Stochastic Second-Order Methods

Alessandro Stanghellini **Joel Vetterlin**

In this report we face the following minimisation problem

$$x^* = \arg\min_x \left[f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) \right] \tag{1}$$

using three different stochastic Second-Order optimisation methods and two different functions f. The first being the logistic loss function of a linear classification model, and the second a non-convex regularised version of the previous one. For the methods we chose a stochastic version of the Newton method (SN), a stochastic Newton method using cubic regularisation (SCN) and a sub-sampled cubic regularisation method (SCR) which has been formulated for non-convex optimisation.

The tests are performed on the A9a and IJCNN1 datasets. Since both of these have binary classes, the loss functions that we use are the following:

$$L(w) = L(y, w) = -\frac{1}{n} \sum_{i=1}^{n} (y_i \log(\sigma(w^T x_i)) + (1 - y_i) \log(1 - \sigma(w^T x_i))) \tag{2}$$

and

$$L_{reg}(w) = L_{reg}(y, w) = L(y, w) + r(w) \tag{3}$$

where

$$r(w) = \lambda \sum_{i=1}^{d} \frac{\alpha w_i^2}{1 + \alpha w_i^2} \quad \text{and} \quad \sigma(z) = \frac{1}{1 + e^{-z}}$$

for $x_i \in \mathbb{R}^d$, $y_i \in \{0,\, 1\}$ with $i = 1, \ldots, n$, $w \in \mathbb{R}^d$ and $\lambda,\, \alpha > 0$ some fixed constants.

Since we will only evaluate on these two Loss functions, which are both twice differentiable, we can compute the gradient and Hessian. However, since all three of our methods are stochastic or sub-sampled methods, we need to compute an approximation of these functions. In particular we consider the following unbiased estimators:

$$g = \nabla L(x_k) \approx g_k := \frac{1}{n} \sum_{i \in S^k} \nabla f_i(x_k) \tag{4}$$

$$H = \nabla^2 L(x_k) \approx H_k := \frac{1}{n} \sum_{i \in S^k} \nabla^2 f_i(x_k) \tag{5}$$

for a subset $S^k \subset \{1, 2, \ldots, n\}$.

1 Stochastic Newton method

Our implementation of the Stochastic Newton Method is very similar to the Newton Method, except that we approximate the Gradient and Hessian, as defined in Equation (4). This approximation is a lot cheaper for very large Datasets. Since we are now in a stochastic environment, in order to keep a decreasing loss per each iteration, we update the weights w_k only for successful iterations. With those two changes to the classic Newton Method we have algorithm 2.

Also notice that we merged the damped and undamped phase of the Newton algorithm by adding a learning rate, which should be set to 1 for the undamped phase in order to achieve local quadratic convergence. We will discuss more about this parameter later in the experiment section.

Algorithm 1 Stochastic Newton (SN)

initialize: weights $w_0 \in \mathbb{R}^d$, minibatch size $\tau \in \{1, 2, \ldots, n\}$
for $k = 0, 1, 2, \ldots$ **do**
    choose random Batch of size τ: $S_k \subset \{1, 2, ..., n\}$
    $H_k = \frac{1}{\tau} \sum_{i \in S_k} \nabla^2 f_i(w_k)$
    $g_k = \frac{1}{\tau} \sum_{i \in S_k} \nabla f_i(w_k)$
    $x_{k+1} = w_k - \lambda H_k^{-1} g_k$
    if $f(x_{k+1}) \leq f(w_k)$ **then**
        $w_{k+1} = x_{k+1}$
    else
        $w_{k+1} = w_k$
    end if
end for

1.1 Convergence analysis

Since we choose a slightly different version of the method presented as Stochastic Newton in Kovalev et al. (2019) we have a convergence result that's slightly different in the formulation but leads to almost the same result. We have proved the following:

Theorem 1 *Let f as in (1) and assume that each f_i is μ strongly convex and has a H-Lipschitz continuous Hessian. Then*

$$E(\|x_{k+1} - x^*\||S_k) \leq \frac{H}{2\mu} E\left(\frac{1}{\tau} \sum_{i \in S_k} \|x_k - x^*\|^2 | S_k\right)$$

where S_k is the random subset of indexes used to estimate the gradient and Hessian and $\tau = |S_k|$. Note that if $\tau = n$ we obtain the quadratic convergence of the non stochastic Newton method.

