, use Lemma 4.1 and the assumption on $\overline{x}$ to get for $s \leq |\overline{s}|$ that

$$f(\overline{x}+sd)+\frac{L}{2}\|\overline{x}+sd-x_k\|^3 \leq f(\overline{x})-\sigma s^2+\frac{L}{6}|s|^3+\frac{L}{2}(\epsilon^2+2s\langle \overline{x}-x_k,d\rangle+|s|^2)^{3/2}$$
$$\leq f(\overline{x})-\sigma s^2+\frac{L}{6}|s|^3+\frac{L}{2}(\epsilon^2+|s|^2)^{3/2},$$

where the last inequality holds as we can choose the sign of s as we want.
Set now $\epsilon := \min\{\frac{\sigma}{2L}, \overline{s}\} \leq \overline{s}$ and $s := \epsilon$. Then it holds

$$f(x_{k+1}) \leq f(\overline{x})-\sigma s^2+\frac{L}{6}s^3+\frac{9L}{6}s^3 \leq f(\overline{x})-\sigma s^2+\frac{5L}{3}\frac{\sigma}{2L}s^2 = f(\overline{x})-\frac{1}{6}\sigma s^2.$$

Set then $\delta := \frac{1}{6}\sigma\epsilon^2$, which gives the result.

□

Second result uses the simplified version of Algorithm 4.1:

$$x_{k+1} = T_{M_k}(x_k), \quad k \geq 0, \tag{9}$$

where $M_k \in (0, 2L]$.

Theorem 4.4. *Let $f''(x_0) \succ 0$ and $\delta_0 \leq \frac{1}{4}$. Let the sequence (x_k) be generated by (9). Further, denote*

$$\delta_k = \frac{L\,\|f'(x_k)\|}{\lambda_n^2(f''(x_k))}.$$

Then:

1. *For all $k \geq 0$, the values δ_k converge quadratically to zero:*

$$\delta_{k+1} \leq \frac{3}{2}\Big(\frac{\delta_k}{1-\delta_k}\Big)^2 \leq \frac{8}{3}\delta_k^2.$$

2. *Smallest eigenvalues of all Hessians $f''(x_k)$ are bounded as follows:*

$$e^{-1}\lambda_n(f''(x_0)) \leq \lambda_n(f''(x_k)) \leq e^{3/4}\lambda_n(f''(x_0)).$$

3. *The sequence (x_k) converges quadratically to a point x^*, which is a non-degenerate local minimum of function f. Furthermore, for all $k \geq 1$ we have*

$$\|f'(x_k)\| \leq \lambda_n^2(f''(x_0))\frac{3e^{3/2}}{8L}\Big(\frac{2}{3}\Big)^{2^k}.$$

Proof. Take some $k \geq 0$ such that $f''(x_k) \succ 0$ and $\delta_k \leq \frac{1}{4}$ (we can always take $k = 0$ according to the assumption). Then, using Remark 4.1, we get for the simplified version of Algorithm 4.1, defined in (9), that

$$r_{M_k}(x_k) = \|T_{M_k}(x_k) - x_k\| = \left\|(f''(x_k) + r_{M_k}(x_k)\frac{M_k}{2}I)^{-1}f'(x_k)\right\| \leq \frac{\|f'(x_k)\|}{\lambda_n(f''(x_k))}.$$

Using the inequality (7), we also have that in the next step Hessian is positive definite by

$$\begin{aligned}\lambda_n(f''(x_{k+1})) &\geq \lambda_n(f''(x_k)) - r_{M_k}(x_k)L \\ &\geq \lambda_n(f''(x_k)) - \frac{\|f'(x_k)\|}{\lambda_n(f''(x_k))}L = (1-\delta_k)\lambda_n(f''(x_k)) > 0.\end{aligned}$$

Using the above results, the upper bound on $\|f'(x_{k+1})\|$ from Lemma 4.2 and the upper bound on M_k, we obtain the first statement of the theorem by:

$$\begin{aligned}\delta_{k+1} &= \frac{L\,\|f'(x_{k+1})\|}{\lambda_n^2(f''(x_{k+1}))} \\ &\leq \frac{3L^2 r_{M_k}^2(x_k)}{2(1-\delta_k)^2\lambda_n^2(f''(x_k))} \\ &= \frac{3}{2}\Big(\frac{\delta_k}{1-\delta_k}\Big)^2 \leq \frac{8}{3}\delta_k^2 \leq \frac{2}{3}\delta_k \leq \frac{1}{4}.\end{aligned}$$

Not that we get the first statement, as the derivation above can be repeated for any $k \geq 0$, as we have shown that if $f''(x_k) \succ 0, \delta_k \leq \frac{1}{4}$, then also $f''(x_{k+1}) \succ 0, \delta_{k+1} \leq \frac{1}{4}$. Now, the lower bound of the second statement of the theorem is proven by

$$\begin{aligned}\ln\frac{\lambda_n(f''(x_k))}{\lambda_n(f''(x_0))} &\geq \sum_{i=0}^{k}\ln(1-\delta_i) \geq -\sum_{i=0}^{k}\delta_i\sum_{j=0}^{\infty}\delta_i^j = -\sum_{i=0}^{k}\frac{\delta_i}{1-\delta_i} \\ &\geq -\frac{1}{1-\delta_0}\sum_{i=0}^{k}\delta_i \geq -\frac{\delta_0}{1-\delta_0}\sum_{i=0}^{\infty}\Big(\frac{2}{3}\Big)^i \geq -1\end{aligned}$$

where in the second inequality we used the Taylor series expansion of $\ln(1-\delta_i)$ as $0 \leq \delta_i < 1$ for all $i \geq 0$.
To prove the upper bound, use that, similarly to the inequality (7), $\lambda_n(f''(x_{k+1})) \leq \lambda_n(f''(x_k)) + r_{M_k}(x_k)L \leq (1+\delta_k)\lambda_n(f''(x_k))$. This gives us

$$\ln\frac{\lambda_n(f''(x_k))}{\lambda_n(f''(x_0))} \leq \sum_{i=0}^{k}\ln(1+\delta_i) \leq \sum_{i=0}^{\infty}\ln(1+\delta_i) \leq \sum_{i=0}^{\infty}\delta_i \leq \frac{3}{4}.$$

Thus, it is only left to prove the third statement of the theorem. We show first that (x_i) is a Cauchy sequence in $\mathbb{R}^n$ and thus convergent to some unique x^*. It follows directly by the triangle inequality and the following derivation:

$$r_{M_k}(x_k) \leq \frac{\|f'(x_k)\|}{\lambda_n(f''(x_k))} = \frac{\delta_k \lambda_n(f''(x_k))}{L} \leq \frac{e^{3/4}\lambda_n(f''(x_0))\delta_k}{L},$$

which is a combination of the results above. Because $f''(x_0) \succ 0$, second proven statement holds, and the eigenvalues of a $f''(x)$ are continuous functions of x due to [47], we have that $f''(x^*) \succ 0$.
The last inequality follows directly from

$$\delta_k \leq \frac{3}{2}\frac{\delta_{k-1}^2}{(1-\delta_0)^2} \leq \frac{8}{3}\delta_{k-1}^2 \leq \frac{3}{8}\Big(\frac{8}{3}\delta_0\Big)^{2^k} \leq \frac{3}{8}\Big(\frac{2}{3}\Big)^{2^k},$$

for all $k \geq 1$.

□

Remark 4.8. *The theorem above guarantees local quadratic convergence under the assumption of non-degeneracy. The paper [8] gives similar guarantees under a milder local error bound (EB) condition.*

4.6 Empirical considerations

Let us estimate the cost of finding $M_k \in [L_0, 2L]$ in Algorithm 4.1 such that

$$f(T_{M_k}(x_k)) \leq \overline{f}_{M_k}(x_k),$$

which holds for $M_k \geq L$. Since, as we know from Lemma 4.5, $f(x_k) - \overline{f}_{M_k}(x_k) \geq \frac{M_k}{12} r_{M_k}^3(x_k)$ and $\overline{f}_{M_k}(x_k) \leq \min_{y \in Q}[f(y) + \frac{L+M_k}{6}\|y - x_k\|^3]$, choosing M_k that is too big will lead to smaller steps in algorithm. Thus, we will do dynamic backtracking line search to estimate M_k:

$$\begin{aligned}
&M_k \leftarrow\in [L_0, 2L];\\
&\textbf{while } f(T_{M_k}(x_k)) > f(x_k) \textbf{ do } M_k \leftarrow 2M_k;\\
&x_{k+1} \leftarrow T_{M_k}(x_k); \quad M_{k+1} \leftarrow \max\{\frac{1}{2}M_k, L_0\}.
\end{aligned}$$

Using this line search, the total number of computations of the mapping T_M, if we assume N iterations of the process, is bounded from above by

$$2N + \log_2 \frac{2L}{L_0}.$$

To conclude, note that in Algorithm 4.1 it is possible to find elements of Levenberg-Marquardt approach (from the definition of the set $\mathcal{D}$), or a trust-region approach (from theorem and related discussion above), or a line search technique (as well from the discussion above).

5 Stochastic Cubic Regularization (SCR) for Fast Non-convex Optimization

5.1 Stochastic second-order optimization methods

Exactly computing function values, gradients and Hessians is infeasible for large-scale problems, such as training of deep neural networks (DNN). Therefore, we consider further the problem of **non-convex** optimization in the stochastic approximation framework ([22]):

$$\min_{x \in \mathbb{R}^n} f(x) := \mathbb{E}_{\xi \sim \mathcal{D}}[f(x; \xi)], \tag{10}$$

where we have access only to a stochastic function $f(x; \xi)$. Further, if not specifically noticed, the uniform sampling is assumed. A further improvement can be made by using more sophisticated sampling techniques (e.g., [36]).
Our target then is to find an ϵ-second-order stationary point.

5.2 Related work

The major bottleneck of CR especially in the stochastic case involves solving the cubic sub-problem, for which various techniques have been proposed ([15], [18], [42]).

5.3 Introduction

In this section we present a fully stochastic version of the cubic regularized method. It is theoretically and empirically in most of the cases (as it will be shown later) faster than SGD. It proposes a solution that works both for an online setting (data arrives sequentially) and offline settings (data is fixed).
Moreover, it uses only stochastic stochastic gradients $f'(x, \xi)$ and stochastic Hessian-vector products $f''(x, \xi)v$ for some vector v (computing which can be as cheap as computing a gradient, when our function is represented as an arithmetic circuit ([34])), and makes following assumptions, alognside with the assumption that f is bounded from below by some optimal value f^*:

Assumption 5.1. *1. The function f has l-Lipschitz gradients and L-Lipschitz Hessians, which are standard assumptions in works on escaping saddle points and finding local minima.*

2. The function $f(x,\xi)$ satisfies:

$$\forall x \in \mathbb{R}^n \quad \mathbb{E}\Big[\, \|f'(x,\xi) - f'(x)\|^2 \,\Big] \le \sigma_1^2 \quad \textit{and} \quad \|f'(x,\xi) - f'(x)\| \le M_1 \quad \textit{a.s.},$$
$$\forall x \in \mathbb{R}^n \quad \mathbb{E}\Big[\, \|f''(x,\xi) - f''(x)\|^2 \,\Big] \le \sigma_2^2 \quad \textit{and} \quad \|f''(x,\xi) - f''(x)\| \le M_2 \quad \textit{a.s.}$$