See the appendix (A.1) for the proof.

2 Stochastic Cubic Newton

The idea behind cubic regularisation is to approximate a regular enough function f with a regularised second-order expansion, namely:

$$f(x) \approx f(x_k) + \nabla f(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f(x_k)(x - x_k) + \frac{M}{6}\|x - x_k\|^3.$$

This is a global upper bound of f for any $M \geq H$, where H is the Lipschitz constant of the Hessian of f. We also define

$$\phi_i^k(x) := f_i(x_k) + \nabla f_i(x_k)^T (x - x_k) + \frac{1}{2}(x - x_k)^T \nabla^2 f_i(x_k)(x - x_k)$$

With this setup we are ready to present our SCN algorithm.

Algorithm 2 Stochastic Cubic Newton (SCN)

initialize: weights $w_0 \in \mathbb{R}^d$, minibatch size $\tau \in \{1, 2, \dots, n\}$
for $k = 0, 1, 2, \dots$ **do**
    choose random Batch of size τ: $S_k \subset \{1, 2, ..., n\}$
    $H_k = \frac{1}{\tau} \sum_{i \in S_k} \nabla^2 f_i(w_k)$
    $g_k = \frac{1}{\tau} \sum_{i \in S_k} \nabla f_i(w_k)$
    $x_{k+1} = \arg\min_{x \in \mathbb{R}^d} \frac{1}{\tau} \sum_{i \in S_k} \big(\phi_i^k(x - w_k) + \frac{M}{6}\|x - w_k\|^3\big)$
    if $f(x_{k+1}) \leq f(w_k)$ **then**
        $w_{k+1} = x_{k+1}$
    else
        $w_{k+1} = w_k$
    end if
end for

Despite it not being immediately explicit we use H_k and g_k in the sum of ϕ_i^k at each iteration. Also, before getting to the convergence analysis, it's worth mentioning that to minimise the intermediate cubic problem we use a standard gradient descent approach, but we are aware about the existence of possibly more efficient methods. This choice was made to reduce slightly the amount of new arguments for this presentation since we believed that we covered a wide enough amount of topics and not for a lack of interest.

2.1 Convergence analysis

Similarly as before, we propose a variation of the result displayed in Kovalev et al. (2019), which is actually an adapted version of an intermediate result.

Theorem 2 *Let f as in (1) and assume that each f_i is μ strongly convex and has a H-Lipschitz continuous Hessian. Choose $M \geq H$, then*

$$E_k(f(x_{k+1}) - f(x^*)) \leq \frac{\sqrt{2}(M + H)}{3\mu^{\frac{3}{2}}} E_k\big(\frac{1}{\tau} \sum_{i \in S_k} \|f_i(x_k) - f_i(x^*)\|\big).$$

For the proof one can see Lemma 4,5 from the appendix of Kovalev et al. (2019).

3 Sub-sampled cubic regularisation method

Similarly as what has been done in the previous section, we use unbiased estimates of both gradient and Hessian but this time constructed using two independent samples of points which we denote as S_g and S_B. Again we want to minimise a cubic model at each iteration, namely:

$$m_k(s_k) := f(x_k) + s_k^T g_k + \frac{1}{2} s_k B_k s_k + \frac{\sigma_k}{3}\|s_k\|^3 \tag{6}$$

where

$$g_k = \frac{1}{|S_g|} \sum_{i \in S_g} \nabla f_i(x_k)$$

and

$$B_k := \frac{1}{|S_B|} \sum_{i \in S_B} \nabla^2 f_i(x_k).$$

Then the model gradient with respect to s_k is

$$\nabla m_k(s_k) = g_k + B_k s_k + \sigma_k s_k \|s_k\| \tag{7}$$

The method that we used is the one formulated in Kohler and Lucchi (2017b) and it's the following. At each iteration we sub-sample two sets of datapoints that we use to calculate the estimates of the gradient and Hessian as shown previously. Then we approximately minimise (6) using Cauchy points

obtained by globally minimise the same model along the current negative gradient direction. Then updates the regularisation parameter σ_k based on how well the model approximates the real objective. This allow us to have longer or shorter step depending if the iteration was successful or unsuccessful respectively.

Algorithm 3 Sub-sampled Cubic Regularisation (SCR)

1: **Input:** Starting point $x_0 \in \mathbb{R}^d$, $\gamma > 1,\ 1 > \eta_2 > \eta_1 > 0$ and $\sigma_0 > 0$
2: **for** $k = 0, 1, 2, \dots$ **do**
3: Sample gradient g_k and Hessian H_k
4: Obtain s_k by approximately solving $m_k(s_k)$ (6)
5: Compute $f(x_k + s_k)$ and

$$\rho_k = \frac{f(x_k) - f(x_k + s_k)}{f(x_k) - m_k(s_k)} \tag{8}$$

6: Set

$$x_{k+1} = \begin{cases} x_k + s_k & \text{if } \rho_k \geq \eta_1 \\ x_k & \text{otherwise} \end{cases} \tag{9}$$

7: Set

$$\sigma_{k+1} = \begin{cases} \max\{\min\{\sigma_k, \|g_k\|\}, \epsilon_m\} & \text{if } \rho_k > \eta_2 \ \text{ very successful iteration} \\ \sigma_k & \text{if } \eta_2 \geq \rho_k \geq \eta_1 \ \text{ successful iteration} \\ \gamma\sigma_k & \text{otherwise } \ \text{unsuccessful iteration,} \end{cases} \tag{10}$$

where $\epsilon_m \approx 10^{-16}$ is the relative machine precision.
8: **end for**

The Cauchy point mentioned in the algorithm are calculated as follows

$$s_k = -\alpha_k g_k$$

where

$$\alpha_k = \arg\min_{\alpha > 0} m_k(-\alpha g_k).$$

We then using (6) have that

$$m_k(-\alpha g_k) = f(x_k) - \alpha g_k^T g_k + \frac{\alpha^2}{2} g_k B_k g_k - \alpha^3 \frac{\sigma_k}{3} \|g_k\|^3$$

and derivating with respect to α we obtain

$$\frac{\partial}{\partial \alpha} m_k(\alpha) = -\|g_k\|^2 + \alpha g_k B_k g_k + \alpha^2 \sigma_k \|g_k\|^3.$$

Setting this to be equal to 0 and solving for α we find two real values and we choose the largest of the two, namely

$$\alpha_k = \frac{-g_k B_k g_k + \sqrt{(g_k B_k g_k)^2 - 4\|g_k\|^5}}{2\|g_k\|^3}.$$

As shown in both Cartis C. (2011) and Kohler and Lucchi (2017b), one can actually calculate the exact minimum point or use more precise approximation methods. However, the methods that are used to perform this improved approximation are a bit out of scope for the interest of this report so we will limit our self to this more accessible approach.

3.1 Convergence analysis

First of all we need to state some assumptions.

Assumption 3 (Approximate model minimizer)

$$s_k^T g_k + s_k^T B_k s_k + \sigma_k \|s_k\|^3 = 0 \tag{11}$$

$$s_k^T B_k s_k + \sigma_k \|s_k\|^3 \geq 0 \tag{12}$$

Assumption 4 (Continuity) *The functions $f_i \in C^2(\mathbb{R}^d)$, g_i and H_i are Lipschitz continuous for all i, with Lipschitz constants κ_f, κ_g and κ_H respectively.*

We also require that at each iteration the approximation of the gradient and Hessian are reasonably accurate.

Assumption 5 (Sufficient Agreement of H and B)

$$\|(B_k - H(x_k))s_k\| \leq C\|s_k\|^2, \ \forall k \geq 0, C > 0. \tag{13}$$

Assumption 6 (Sufficient Agreement of ∇f and g)

$$\|\nabla f(x_k) - g(x_k)\| \leq M\|s_k\|^2, \ \forall k \geq 0, \ M > 0. \tag{14}$$

Observe that (4) can be easily extended g_k and B_k for any sample size. In fact, using the triangular inequality we have, for example,

$$\begin{aligned}\|g_k(x) - g_k(y)\| &= \|\frac{1}{|S_g|}\sum_{i\in S_g} \nabla f_i(x) - \nabla f_i(y)\| \leq \frac{1}{|S_g|}\sum_{i\in S_g}\|\nabla f_i(x) - \nabla f_i(y)\| \\ &\leq \kappa_g\|x - y\|\end{aligned}$$

These last two assumptions are fulfilled with high probability under specific sampling conditions. In particular we can state the following theorems.