We denote further stochastic gradient by

$$g_k = \frac{1}{|S_1|} \sum_{\xi_i \in S_1} f'(x_k, \xi_i) \quad \forall k \ge 0$$

and stochastic Hessian by

$$B_k = \frac{1}{|S_2|} \sum_{\xi_i \in S_2} f''(x_k, \xi_i) \quad \forall k \ge 0$$

for every iteration k.
At every step we draw enough samples $|S_1|$ and $|S_2|$ so that the concentration conditions

$$\begin{aligned} \|g_k - f'(x_k)\| &\le c_1\epsilon, \\ \forall v \in \mathbb{R}^n : \quad \|(B_k - f''(x_k))v\| &\le c_2\sqrt{L\epsilon}\,\|v\| \end{aligned}$$

are satisfied for c_1, c_2 small enough. This conditions are indeed satisfied with the following assumptions:

Lemma 5.1. *For any fixed constants c_1, c_2, we can pick stochastic gradient and Hessian-vector product mini-batch sizes $n_1 = \widetilde{\mathcal{O}}\Big(\max\Big\{\frac{M_1}{c_1\epsilon}, \frac{c_1^2\sigma_1^2}{\epsilon^2}\Big\}\Big)$, respectively $n_2 = \widetilde{\mathcal{O}}\Big(\max\Big\{\frac{M_2}{c_2\sqrt{L\epsilon}}, \frac{\sigma_2^2}{c_2^2 L\epsilon}\Big\}\Big)$ so that with probability at least $1 - 2\delta'$,*

$$\begin{aligned} \|g_k - f'(x_k)\| &\le c_1\epsilon, \\ \forall v \in \mathbb{R}^n : \quad \|(B_k - f''(x_k))v\| &\le c_2\sqrt{L\epsilon}\,\|v\|\,. \end{aligned}$$

Let further $m_k, \widetilde{m}_k$ denote respectively the **stochastic cubic sub-model** and the **change in the sub-model value**

$$m_k(x) = f(x_k) + \langle g_k, x - x_k\rangle + \frac{1}{2}\langle B_k(x - x_k), x - x_k\rangle + \frac{L}{6}\|x - x_k\|^3\,,$$

and

$$\widetilde{m}_k(\Delta) = \langle g_k, \Delta \rangle + \frac{1}{2}\langle B_k \Delta, \Delta \rangle + \frac{L}{6}\|\Delta\|^3 \quad (= m_k(x_k + \Delta) - m_k(x_k)).$$

As in the non-stochastic case, we want to minimize the cubic sub-problem and the change in the sub-model value, as it is natural to do according to the following claim which states informally that if the cubic sub-model has large descent $m_k(x_{t+1}) - m_k(x_k)$ then the true function also has large descent $f(x_{t+1}) - f(x_k)$.

Remark 5.1. *In the Claim 5.1 below,* ***Case 1*** *is not considered, as in that case the statement of the claim is trivially true by choosing the constants such that* $-\frac{7}{10} \leq -(\frac{1-c_3}{96})$.

Claim 5.1. *Assume that we are in the setting of Lemma 5.1 with constants* c_1, c_2 *small enough. If the CubicSubsolver 5.2 satisfies* ***Case 2*** *of the Condition 5.1 and if* $m_k(x_k + \Delta) - m_k(x_k) \leq -(\frac{1-c_3}{96})\sqrt{\frac{\epsilon^3}{L}}$, *then* $f_k(x_k + \Delta) - f_k(x_k) \leq -(\frac{1-c_3-c_5}{96})\sqrt{\frac{\epsilon^3}{L}}$.

This statement is especially important for the stochastic setting, when we do not have access to the true function, but only the cubic sub-model. Thus, we want to find an approximation Δ of the solution to the the cubic sub-model $\Delta^* = \arg\min_\Delta \widetilde{m}_k(\Delta)$. This approximation should satisfy then the following:

Condition 5.1. *For any fixed, small constant* c *and* $K_1 = \frac{7}{10}$, *CubicSubsolver 5.2 terminates within* $\mathcal{T}(\epsilon)$ *gradient iterations (which depends on* c *in the* ***Case 2****) and returns* Δ *satisfying at least one of the following:*

1. $\max\{\widetilde{m}_k(\Delta), f(x_k + \Delta) - f(x_k)\} \leq -K_1\sqrt{\frac{\epsilon^3}{L}}$. *(****Case 1****)*
2. $\|\Delta\| \leq \|\Delta^*\| + c\sqrt{\frac{\epsilon}{L}}$ *and, if* $\|\Delta^*\| \geq \frac{1}{2}\sqrt{\frac{\epsilon}{L}}$, *then* $\widetilde{m}_k(\Delta) \leq \widetilde{m}_k(\Delta^*) + \frac{c}{12}L\|\Delta^*\|^3$. *(****Case 2****)*

Below we introduce the main algorithm with two subroutines for solving the cubic sub-problem: one that satisfies Condition 5.1 (more details in Lemmas 5.2 and 5.4) and another that ensures that the final point will be an ϵ-second-order stationary point up to rescaling.

Algorithm 5.1: SCR

Input : mini-batch sizes n_1, n_2 of two **independent** mini-batches (one can also use increasing sizes as in [37]), initialization x_0, number of iterations T_{out}, and final tolerance ϵ

Output: x_* if the early termination condition was reached, otherwise the final iterate $x_{T_{\text{out}}+1}$

for $k = 0, ..., T_{out}$ **do**

  Sample $S_1 \leftarrow \{\xi_i\}_{i=1}^{n_1}, S_2 \leftarrow \{\xi_i\}_{i=1}^{n_2}$;

  $g_k \leftarrow \frac{1}{|S_1|} \sum_{\xi_i \in S_1} f'(x_k, \xi_i)$;

  $B_k[\cdot] \leftarrow \frac{1}{|S_2|} \sum_{\xi_i \in S_2} f''(x_k, \xi_i)(\cdot)$ # $B_k[\cdot]$ stands for Hessian-vector products;

  $\Delta, \Delta_m \leftarrow$ CubicSubsolver $(g_k, B_k[\cdot], \epsilon)$ # $\Delta_m = \widetilde{m}_k(\Delta)$;

  $x_{t+1} \leftarrow x_k + \Delta$;

  # Check, if we are doing enough progress ;

  # The constant $-\frac{1}{100}$ is chosen for simplicity. In the analysis value $-(\frac{1-c_3}{96})$ used instead, where c_3 is such that we have the desired descent at every step using the Claim 5.1 ;

  if $\Delta_m \geq -\frac{1}{100}\sqrt{\frac{\epsilon^3}{L}}$ **then**

    $\Delta \leftarrow$ CubicFinalSubsolver $(g_k, B_k[\cdot], \epsilon)$;

    $x^* \leftarrow x_k + \Delta$;

    return x^*;

  end

end

return $x_{T_{\text{out}}+1}$;

Algorithm 5.2: CubicSubsolver

Input : g, $B[\cdot]$, tolerance ϵ
Output: Δ, Δ_m
# Check if we can satisfy **(Case 1)** of the condition 5.1 with the Cauchy point (more details in Lemma 5.2);
if $\|g\| \geq \frac{l^2}{L}$ **then**
  # This step has the complexity $\mathcal{O}(1)$ (more details in Lemma 5.2);
  $R_c \leftarrow -\frac{\langle Bg,g\rangle}{L\|g\|^2} + \sqrt{\left(\frac{\langle Bg,g\rangle}{L\|g\|^2}\right)^2 + \frac{2\|g\|}{L}}$ # The Cauchy radius;
  $\Delta \leftarrow -R_c \frac{g}{\|g\|}$ # the Cauchy point;
else
  # Otherwise, we satisfy **(Case 2)** of the condition 5.1 (more details in Lemma 5.4);
  # This step has the complexity $\widetilde{\mathcal{O}}(\frac{l}{\sqrt{L\epsilon}})$ (more details in Lemma 5.4);
  $\Delta \leftarrow 0, \sigma \leftarrow c'\epsilon, \eta \leftarrow \frac{1}{20l}$;
  # Justification for drawing $\widetilde{g}$ from the **perturbation ball** can be found in [26], Lemma 15 ;
  $\widetilde{g} \leftarrow g + \sigma\theta$ for $\theta \sim unif(\mathbb{S}^{n-1})$;
  for $k = 1, ..., \mathcal{T}(\epsilon)$ **do**
    $\Delta \leftarrow \Delta - \eta(\widetilde{g} + B\Delta + \frac{L}{2}\|\Delta\|\Delta)$;
  end
end
$\Delta_m = \widetilde{m}_k(\Delta)$;

Algorithm 5.3: CubicFinalSubsolver

Input : g, $B[\cdot]$, tolerance ϵ
Output: Δ
$\Delta \leftarrow 0, g_m \leftarrow g, \eta \leftarrow \frac{1}{20l}$;
while $\|g_m\| > \frac{\epsilon}{2}$ **do**
  $\Delta \leftarrow \Delta - \eta g_m$;
  $g_m \leftarrow g + B\Delta + \frac{L}{2}\|\Delta\|\Delta$;
end

5.4 Main results

Remark 5.2. *Note, further we assume that* $\epsilon \leq \frac{l^2}{L}$, *or* $\|g_k\| \geq \epsilon$ *is satisfied - otherwise, by the Lipschitz-gradient assumption, we get that* $\lambda_{min}(f''(x)) \geq -\sqrt{L\epsilon}$, *and*

$\|g_k\| \leq \epsilon$.

Condition 5.1 is theoretically important, as it gives us the rate of convergence to an ϵ-second-order stationary point:

Theorem 5.1. *There exists a constant c such that if f satisfies Assumption 5.1, CubicSubsolver satisfies Condition 5.1 with c and $n_1 \geq \max\{\frac{M_1}{c\epsilon}, \frac{\sigma_1^2}{c^2\epsilon^2}\} \log\left(\frac{n\sqrt{L}\Delta_f}{\epsilon^{1.5}\delta c}\right), n_2 \geq \max\{\frac{M_2}{c\sqrt{L\epsilon}}, \frac{\sigma_2^2}{c^2 L\epsilon}\} \log\left(\frac{n\sqrt{L}\Delta_f}{\epsilon^{1.5}\delta c}\right)$, then for all $\delta > 0$ and $\Delta_f \geq f(x_0) - f^*$, Algorithm 5.1 will output an ϵ-second-order stationary point of f with probability at least $1 - \delta$ within*

$$\widetilde{\mathcal{O}}\Big(\frac{\sqrt{L}\Delta_f}{\epsilon^{1.5}}\Big(\max\Big\{\frac{M_1}{c\epsilon}, \frac{\sigma_1^2}{c^2\epsilon^2}\Big\} + \max\Big\{\frac{M_2}{c\sqrt{L\epsilon}}, \frac{\sigma_2^2}{c^2 L\epsilon}\Big\} \cdot \mathcal{T}(\epsilon)\Big)\Big) = \widetilde{\mathcal{O}}(\epsilon^{-3.5})$$

total stochastic gradient and Hessian-vector product evaluations.

The next two Lemmas show that Condition 5.1 is indeed satisfied. The first one states that for the Cauchy step choice in Algorithm 5.2 we have the following result:

Lemma 5.2. *Assume the same setting as in Lemma 5.1 with c_1, c_2 sufficiently small. If $\|g_k\| \geq \frac{l^2}{L}$, the Cauchy step defined by $\Delta = -R_c \frac{g_k}{\|g_k\|}$ will satisfy*

$$\widetilde{m}_k(\Delta) \leq -K_1\sqrt{\frac{\epsilon^3}{L}}, \quad f(x_k + \Delta) - f(x_k) \leq -K_1\sqrt{\frac{\epsilon^3}{L}}$$

*for $K_1 = -\frac{7}{20}$. Thus, when $\|g_k\| \geq \frac{l^2}{L}$ Algorithm 5.2 will satisfy the **(Case 1)** in Condition 5.1.*
The complexity of computing the Cauchy step is $\mathcal{O}(1)$.

The second one gives guarantees for the gradient descent loop in Algorithm 5.2 (till the end of this section we will refer to it simply as gradient descent). Let us make first following notations for some step k:

$$\beta := \|B_k\|, \gamma = -\lambda_{min}(B_k), \text{ and } R = \frac{\beta}{L} + \sqrt{\Big(\frac{\beta}{L}\Big)^2 + \frac{2\|g_k\|}{L}}.$$

Let us make the following assumptions as well:

Assumption 5.2. *The step size η of the gradient descent in Algorithm 5.2 satisfies*

$$0 < \eta < \frac{1}{4(\beta + \frac{L}{2}R)}.$$

Assumption 5.3. *The initial iterate Δ of the gradient descent is set to*

$$\Delta = -r\frac{g_0}{\|g_0\|}, \text{ where } 0 \leq r \leq R_c.$$

Above we have stated assumptions and definitions that are used in Theorem 3.2 from [17]. This theorem is crucial then in the proof of Lemma 5.4 below for the complexity of the sufficient descent of the gradient descent.
First, let us restate Theorem 3.2, and the important parts of Corollary 2.5 and Lemma 4.6 from [17]:

Theorem 5.2. *Let Assumptions 5.2 and 5.3 be satisfied, $\widetilde{g} = g + \sigma q$ for $\sigma = \frac{L\epsilon'}{\beta + L\|\Delta^*\|}\frac{\bar{\sigma}}{12}$ such that $\bar{\sigma} \leq 1$, and $q \sim unif(\mathbb{S}^{n-1})$. Then*

$$\widetilde{m}(\Delta) \leq \widetilde{m}(\Delta^*) + (1 + \bar{\sigma}\epsilon')$$

for

$$t \geq \mathcal{T}(\epsilon') = \frac{1+\bar{\sigma}}{\eta}\min\left\{\frac{1}{\frac{L}{2}\|\Delta^*\| - \gamma}, \frac{10\|\Delta^*\|^2}{\epsilon'}\right\} \cdot \left(6\log\left(1 + \chi_{\{\gamma>0\}}\frac{3\sqrt{n}}{\bar{\sigma}\delta'}\right) + 14\log\left(\frac{(\beta + L\|\Delta^*\|)\|\Delta^*\|^2}{\epsilon'}\right)\right)$$

with probability $1 - \delta'$.

Corollary 5.1. *Let Assumptions 5.2 and 5.3 be satisfied. Then the iterates Δ of the gradient descent at every step satisfy $\langle \Delta, \widetilde{m}'(\Delta)\rangle \leq 0$, the norms $\|\Delta\|$ are non-decreasing and $\|\Delta\| \leq \left\|\widetilde{\Delta^*}\right\|$, where $\widetilde{\Delta^*}$ denotes the exact minimizer of the perturbed problem, i.e., when g_k is replaced with $\widetilde{g_k}$.*

Lemma 5.3. *In the setting of the perturbed problem as in the previous Corollary 5.1 we have that*

$$|\,\|\Delta^*\| - \left\|\widetilde{\Delta^*}\right\|\,| \leq \frac{4\sigma}{L}$$

They are used to show the complexity of the sufficient descent:

Lemma 5.4. *Assume the same setting as in Lemma 5.1 with c_1, c_2 sufficiently small, $\|g_k\| \leq \frac{l^2}{L}, \eta = \mathcal{O}(\frac{1}{l}), \widetilde{g}_k = g_k + \sigma q$ for $\sigma = O(\epsilon)$, and $q \sim unif(\mathbb{S}^{n-1})$. If $\|\Delta^*\| \geq \frac{1}{2}\sqrt{\frac{\epsilon}{L}}$ and $\frac{L}{2}\|\Delta^*\| - \gamma \leq \frac{\epsilon'}{10\|\Delta^*\|^2}$ then the gradient descent iterate $\Delta_{\mathcal{T}(\epsilon')}$ for $\mathcal{T}(\epsilon') = \widetilde{\mathcal{O}}(\frac{l}{\sqrt{L\epsilon}})$ will satisfy*

$$\widetilde{m}_k(\Delta_{\mathcal{T}(\epsilon')}) \leq \widetilde{m}_k(\Delta^*) + c_3\frac{L}{12}\|\Delta^*\|^3$$

with probability $1-\delta'$*, where* δ' *is as in Theorem 5.2 and will always satisfy*

$$\left\|\Delta_{\mathcal{T}(\epsilon')}\right\| \leq \|\Delta^*\| + c_4\sqrt{\frac{\epsilon}{L}}$$

for $\epsilon' = c_3\frac{L}{24}\|\Delta^*\|^3$*. Thus, when* $\|g_k\| \leq \frac{l^2}{L}$*, Algorithm 5.2 will satisfy the* ***(Case 2)*** *in Condition 5.1.*

Remark 5.3. *We will eventually choose* $\delta' = \mathcal{O}(\epsilon^{1.5})$*, which means, that the smaller we want our approximation error to be, the smaller should be the probability of failure of the guarantees at every step.*

Now, Lemmas 5.2 and 5.4 and Theorem 5.1 are used to get the following Corollary, which is the main theoretical result for Algorithm 5.1. Note that theoretically it gives an improvement over the $\widetilde{\mathcal{O}}(\epsilon^{-4})$ rate of SGD ([25]).

Corollary 5.2. *Under the same setting as in Theorem 5.1, if* $\epsilon \leq \min\{\frac{\sigma_1^2}{c_1 M_1}, \frac{\sigma_2^4}{c_2^2 M_2^2 L}\}$*, and if we use Algorithm 5.2 for the CubicSubsolver, then with the probability greater than* $1-\delta$ *and* $\Delta_f \geq f(x_0)-f^*$ *Algorithm 5.1 will return the* ϵ*-second-order stationary point of* $f(x)$ *within*

$$\widetilde{\mathcal{O}}\Big(\frac{\sqrt{L}\Delta_f}{\epsilon^{1.5}}\Big(\frac{\sigma_1^2}{\epsilon^2} + \frac{\sigma_2^2}{L\epsilon}\frac{l}{\sqrt{L\epsilon}}\Big)\Big) = \widetilde{\mathcal{O}}(\epsilon^{-3.5})$$

total stochastic gradient and Hessian-vector product evaluations.

Remark 5.4. *In the previous corollary we have four main terms:*

1. $\frac{\sqrt{L}\Delta_f}{\epsilon^{1.5}}$ *comes from the satisfaction of Condition 5.1 at each iteration of the loop in Algorithm 5.1,*
2. $\frac{\sigma_1^2}{\epsilon^2}$ *comes from the sample size of stochastic gradients being big enough for Condition 5.1 to be satisfied (more details in Lemma 5.1),*
3. $\frac{\sigma_2^2}{L\epsilon}$ *comes from the sample size of stochastic Hessian-vector products being big enough for Condition 5.1 to be satisfied (more details in Lemma 5.1),*
4. $\frac{l}{\sqrt{L\epsilon}}$ *comes from the number of iterations of the loop in the CubicSubsolver 5.2 corresponding to the* ***(Case 2)*** *(more details in Lemma 5.4).*

6 Alternative algorithms for CubicSubsolver

As gradient descent is considered as a most basic algorithm, we want to discuss some alternative sub-solvers. We will be interested in empirical algorithms and their practical relevance for us in this section. Note that in the experiments we will not use Algorithm 5.3.

6.1 Adaptive gradient descent (AdGD)

This method ([23]) removes the need of tuning an additional hyper-parameter - learning rate - in the gradient descent as a sub-solver. It does so by approximating the learning rate as an approximation of the inverse local Lipschitz constant. Because polynomials (and the cubic sub-problem is a polynomial) are locally Lipschitz continuous, it is of special interest for us.
Concretely, the adaptive stochastic gradient descent method in our case is described in the following algorithm:

Algorithm 6.1: Adaptive SGD

Input : $g, B[\cdot]$ - stochastic gradient (sampled in the outer step of Algorithm 5.3), Hessian-vector product (sampled at every product), Δ_0 - initialized by the Cauchy step, $T_{\text{stop}}, \alpha > 0$

Output: $\Delta_{T_{\text{stop}}}$

$\hat{g}_0 \leftarrow g, \Delta_1 \leftarrow \Delta_0 - \hat{g}_0, \lambda_0 > 0, \theta_0 = +\infty;$

for $k = 1, ..., T_{stop}$ **do**

$\hat{g}_k \leftarrow g + B\Delta + \frac{L}{2} \|\Delta\| \Delta;$

$L \leftarrow \frac{\|\hat{g}_k - \hat{g}_{k-1}\|}{\|\Delta_k - \Delta_{k-1}\|}$;

$\lambda_k \leftarrow \min\{\sqrt{1 + \theta_{k-1}}\lambda_{k-1}, \frac{\alpha}{L_k}\}$;

$\Delta_k \leftarrow \Delta_{k-1} - \lambda_k \hat{g}_k;$

$\theta_k = \frac{\lambda_k}{\lambda_{k-1}}$;

end

6.2 Solving a linear system

This is a direct modification of **Case 2** in Algorithm 5.2, where instead of

$$\Delta \leftarrow \Delta - \eta(\widetilde{g} + B\Delta + \frac{L}{2} \|\Delta\| \Delta),$$

we will solve the respective linear system by:

$$\Delta \leftarrow -(B + \frac{L}{2} \|\Delta\|)^{-1} \widetilde{g},$$

where the inverse matrix-vector products (we say so instead of the "inverse Hessian-vector products" because we would like to invert not the approximated Hessian but the sum of it and some multiple of identity) will be approximated with one of the methods discussed in the next section. This method allows as well to avoid the tuning of the learning rate.

7 Hessian approximation

In order to both efficiently compute inverse matrix-vector products, as introduced in the previous section, and Hessian-vector products required in the gradient descent based cubic sub-solver, we will introduce following empirical schemes and discuss their practical relevance.

7.1 AdaHessian

As in Adam ([44]), this method relies on the exponential averaging of the first and second moments. Concretely, in our setting, at every step $k \geq 1$, it computes the averaged stochastic gradient $\bar{g}_k$ and Hessian $\bar{B}_k$ based on the past stochastic gradients g_k and stochastic Hessian diagonal approximations $\widetilde{B}_k$ as follows:

$$\bar{g}_k = \frac{(1-\beta_1)\sum_{i=1}^{t} \beta_1^{t-i} g_i}{1-\beta_1^t},$$

$$\bar{B}_k = \frac{(1-\beta_2)\sum_{i=1}^{t} \beta_2^{t-i} \widetilde{B}_i \odot \widetilde{B}_i}{1-\beta_2^t}.$$

Here, $0 < \beta_1, \beta_2 < 1$ are the first and the second moment hyper-parameters, that have the same values as in Adam. The stochastic Hessian diagonal approximations $\widetilde{B}_k$ at every step are computed using Hutchinson's method:

$$\widetilde{B}_k = z \odot (B_k z),$$

where $z \sim$ Rademacher(0.5). This approximation is motivated by the fact that $\mathbb{E}(\widetilde{B}_k) = \text{diag}(B_k)$ ([45]). The resulting $\bar{g}_k, \bar{B}_k$ are then used instead of g_k, B_k in the aforementioned sub-problem solvers. Note that the approximation of the stochastic Hessian with a vector allows to easily invert it, thus it can be used directly both for the Hessian-vector and inverse matrix-vector products.

7.2 SdLBFGS

This approximation of the inverse matrix-vector products relies on the adaptation of the L-BFGS method to the stochastic non-convex setting (10) - and is thus called stochastic damped L-BFGS (SdLBFGS)([41]). SdLBFGS preserves the positive-definiteness of the approximation to the matrix of interest in the stochastic non-convex case. The strongly convex case is easier, as the matrix preserves positive-definiteness, because the curvature condition

$$s_{k-1}^T y_{k-1} > 0$$

is satisfied then. The non-convex deterministic case can satisfy the curvature condition using line search. This, however, is not feasible in the non-convex stochastic case (10). It is done using the modified two-loop recursion (Procedure 3.1 in [41]).