Theorem 7 *If*

$$|S_{g,k}| \geq \frac{32\kappa_f^2(\log((2d)/\delta) + 1/4)}{M^2\|s_k\|^4}, \ M \geq 0, \ \forall k \geq 0 \tag{15}$$

then g_k satisfies the sufficient agreement condition (6) with probability $1 - \delta$.

Theorem 8 *If*

$$|S_{B,k}| \geq \frac{12\kappa_g^2(\log(2d/\delta)}{(C\|s_k\|)^2}, \ C \geq 0, \ \forall k \geq 0 \tag{16}$$

then B_k satisfies the sufficient agreement condition (5) with probability $1 - \delta$.

Both these result rely on concentration inequalities for vector and matrix random variables respectively and we refer to Kohler and Lucchi (2017b) for detailed proofs of these results.

From this we can also deduce that the sample sizes grow with d, κ_f and κ_g. Also the step size tends to zero as the method reaches a second-order critical point, thus sample sizes must grow up to n with the number of iterations, which clearly means that both the estimates of the gradient and Hessian converge to the true values.

Finally, convergence results stated in Kohler and Lucchi (2017b) refers mainly to the SCR algorithm using more elaborated methods to approximate a solution of (6) as mentioned previously. Due to this slight modification we well briefly present the results proposed in the aforementioned paper.

First of all,it is shown (Theorem 13) that under the additional assumption of convergence to some x^*, the rate of convergence is locally quadratic, with high probability.

Furthermore, if the sequence $\{f(x_k)\}$ is bounded by some $f_{inf} > -\infty$, then the convergence to a first order critical point is guaranteed, namely

$$\lim_{k\to\infty} \|\nabla f(x_k)\| = 0.$$

Lastly, the most important result states that the global convergence is actually to a second-order point, provided that such point exists.