7.3 WoodFisher

This approximation to the inverse matrix-vector products is motivated by the natural gradient descent ([49]). Here, we assume the conditional probability distribution $p_x(y_{\text{out}}|y_{\text{in}}) := p(y_{\text{out}}|f(y_{\text{in}}, x))$ of the output $y_{\text{out}} \in \mathbb{R}^{\text{out}}$, given the input $y_{\text{in}} \in \mathbb{R}^{\text{in}}$ and the model f, parametrized by x. We assume as well that there are N i.i.d. training samples $\{(y_{\text{in}}^i, y_{\text{in}}^i)\}_{i=1}^N$. Natural gradient descent then does constrained minimization at each k of the first-order approximation of the function f around x_k:

$$x_{k+1} = \arg\min_{x \in \mathbb{R}^n} f(x_k) + \langle f'(x_k), \Delta_k \rangle + \frac{\gamma}{2} \|\Delta_k\|_A^2 ,$$

where $\Delta_k := x - x_k$, update of which is, in turn, similar to the damped Newton method (1):

$$x_{k+1} = x_k - h_k A^{-1} f'(x_k),$$

where A is a weight matrix, chosen so that the KL-divergence between the probability distributions $p_{x_k}(y_{\text{out}}|y_{\text{in}}), p_{x_k+\Delta_k}(y_{\text{out}}|y_{\text{in}})$ stays constant. Intuitively, it is done to ensure the constant "speed" during the optimization.
It can be further shown ([48]) that, using the second-order Taylor expansion of $\ln p_{x_k+\Delta_k}(y_{\text{out}}|y_{\text{in}})$ around x_k,

$$\text{KL}(p_{x_k}(y_{\text{out}}|y_{\text{in}})||p_{x_k+\Delta_k}(y_{\text{out}}|y_{\text{in}}) \approx \frac{1}{2}\Delta_k^T F(x_k)\Delta_k = \frac{1}{2}\|\Delta_k\|_{F(x_k)}^2 ,$$

where $v(x_k) := \ln p_{x_k}(y_{\text{out}}|y_{\text{in}})$, and $F(x_k) := \mathbb{E}_{p_{x_k}(y_{\text{out}}|y_{\text{in}})}[v'(x_k)v'(x_k)^T]$
$= \mathbb{E}_{p_{x_k}(y_{\text{out}}|y_{\text{in}})}[-v''(x_k)]$. So we set A equal to $F(x_k)$. This is the Fisher information matrix at x_k, and we will call further $\widetilde{F}(x_k) := \frac{1}{N}\sum_{i=0}^N v_i'(x_k)v_i'(x_k)^T$, where

$v_i'(x_k) = \ln p_{x_k}(y^i_{\text{out}}|y^i_{\text{in}})$, the empirical Fisher information matrix at x_k.
In the case of the ℓ_2 loss function in regression and conditional probability distribution assumed to be a Gaussian distribution with a unit variance the empirical Fisher information matrix at x_k is known to approximate the Hessian at x_k (see e.g. [50], where also other generalizations are discussed).
In the experiments we use the update based on the Woodbury matrix identity ([51]) as in [52], Section 3.2.

8 Experiments

Here we focus on empirical results of the original SCR algorithm together with the improvements discussed above, such as the approximation of the stochastic Hessian matrix using AdaHessian, sdLBFGS, WoodFisher as well as solving the sub-problem using AdGD, and solving a linear system.
We compare the performance of the mentioned algorithms with Adam and SGD (without momentum) on four problems: synthetic problem of the w-shaped function, deep autoencoder (AE) ([56]) and linear regression applied to the MNIST dataset, convolutional neural network (CNN) ([57]) and deep residual neural network (ResNet-18) ([55]) applied to CIFAR dataset. As we will see, not only theoretically but also empirically SCR outperforms SGD in most of the cases and the proposed improvements of SCR outperform the original version of SCR discussed before in most of the cases.
For all problems except for the synthetic problem of the w-shaped function we will

- initialize weights using the uniform Xavier method ([59]);
- run first SGD for 10 epochs (full passes of the training data), where the learning rate (LR) is chosen during the cross-validation (CV) on 3 epochs (for more details regarding CV, see Appendix A) - which we will call the warm-start;
- initiate the respective algorithm with the parameters learned during the warm-start and run them for 90 epochs - which will result in a total of 100 epochs;
- apply the smoothing with the uniform windowing function of length 10 to make the results with high variance of the training loss more readable.

The use of warm-start is motivated by the paper [39], where two regimes of training are proposed.
For problems except for the synthetic problem of the w-shaped function the CV goes through the following set of parameters: LR from $A_{\text{LR}} := \{c \cdot 10^{-j} | c \in \{1, 3, 5, 6\}, j \in$

$\{1, 2, 3, 4, 5\}\}$, parameter M of $\xi_{2,x,M}$ - from $A_M := \{1, 10, 100, 500, 1000\}$ and the number of inner iterations $\mathcal{T}(\epsilon)$ - from $A_{\mathcal{T}} := \{1, 4, 10\}$. Moreover, we set the batch size for the stochastic (inverse) Hessian-vector product to 10. Considering that this may not approximate the (inverse) Hessian-vector product well enough, we will run additional experiment to check the approximation quality as described below. In all the experiments except for the CNN experiment we will see, however, that the value 10 gives a sufficient approximation.
Further we will denote by number of *computations* the total number of both stochastic gradient as well as stochastic (inverse) Hessian-vector product evaluations and by *oracle calls* the total number of samples used for each computation. For every experiment except for the synthetic problem of the w-shaped function we will provide 3 plots:

- for SGD, Adam, and the 2 best modifications of the SCR algorithm based on the results of the CV - the loss on the full training dataset (training loss) with respect to the oracle calls, which the theory on the convergence rate introduced in Section 5 supports;
- for the behaviour of the norm of the stochastic gradient of the best modification of the SCR based on the results of the experiments with 100 epochs;
- for the comparison of the training loss with respect to the number of computations of the best (chosen based on the results of the experiments with 100 epochs) modification of the SCR using the batch size for the Hessian-vector product of 10, respectively 100.

Experiments are done in PyTorch [40], and can be still speeded up using efficient Hessian-vector product computation using the Pearlmutter method [34].

8.1 Synthetic w-function

First, we would like to see, how SCR performs against SGD on the simple task with a saddle point. For this, as in [1], we use the function $w(x)$ with small negative curvature at the origin with 1-Lipschitz second derivative. Concretely, we want to minimize

$$\min_{x \in \mathbb{R}^2} [w(x_1) + 10x_2^2],$$

where w at $x \in \mathbb{R}$ is defined using a slope parameter ϵ and a length parameter L:

$$\begin{cases} \sqrt{\epsilon}(x+(L+1)\sqrt{\epsilon})^2 - \frac{1}{3}(x+(L+1)\sqrt{\epsilon})^3 - \frac{1}{3}(3L+1)\epsilon^{1.5}, & \text{if } x \leq -L\sqrt{\epsilon}; \\ \epsilon x + \frac{\epsilon^{1.5}}{3}, & \text{if } -L\sqrt{\epsilon} < x \leq -\sqrt{\epsilon}; \\ -\sqrt{\epsilon}x^2 - \frac{x^3}{3}, & \text{if } -\sqrt{\epsilon} < x \leq 0; \\ -\sqrt{\epsilon}x^2 + \frac{x^3}{3}, & \text{if } 0 < x \leq \sqrt{\epsilon}; \\ -\epsilon x + \frac{\epsilon^{1.5}}{3}, & \text{if } \sqrt{\epsilon} < x \leq L\sqrt{\epsilon}; \\ \sqrt{\epsilon}(x-(L+1)\sqrt{\epsilon})^2 + \frac{1}{3}(x-(L+1)\sqrt{\epsilon})^3 - \frac{1}{3}(3L+1)\epsilon^{1.5}, & \text{if } L\sqrt{\epsilon} \leq x. \end{cases}$$

In the experiments we set $\epsilon = 0.01, L = 5$, choose the best (with respect to the loss, where by loss we mean in this experiment the function evaluation at a given point) LR from A_{LR} and the best batch size for the Hessian-vector product from the set $\{10, 30, 100, 300\}$.

Below we provide the plots of the cross section of the w-function, the loss, and the behaviour of the norm of the stochastic gradient after the first step, at which the norm is 0. Only in this experiment it was required to prohibit big increases of loss by comparing the function evaluations at each two consecutive steps. Here it does not play a big role, as the function evaluations are cheap.

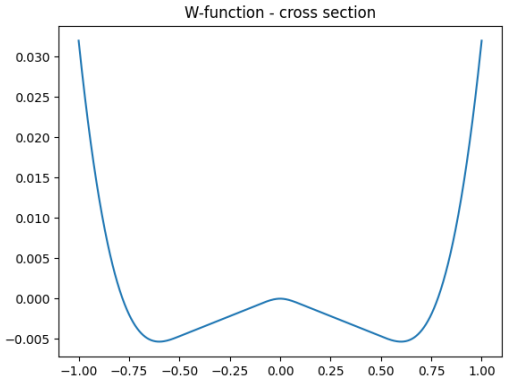


Figure 2: Cross section of the function we minimize

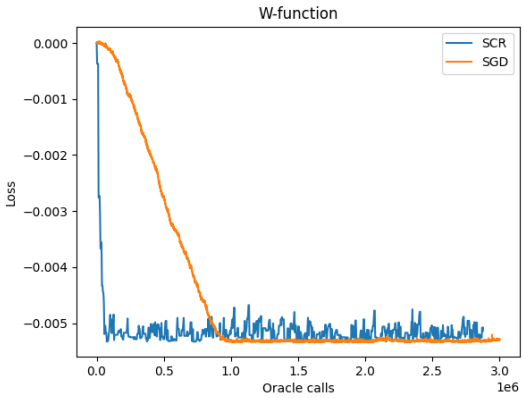


Figure 3: Performance of SGD vs. SCR

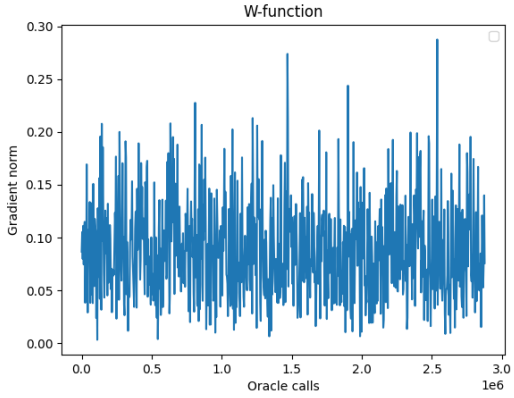


Figure 4: Behaviour of the stochastic gradient norm using SCR

As we can see, in the given setting of a synthetic saddle point problem SCR outperforms SGD and thus motivates us to run experiments on a series of non-synthetic problems.

8.2 Linear regression

In this experiment we use MNIST dataset, where each datapoint is in $[0,1]^{28\times 28}$ (28×28 grayscale images of digits), the mapping that we want to learn is $[0,1]^{28\times 28} \mapsto \mathbb{R}^{10}$ (we map from the space of the pictures to the space of the one-hot labels of the digits), and we use the ℓ_2 loss between the mapped values and the correct one-hot

labels. The meaning of the plots below is described in the beginning of the current section.
The difference in the length of the lines is due to the different number of oracle calls required to complete the experiments with 100 epochs. The longest corresponds to the modification of SCR with the higher number of inner steps $\mathcal{T}(\epsilon)$. The same holds for the rest of the experiments.