4 Experiment

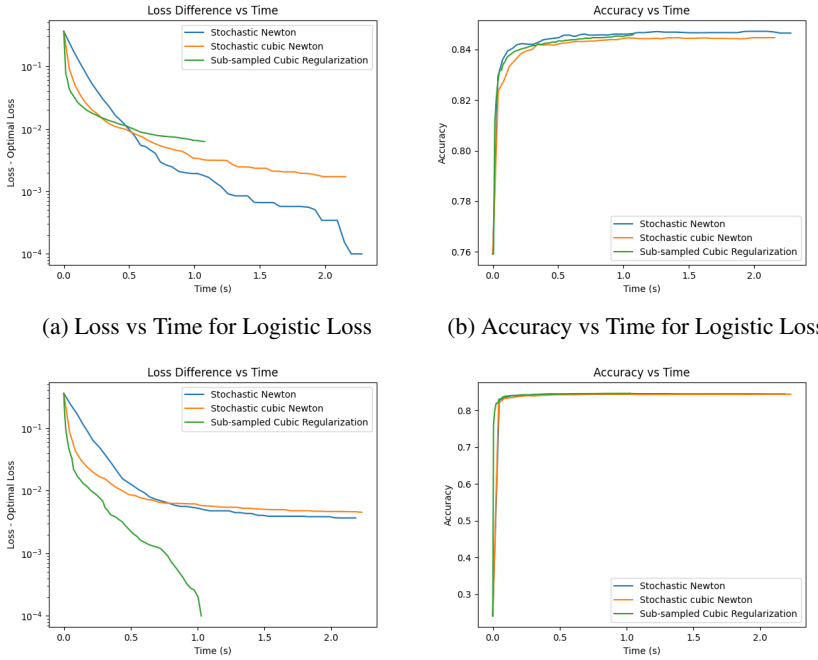
4.1 Set-up

For our experiment we evaluate the 3 mentioned Methods: Stochastic Newton, Stochastic cubic newton and Sub-sampled cubic Regularisation. We test those 3 methods on 2 different Datasets , A9a and IJCNN1, each consisting of 2 classes, as well as 2 different Loss functions. We analyse 2 key parts, where we compare the convergence and accuracy of the methods, and one where we analyse the difference between the Loss functions for each methods.
As mentioned in section (1), we have introduced a learning rate λ to our Stochastic Newton. We have analyzed the use of this learning rate in appendix (A.2), which let us to choosing $\lambda = 0.25$ for the Dataset IJCNN1 and $\lambda = 0.1$ for Dataset A9a. We also analysed different M-values in appendix (A.3), which let us to using $M = 0.02$, which is a similar conclusion to Kovalev et al. (2019).

4.2 Method Analysis

In Figure (1), we immediately see a difference in time for the calculations, SCR is much faster. This is mainly because the code for SCR, which we got from Kohler and Lucchi (2017a), is more optimised than our handwritten code from SN and SCN.There is a clear difference in convergence for the regularised logistic loss and the Logistic loss. For the Logistic loss Stochastic newton and Stochastic cubic newton have a better convergence rate than SCR. However for the regularised logistic loss, SCR is much better. This is because the regularised logistic loss non-convex and SN/SCN both require convexity for global convergence, whereas SCR is built to be able to handle non-convexity.

Figure (2), shows the same trends again but now even clearer. Again for the Logistic loss SN and SCN are better, since they have their required convexity in the loss function, and for the regularised logistic loss SCR is far superior.



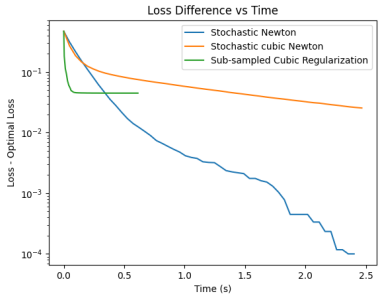
(a) Loss vs Time for Logistic Loss

(b) Accuracy vs Time for Logistic Loss

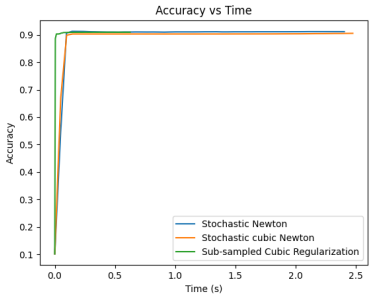
(c) Loss vs Iterations for Regularised logistic Loss

(d) Accuracy vs Iterations for Regularised logistic Loss

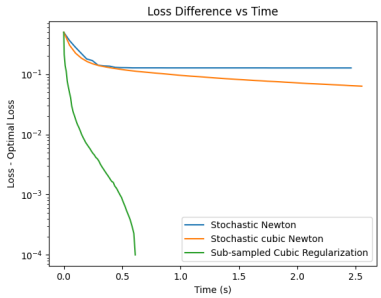
Figure 1: Comparing all Methods on the A9a Dataset



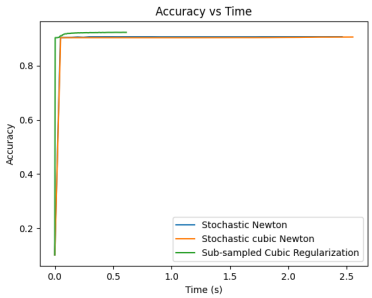
(a) Loss vs Time for Logistic Loss



(b) Accuracy vs Time for Logistic Loss



(c) Loss vs Iterations for Regularised logistic Loss



(d) Accuracy vs Iterations for Regularised logistic Loss

Figure 2: Comparing all Methods on the IJCNN1 Dataset

4.3 Loss function Analysis

We could have applied the aforementioned methods since the chosen loss function can be expressed as an averaged sum. In particular we have considered the functions in (2) and (3). Clearly they are both very regular, for sure at least $C^2(\mathbb{R}^d, \mathbb{R})$. In fact we have

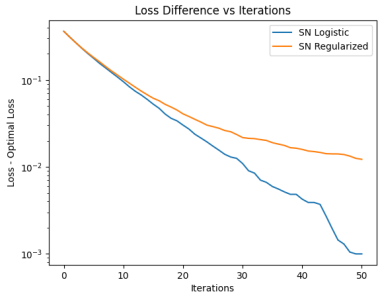
$$\nabla L_i(w) = -y_i(1-\sigma(z_i))x_i + (1-y_i)\sigma(z_i)x_i = x_i(\sigma(z_i)-y_i) \quad \nabla^2 L_i(w) = x_i x_i^T \sigma(z_i)(1-\sigma(z_i))$$

and

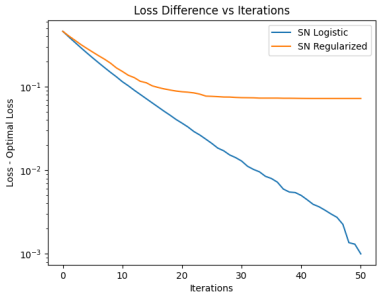
$$\frac{\partial}{\partial w_i} r(w) = 2\lambda \frac{\alpha w_i}{(\alpha w_i^2 + 1)^2} \quad \frac{\partial^2}{\partial w_i^2} r(w) = \lambda \frac{2\alpha - \alpha w_i^2}{(\alpha w_i^2 + 1)^3}.$$

It's clear that L is convex and that instead L_{reg} isn't convex. Both the Hessian matrices are also Lipschitz continuous since they are uniform continuous and the constant depends on the dataset and also on the parameters of the regularisation term for the second one.

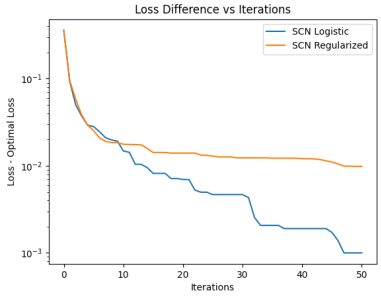
Figure (3) shows the difference between the two Loss functions we analysed between each of the methods and datasets. For Stochastic newton and Stochastic cubic newton which both require convexity, the non-convex regularised logistic loss' improvement slows down at some iteration, whereas the convex Logistic loss keeps improving. This phenomenon is caused by getting stuck in a local minimum instead of going to the global minimum. This is different for the Sub-sampled cubic regularisation. Because this method can handle non-convexity the two losses are either very similar, such as with the A9a dataset, or the non-convex regularised logistic loss is clearly better.



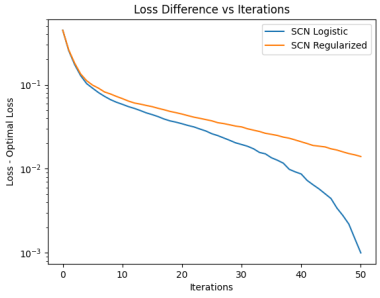
(a) Loss vs Iterations for SN on A9a



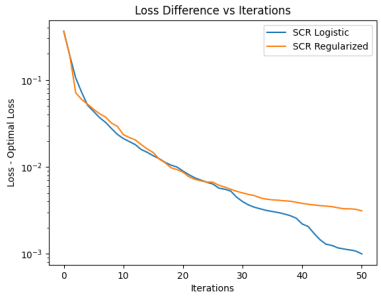
(b) Loss vs Iterations for SN on IJCNN1



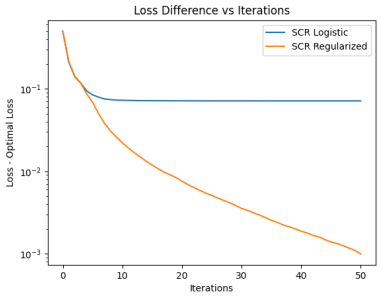
(c) Loss vs Iterations for SCN on A9a



(d) Loss vs Iterations for SCN on IJCNN1



(e) Loss vs Iterations for SCR on A9a



(f) Loss vs Iterations for SCR on IJCNN1

Figure 3: Comparing the two losses for each Method and dataset

5 Conclusions

In this project we looked at three different methods, two of which require convexity and one who doesn't. We tested these methods on two datasets and two loss functions. In this analysis we saw, as expected, that stochastic Newton as well as stochastic cubic newton perform better with the convex logistic loss compared to the sub-sampled cubic regularisation. On the other side, the SCR performed better than the SN and SCN for the non-convex regularised logistic loss.