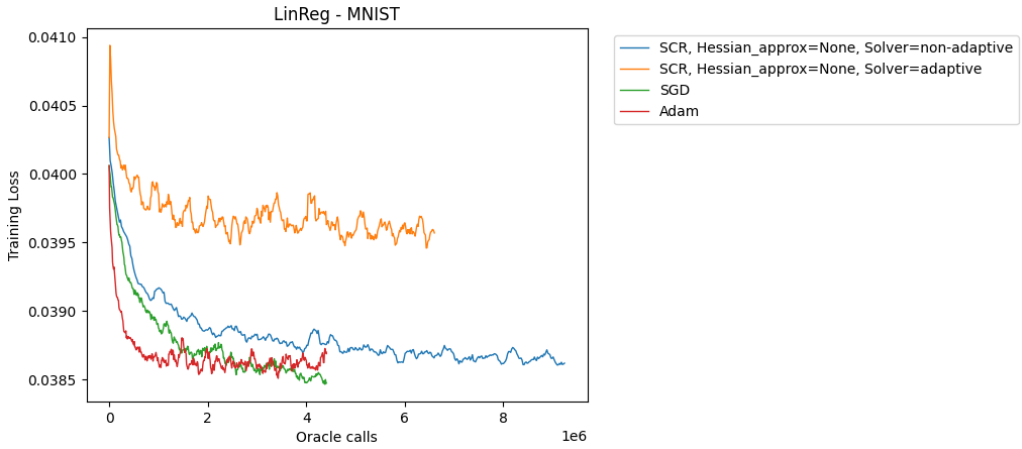


Figure 5: Performance of SGD vs. SCR vs. Adam

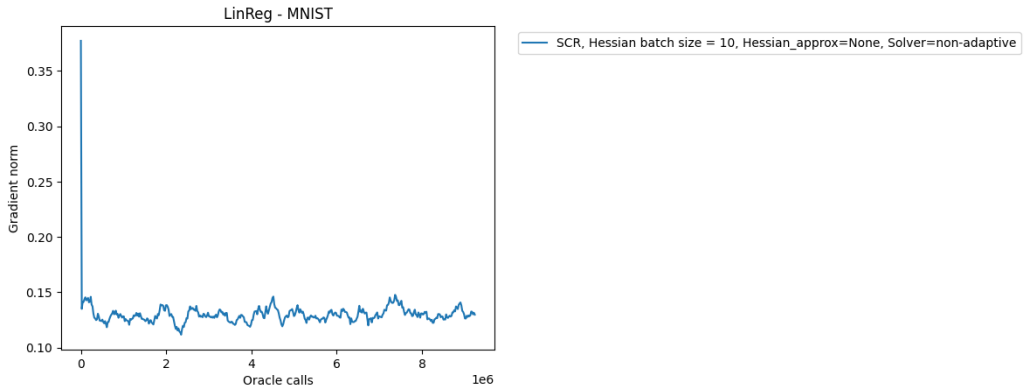


Figure 6: Behaviour of the stochastic gradient norm using SCR

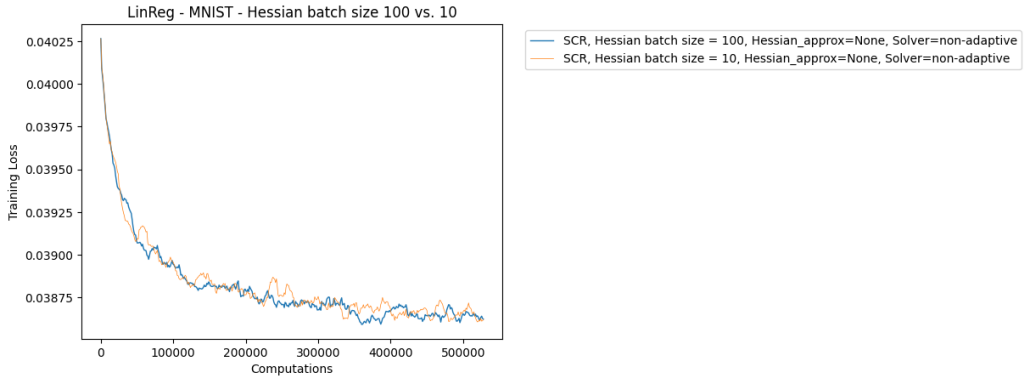


Figure 7: Influence of the increasing Hessian batch size

We would expect for a Newton-based method to have a faster convergence than that of the gradient descent for the convex loss function of this setting. However, in the stochastic setting after the warm-start it does not seem to hold. As we can see in Appendix A, from the CV on 3 epochs, before the warm-start, SCR outperforms SGD. This might indicate that another CV is required after the warm-start. Here we see additionally that the tenfold increase of the batch size for the stochastic Hessian-vector product does not result in the significant improvement of the speed of convergence.

8.3 Autoencoder (AE)

In this experiment we use the same dataset and the loss function as in the Linear regression experiment above. Here, however, we want to find another mapping - from $[0,1]^{28\times 28}$ to $[0,1]^{28\times 28}$ - which for the sake of simplicity is characterized as encoder of architecture $(28\times 28)\to 512\to 256\to 128\to 32$ with the symmetric decoder. The goal of this mapping, is to generate images of digits (elements of $[0,1]^{28\times 28}$) similar to those in the MNIST dataset.

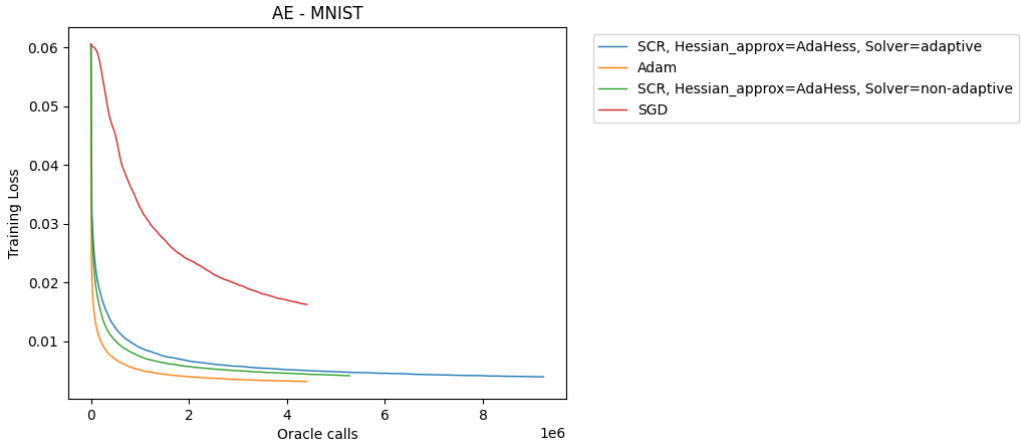


Figure 8: Performance of SGD vs. SCR vs. Adam

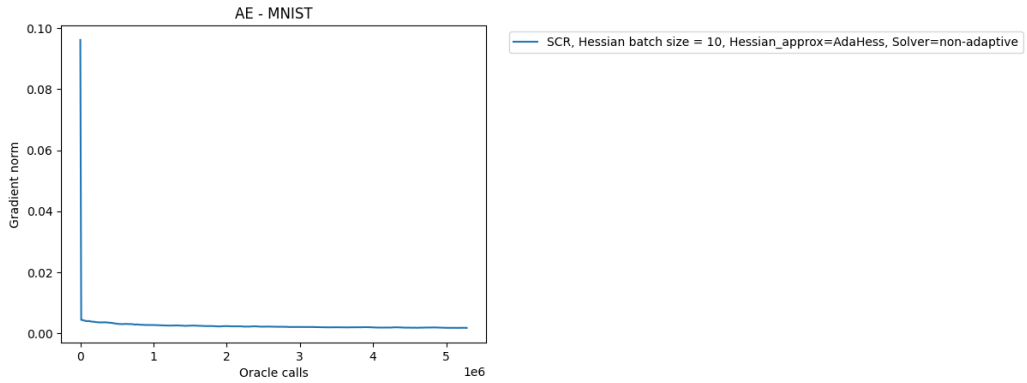


Figure 9: Behaviour of the stochastic gradient norm using SCR

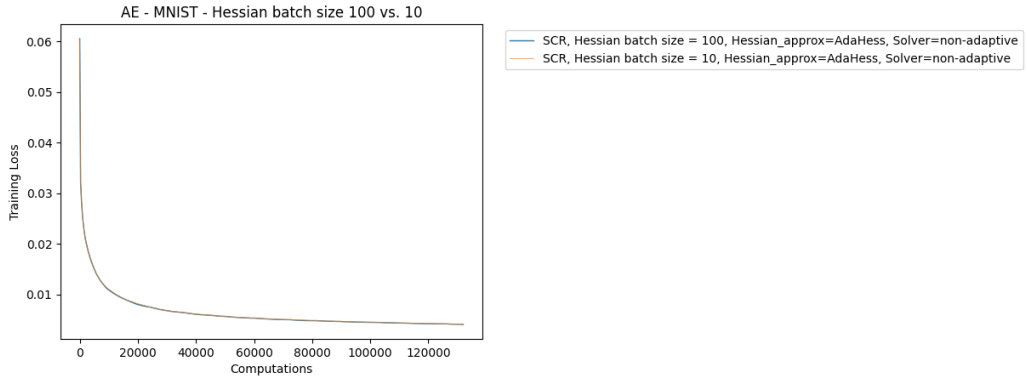


Figure 10: Influence of the increasing Hessian batch size

Here, we observe that, similarly to the results of [1], the SCR-based optimizers outperform SGD, which is supported by the theory introduced in the Section 5. Moreover, the convergence is close to the one of Adam, which is considered empirically to be one of the best optimizers for the stochastic optimization in terms of the speed of convergence. As in the case of the Linear regression experiment, increasing the batch size for the stochastic Hessian-vector product does not result in the significant improvement of the speed of convergence.

8.4 Convolutional neural network (CNN)

For the next two experiments we will use CIFAR dataset, where each datapoint is in $[0,1]^{32\times 32}$ (32×32 colour images in 10 classes), the mapping is from $[0,1]^{32\times 32}$ to $\mathbb{R}^{10}$,

and the loss function is the cross-entropy function between the the mapped vector $d \in \mathbb{R}^{10}$, which indicates, how probable it is that the given datapoint belongs to each of the classes $\{1, ..., 10\}$ and the correct class $C \in \{1, ..., 10\}$:

$$\text{loss}(d, C) = -\ln\left(\frac{\exp(d_C)}{\sum_{i\in\{1,...,10\}} \exp(d_i)}\right).$$

The architecture that characterizes the mapping is taken from Source.

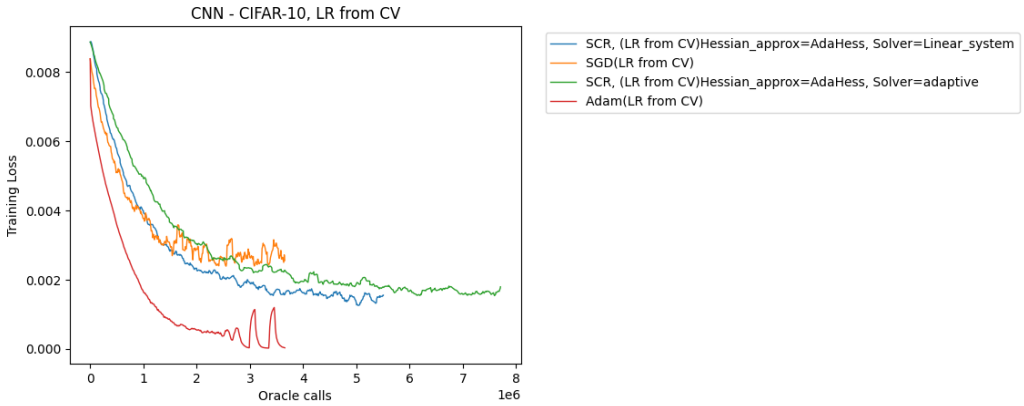


Figure 11: Performance of SGD vs. SCR vs. Adam with the LR taken from the CV on 3 epochs

In the figure above we can see that SGD has high variance of the training loss (even after applying the smoothing), thus we would like to see, if decreasing the LR would result in the less stochastic performance, which is clearly the case on the figure below. We see moreover that on the figure below decreasing LR results in the faster convergence and less stochastic performance of the SCR-based optimizers. This again indicates that the second CV after the warm-start can further improve the results.