This project can be expanded in various directions. For example, further optimisers could be considered. Also with more datasets, the impact on the methods could be more closely examined. To study how the achieved accuracy impacts different variables such as the learning rate, or which methods can handle noisier data better.

References

Cartis C., G. N. . T. P. (2011). Adaptive cubic regularisation methods for unconstrained optimization. part i: motivation, convergence and numerical results.

Kohler, J. M. and Lucchi, A. (2017a). Github folder for sub-sampled cubic regularization. `https://github.com/dalab/subsampled_cubic_regularization/`.

Kohler, J. M. and Lucchi, A. (2017b). Sub-sampled cubic regularization for non-convex optimization.

Kovalev, D., Mishchenko, K., and Richtárik, P. (2019). Stochastic newton and cubic newton methods with simple local linear-quadratic rates.

A Appendix

A.1 Proofs

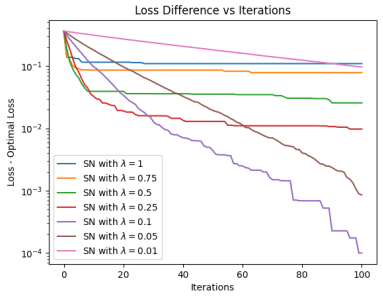
Proof of Theorem 1: Let $E_k = E(\cdot|S_k)$. We have

$$\begin{aligned}
E_k(\|x_{k+1} - x^*\|) &= E_k(\|x_k - x^* - H_k^{-1} g_k\|) \\
&= E_k\left(\|\frac{1}{\tau}\sum_{i\in S_k}(\nabla^2 f_i^{-1}(x_k)(\nabla^2 f_i(x_k)(x_k - x^*) - (\nabla f_i(x_k) - \nabla f_i(x^*)))\|\right) \\
&\leq E_k\left(\frac{1}{\tau}\sum_{i\in S_k}\|(\nabla^2 f_i^{-1}(x_k))(\nabla^2 f_i(x_k)(x_k - x^*) - (\nabla f_i(x_k) - \nabla f_i(x^*)))\|\right) \\
&\leq E_k\left(\frac{1}{\tau}\sum_{i\in S_k}\|\nabla^2 f_i^{-1}(x_k)\|\|\nabla^2 f_i(x_k)(x_k - x^*) - (\nabla f_i(x_k) - \nabla f_i(x^*))\|\right) \\
&\leq E_k\left(\frac{1}{\tau}\sum_{i\in S_k}\frac{1}{\mu}\frac{H}{2}\|x_k - x^*\|^2\right).
\end{aligned}$$

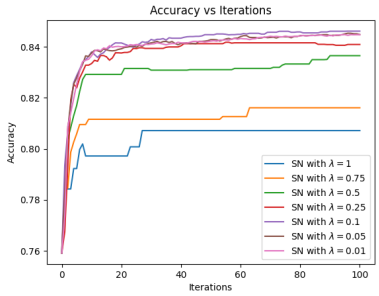
Here the fact that $\|\nabla^2 f_i^{-1}(x_k)\| \leq 1/\mu$ follows from the μ strongly convex property and from H-Lipschitzness of the Hessian Matrices we have

$$\|\nabla^2 f_i(x_k)(x_k - x^*) - (\nabla f_i(x_k) - \nabla f_i(x^*))\| \leq \frac{H}{2}\|x_k - x^*\|^2.$$

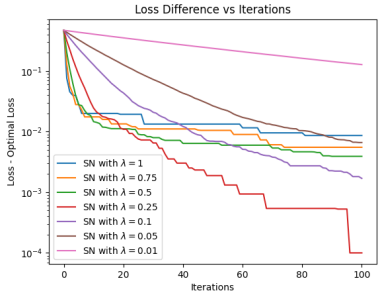
A.2 Learning Rate of Stochastic Newton



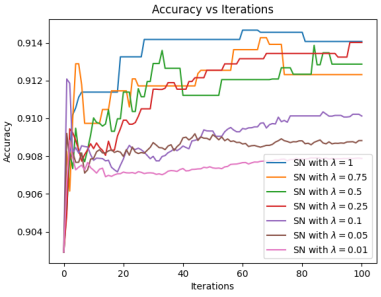
(a) Loss vs Iterations for Dataset 'A9a'