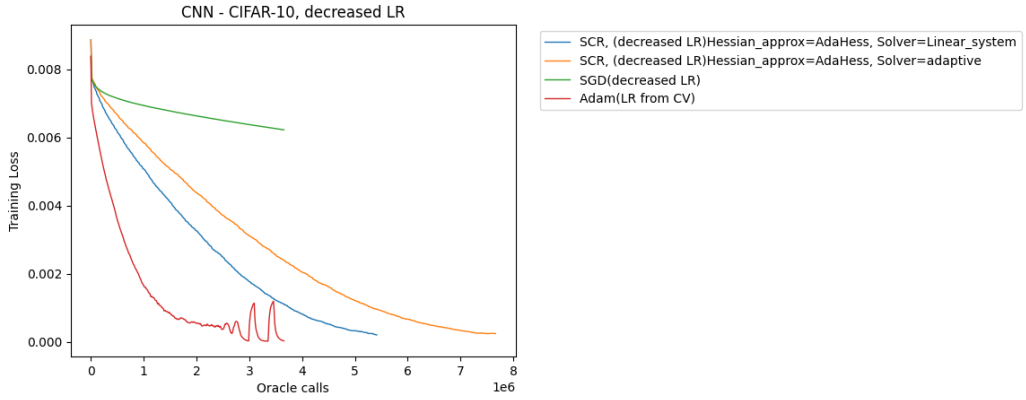


Figure 12: Performance of SGD vs. SCR vs. Adam with the decreased LR

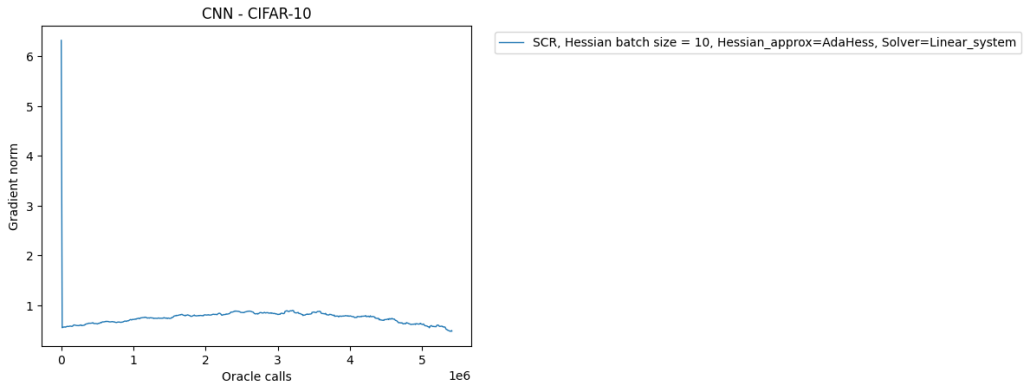


Figure 13: Behaviour of the stochastic gradient norm using SCR

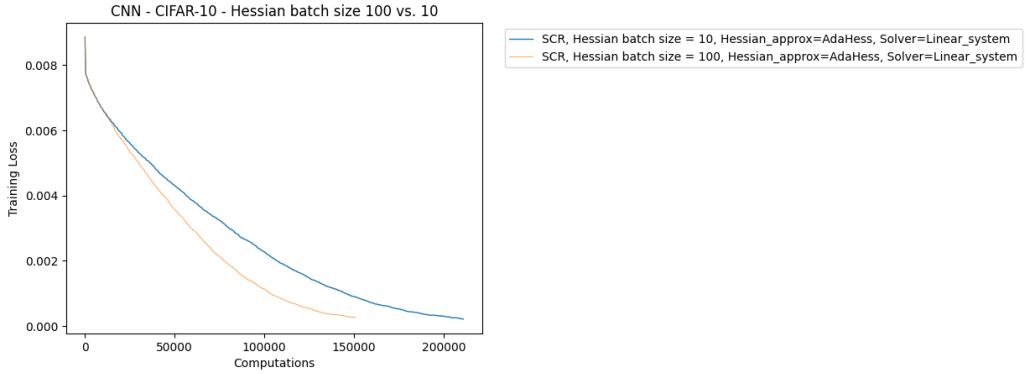


Figure 14: Influence of the increasing Hessian batch size

We can see that SCR-based optimizers outperform SGD, which is supported by the theory introduced in the Section 5, their convergence is also close to the one of Adam. Moreover, in this case increasing the batch size for the stochastic inverse Hessian-vector product clearly results in the faster convergence.

8.5 ResNet-18

In this experiment we use the same dataset and loss function as in the AE experiment above, but the architecture that characterizes the mapping has more parameters and is more complex. It is taken from Source.

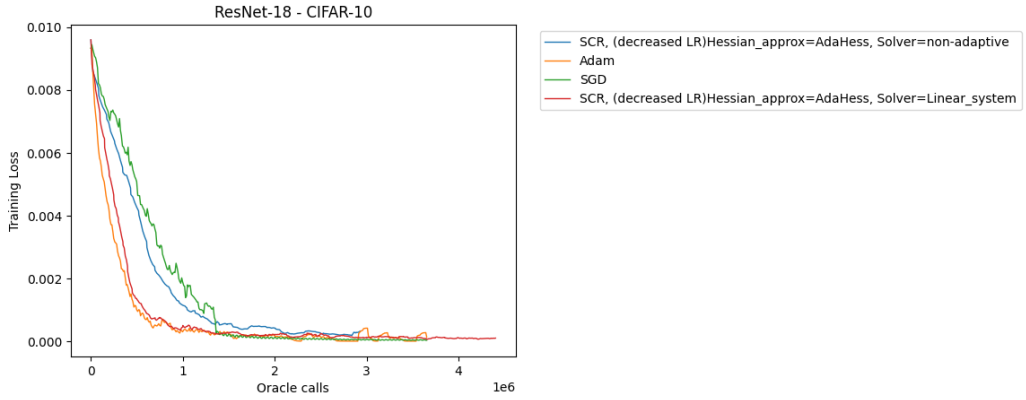


Figure 15: Performance of SGD vs. SCR

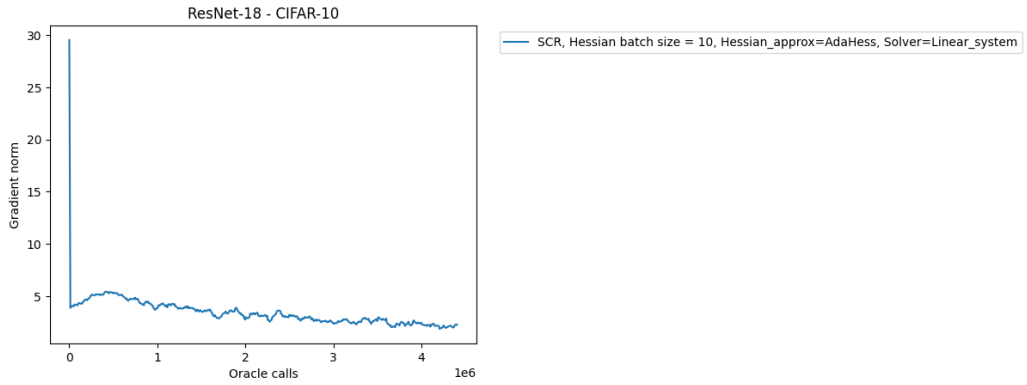


Figure 16: Behaviour of the stochastic gradient norm using SCR

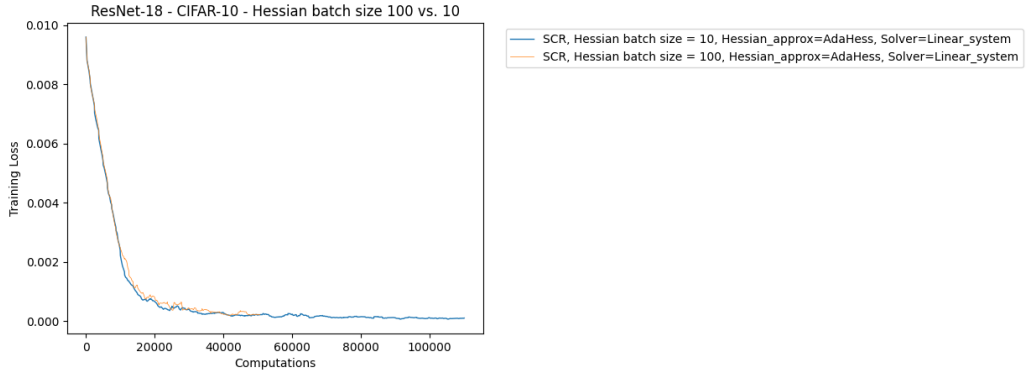


Figure 17: Influence of the increasing Hessian batch size

In this last experiment SCR-based optimizers again outperform SGD and their convergence is as well close to the one of Adam. Here, however, increasing the batch size of the inverse Hessian-vector product does not result in the significant improvement of the speed of convergence.

9 Conclusion

In this thesis we have given an overview of both the mathematical theory of the deterministic CR method as well as the main results for its stochastic modification - SCR. We have furthermore proposed novel improvements of both the cubic subproblem solver and the approximation of the (inverse) Hessian-vector products. In the experiments we could see that based on the CV on 3 epochs (see Appendix A)

the proposed modifications had faster convergence than the original SCR, except for the Linear regression experiment. Moreover, some of them let us remove one of the hyper-parameters - LR - to tune, which saves the time for the CV. We have looked as well empirically at the approximation quality of the stochastic (inverse) Hessian-vector products and seen that in the chosen experiments its value can be set to a value as low as 10 without significantly worsening the speed of convergence.
The proposed improvements had convergence in most of the cases faster than the one of SGD, which is supported by the described theory and similar to the one of Adam, which gives an indication that further additional improvements of SCR can make it a competitive second-order algorithm for stochastic optimization not only theoretically but also in practice.

References

[1] Nilesh Tripuraneni, Mitchell Stern, Chi Jin, Jeffrey Regier, Michael I. Jordan. Stochastic Cubic Regularization for Fast Non-convex Optimization. NeurIPS, 2018.

[2] Yurii Nesterov, Boris T. Polyak. Cubic regularization of Newton method and its global performance. Math. Program, 2006.

[3] Naman Agarwal, Brian Bullins, Elad Hazan. Second-Order Stochastic Optimization for Machine Learning in Linear Time. ArXiv, 2016.

[4] Ian J. Goodfellow, Oriol Vinyals, Andrew M. Saxe. Qualitatively characterizing neural network optimization problems. ICLR, 2015.

[5] Zeyuan Allen-Zhu. Natasha 2: Faster Non-Convex Optimization Than SGD. NeurIPS, 2018.

[6] Yi Zhou, Zhe Wang, Yingbin Liang. Convergence of Cubic Regularization for Non-convex Optimization under KŁ Property. NeurIPS, 2018.

[7] Yi Zhou, Yingbin Liang. Critical Points of Linear Neural Networks: Analytical Forms and Landscape Properties. ICLR, 2018.

[8] Man-Chung Yue, Zirui Zhou, Anthony Man-Cho So. On the Quadratic Convergence of the Cubic Regularization Method under a Local Error Bound Condition. SIAM J. Optim., 2019.

[9] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. SIAM J. Appl. Math., 1963.

[10] Andrew R. Conn, Nicholas I. M. Gould, and Philippe L. Toint. Trust Region Methods. SIAM, 2000.

[11] James M. Ortega, Werner Rheinboldt. Iterative Solution of Nonlinear Equations in Several Variables. SIAM, 1970.

[12] Yurii Nesterov. Introductory Lectures on Convex Optimization. Springer, 2014.

[13] Philipp Moritz, Robert Nishihara, Michael I. Jordan. A Linearly-Convergent Stochastic L-BFGS Algorithm. JMLR, 2016.

[14] Richard H. Byrd, S.L. Hansen, Jorge Nocedal, Yoram Singer. A Stochastic Quasi-Newton Method for Large-Scale Optimization. SIAM J. Optim., 2016.

[15] Naman Agarwal, Zeyuan Allen-Zhu, Brian Bullins, Elad Hazan, Tengyu Ma. Finding Approximate Local Minima Faster than Gradient Descent. ACM, 2017.

[16] Tommaso Bianconcini, Giampaolo Liuzzi, Benedetta Morini, Marco Sciandrone. On the use of iterative methods in cubic regularization for unconstrained optimization. Springer, 2013.

[17] Yair Carmon, John C. Duchi. Gradient Descent Efficiently Finds the Cubic-Regularized Non-Convex Newton Step. arXiv, 2016.

[18] Coralia Cartis, Nicholas I. M. Gould Philippe L. Toint. Adaptive cubic regularisation methods for unconstrained optimization. Springer, 2011.

[19] Liu Liu, Xuanqing Liu, Cho-Jui Hsieh, Dacheng Tao. Stochastic Second-order Methods for Non-convex Optimization with Inexact Hessian and Gradient. arXiv, 2018.

[20] Oriol Vinyals, Daniel Povey. Krylov Subspace Descent for Deep Learning. JMLR, 2012.

[21] Martin Jaggi. Revisiting Frank-Wolfe: Projection-Free Sparse Convex Optimization. 2013, JMLR.

[22] Herbert Robbins, Sutton Monro. A stochastic approximation method. The Annals of Mathematical Statistics, 1951.

[23] Yura Malitsky, Konstantin Mishchenko. Adaptive gradient descent without descent. arXiv, 2019.

[24] Robert M. Gower, Filip Hanzely, Peter Richtárik, Sebastian Stich. Accelerated Stochastic Matrix Inversion: General Theory and Speeding up BFGS Rules for Faster Second-Order Optimization. NeurIPS, 2018.

[25] Rong Ge, Furong Huang, Chi Jin, Yang Yuan. Escaping from saddle points — online stochastic gradient for tensor decomposition. COLT, 2015.

[26] Chi Jin, Rong Ge, Praneeth Netrapalli, Sham M. Kakade, Michael I. Jordan. How to Escape Saddle Points Efficiently. ICML, 2017.

[27] Simon S. Du, Chi Jin, Jason D. Lee, Michael I. Jordan, Barnabas Poczos, Aarti Singh. Gradient Descent Can Take Exponential Time to Escape Saddle Points. NeurIPS, 2017.

[28] Yi Xu, Rong Jin, Tianbao Yang. First-order Stochastic Algorithms for Escaping From Saddle Points in Almost Linear Time. NeurIPS, 2018.

[29] Zeyuan Allen-Zhu. How To Make the Gradients Small Stochastically: Even Faster Convex and Nonconvex SGD. NeurIPS, 2018.

[30] Zeyuan Allen-Zhu, Yuanzhi Li. Even Faster SVD Decomposition Yet Without Agonizing Pain. NIPS, 2016.

[31] Dan Garber, Elad Hazan, Chi Jin, Sham M. Kakade, Cameron Musco, Praneeth Netrapalli, Aaron Sidford. Robust shift-and-invert preconditioning: Faster and more sample efficient algorithms for eigenvector computation. ICML, 2016.

[32] Robert M. Gower, Filip Hanzely, Peter Richtárik, Sebastian Stich. Accelerated Stochastic Matrix Inversion: General Theory and Speeding up BFGS Rules for Faster Second-Order Optimization. NeurIPS, 2018.

[33] Boris T. Polyak. Convexity of quadratic transformations and its use in control and optimization. J. Optim. Theory and Appl., 1998.

[34] Barak A. Pearlmutter. Fast exact multiplication by the Hessian. Neural Computation, 1994.

[35] Lei-Hong Zhang, Chungen Shen, Ren-Cang Li. On the generalized Lanczos trust-region method. SIAM J. Optim., 2017.

[36] Deanna Needell, Nathan Srebro, Rachel Ward. Stochastic Gradient Descent, Weighted Sampling, and the Randomized Kaczmarz algorithm. NeurIPS, 2014.

[37] Michael P. Friedlander, Mark Schmidt. Hybrid Deterministic-Stochastic Methods for Data Fitting. SIAM J. Sci. Comput., 2012.

[38] Rong Ge, Chi Jin, Yi Zheng. No Spurious Local Minima in Nonconvex Low Rank Problems: A Unified Geometric Analysis. ICML, 2017.

[39] Guillaume Leclerc, Aleksander Madry. The Two Regimes of Deep Network Training. Preprint, 2020.

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Librar. NeurIPS, 2015.

[41] Xiao Wang, Shiqian Ma, Donald Goldfarb, Wei Liu. Stochastic Quasi-Newton Methods for Nonconvex Stochastic Optimization. SIAM J. Optim., 2017.

[42] Seonho Park, Seung Hyun Jung, Panos M. Pardalos. Combining Stochastic Adaptive Cubic Regularization with Negative Curvature for Nonconvex Optimization. Journal of Optimization Theory and Applications, 2020.

[43] Joel A. Tropp. An Introduction to Matrix Concentration Inequalities. Foundations and Trends® in Machine Learning, 2015.

[44] Diederik P. Kingma, Jimmy Ba. Adam: A method for stochastic optimization. ICLR, 2015.

[45] Constantine Bekas, Effrosyni Kokiopoulou, Yousef Saad. An estimator for the diagonal of a matrix. Applied numerical mathematics, 2007.

[46] Jérôme Bolte, Shoham Sabach, Marc Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Math. Program., 2014.

[47] Mishael Zedek. Continuity and Location of Zeros of Linear Combinations of Polynomials. Proceedings of the American Mathematical Society, 1965.

[48] Nathan Ratliff. Information geometry and natural gradients. 2013. Available: Source.

[49] Shun'ichi Amari. Neural learning in structured parameter spaces - natural Riemannian gradient. NeurIPS, 1997.

[50] Frederik Kunstner, Lukas Balles, Philipp Hennig. Limitations of the Empirical Fisher Approximation for Natural Gradient Descent. NeurIPS, 2019.

[51] Max A. Woodbury. Inverting modified matrices. Princeton, NJ : Department of Statistics, Princeton University, 1950.

[52] Sidak Pal Singh, Dan Alistarh. WoodFisher: Efficient Second-Order Approximation for Neural Network Compression. arXiv, 2020.

[53] Yann LeCun, Corina Cortes, Christopher J.C. Burges. MNIST handwritten digit database. ATT Labs [Online]. Available: Source.

[54] Alex Krizhevsky, Vinod Nair, Geoffrey Hinton. CIFAR-10 (Canadian Institute for Advanced Research). Available: Source.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. CVPR, 2016.

[56] Pierre Baldi. Autoencoders, unsupervised learning and deep architectures. JMLR, 2012.

[57] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998.

[58] R. Tyrrell Rockafellar, Roger J-B Wets. Variational analysis. Springer, 1998.

[59] Xavier Glorot, Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. AISTATS, 2010.

A CV on 3 epochs - pairwise comparison

On the plots below we can see the comparison of the speed of convergence of the training loss on 3 epochs of SGD with each of the SCR-based method, based on the improvements proposed in Sections 6 and 7. This is done for the problems of the Linear regression, AE, CNN, and ResNet, as these results are then used to select two SCR-based methods with the fastest convergence for the experiments with 100 epochs as described in the Section 8.
According to the discussion in the end of the Section 7.3, we will run the experiments with the stochastic inverse Hessian-vector products approximated using the WoodFisher method only for the Linear regression and AE experiments as only for them we have used the ℓ_2 loss function. Additionally, if all the experiments used a particular modification of the SCR-based method have diverged on a particular dataset, we will not display them.