(b) Accuracy vs Iterations for Dataset 'A9a'



(c) Loss vs Iterations for Dataset 'IJCNN1'



(d) Accuracy vs Iterations for Dataset 'IJCNN1'

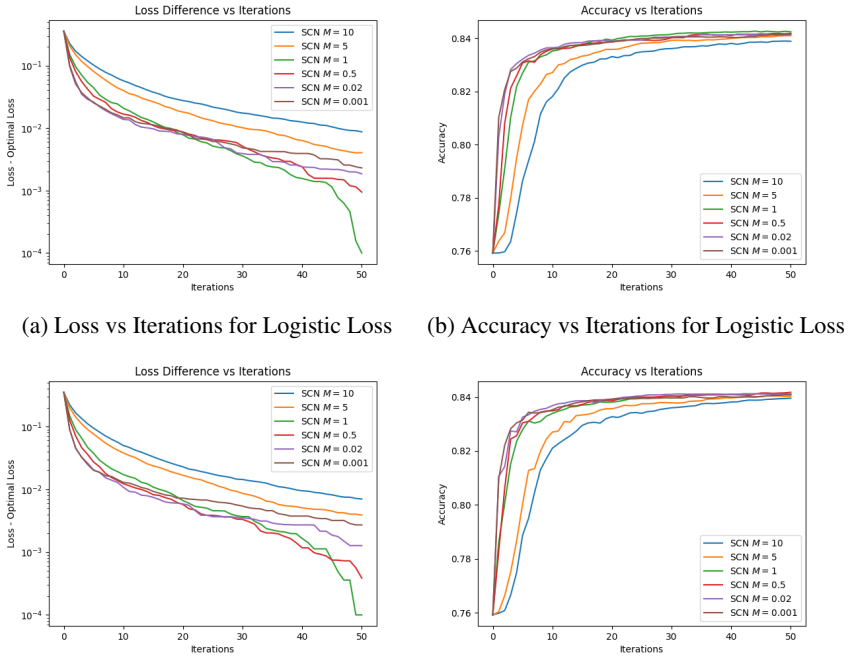
Figure 4: Testing for multiple Learning rates of Stochastic Newton

Traditionally Stochastic Newton does not have a Learning rate. However, we have introduced one in the following way:

$$x_{k+1} = x_k - \lambda H_k g_k$$

The reason being, is that because of the stochastic approach we noticed that after some approximating the chance for a successful iteration (meaning a iteration which reduces the loss function) becomes smaller and smaller. However, if we scale down the step size, this chance for a successful iteration does not become as small. This is likely the case because when approximating the gradient and hessian with a sub-sample a deviation in the direction of the step has a much higher chance to "over-shoot" the slope for larger step sizes than for smaller ones.
In Figure (4) the precision for smaller learning rates is clearly visible. The optimal learning rate will depend on the dataset and the number of epochs of the optimisation. The A9a dataset is less accurate, about 84%, for this $\lambda = 0.1$ for longer runs resp. $\lambda = 0.25$ for shorter runs are best. But for the IJCNN1 dataset, which is more accurate, we can afford to take slightly larger Learning rates.

A.3 M values for Stochastic Cubic Newton



(a) Loss vs Iterations for Logistic Loss (b) Accuracy vs Iterations for Logistic Loss

(c) Loss vs Iterations for Regularised logistic Loss (d) Accuracy vs Iterations for Regularised logistic Loss

Figure 5: Testing for multiple M Values of Stochastic cubic Newton

The Stochastic cubic Newton method uses a variable M, which is an upper bound on the Hessian's Lipschitz constant. To find which M values are good values for the main part of our experiment, we Tested multiple values with a high amount of repetitions to get a good average for that value. The values which stand out are $M = 0.02$ and $M = 1$. $M = 1$ becomes more accurate after about 30 epochs, but in the first 30 epochs $M = 0.02$ is better, and thus we choose $M = 0.02$ for our experiment setting.