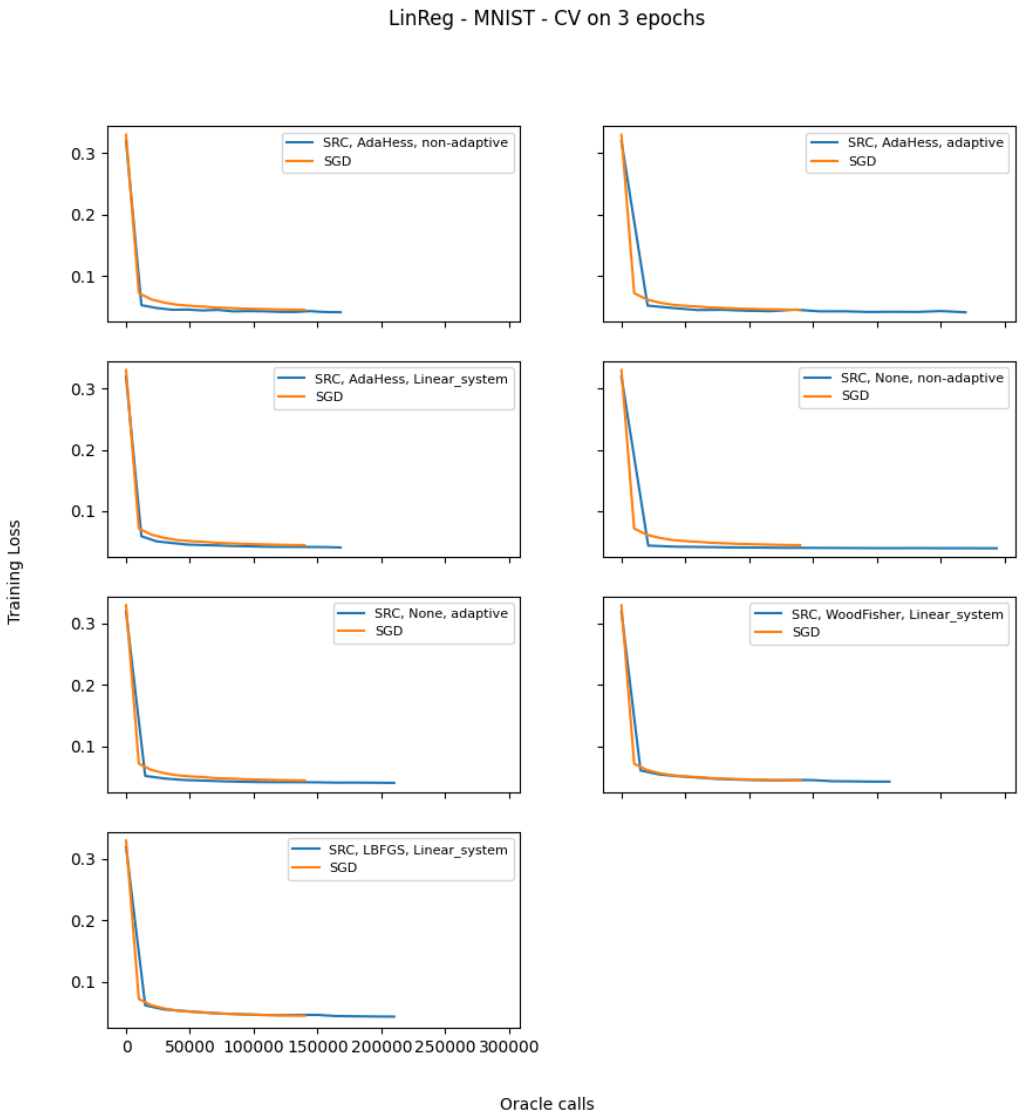


Figure 18: CV on 3 epochs - Linear regression

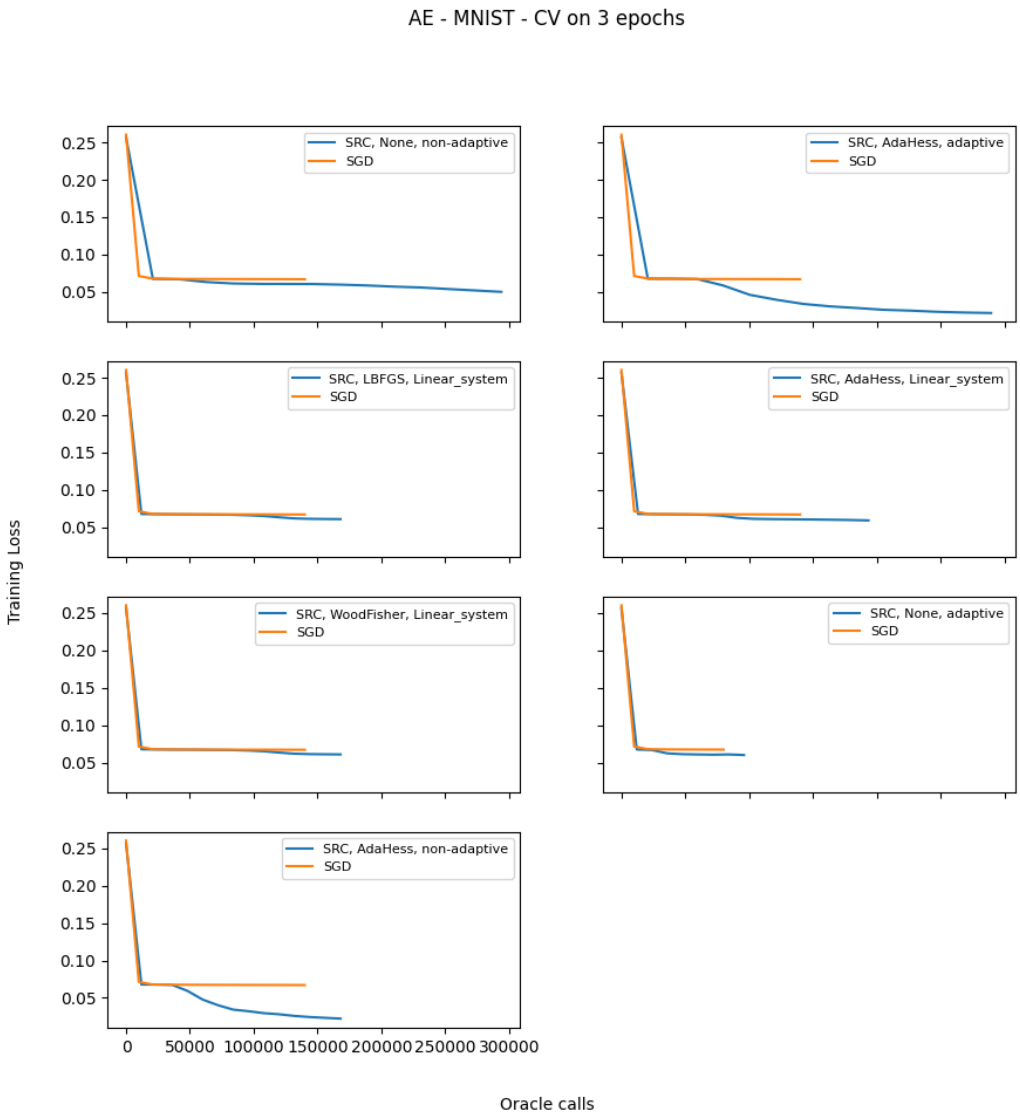


Figure 19: CV on 3 epochs - AE

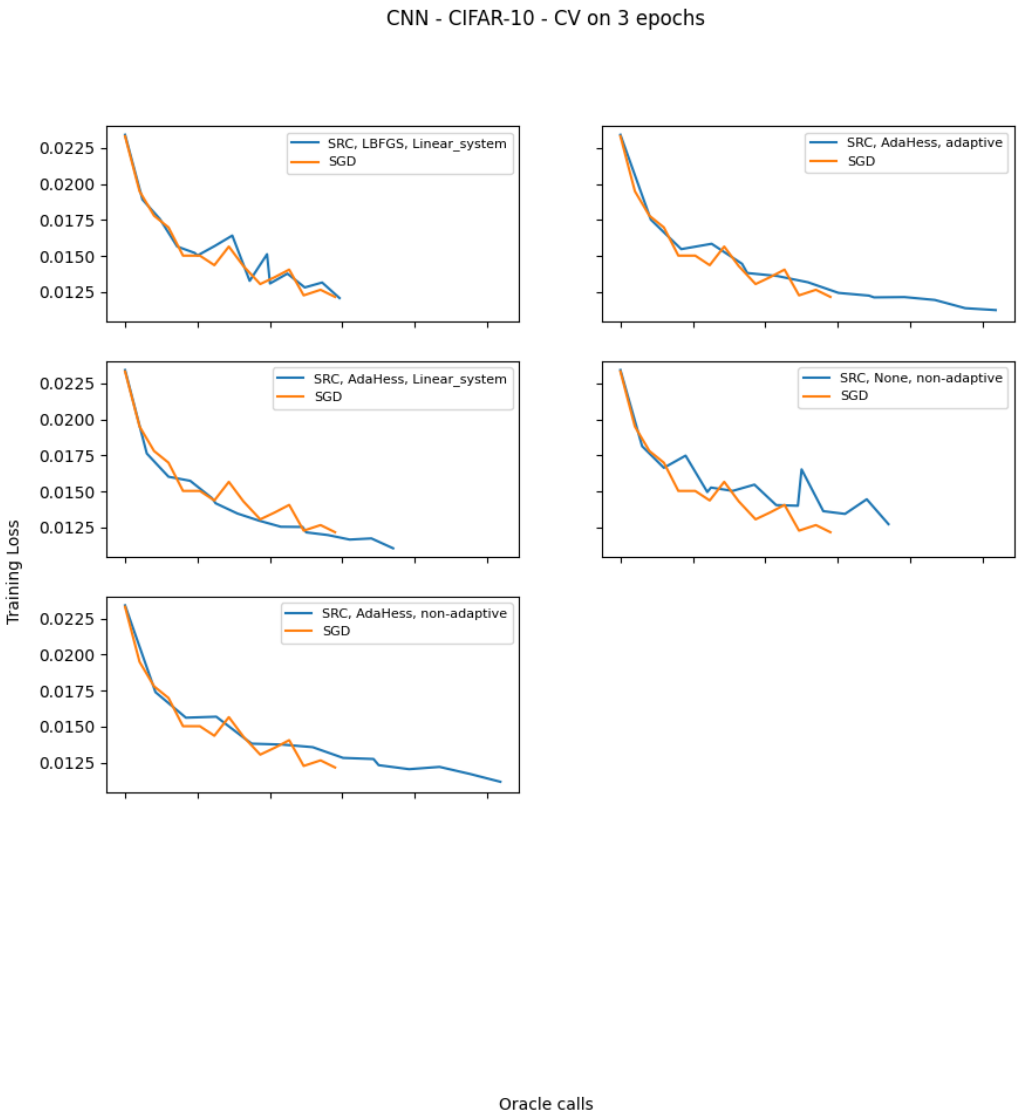


Figure 20: CV on 3 epochs - CNN

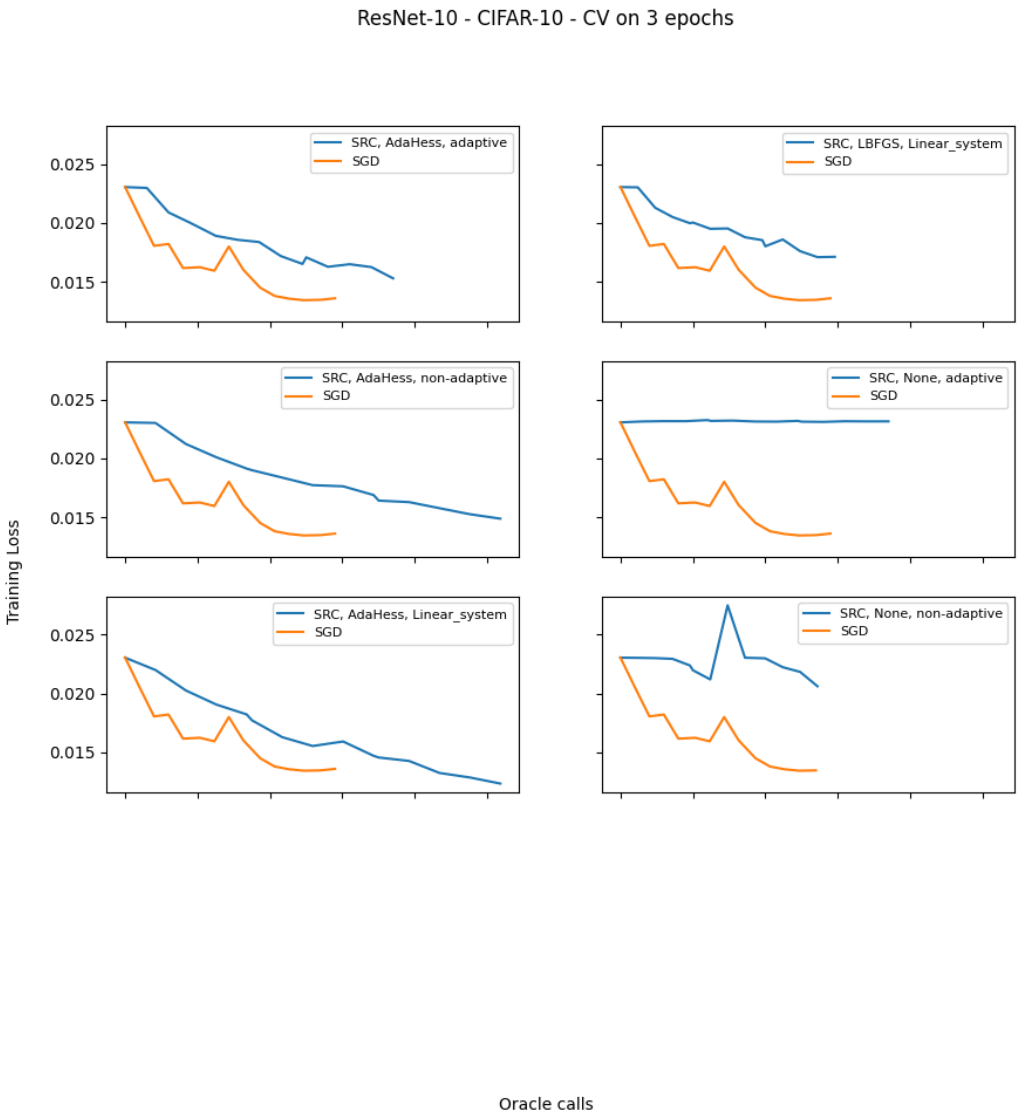


Figure 21: CV on 3 epochs - ResNet-18