Global Convergence Newton

Raffael Colonnello
University of Basel
Raffael.Colonnello@unibas.ch

Fynn Gohlke
University of Basel
Fynn.Gohlke@stud.unibas.ch

Benedikt Heuser
University of Basel
ben.heuser@unibas.ch

Abstract

The abstract paragraph should be indented ½ inch (3 picas) on both the left- and right-hand margins. Use 10 point type, with a vertical spacing (leading) of 11 points. The word **Abstract** must be centered, bold, and in point size 12. Two line spaces precede the abstract. The abstract must be limited to one paragraph.

1 Introduction

In this paper we consider problems of the form

$$\min_{x \in \mathbb{R}^d} f(\mathbf{x}) \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ is a twice-differentiable function. First-order optimization methods are widely used for such problems due to their low per-iteration computational cost and their suitability for parallelization. They often suffer from slow convergence for ill-conditioned objective functions [1]. Newton's method is a popular optimization algorithm that is commonly used to solve optimization problems. It is a second-order optimization algorithm since it uses second-order information of the objective function. Newton's method is known to have fast local convergence guarantees for convex functions. However, the global convergence properties of Newton's method are still an active area of research [2] [3]. In contrast to first-order methods like gradient descent, second-order methods, such as Newton's method can achieve much faster convergence when presented with ill conditioned Hessians by transferring the problem into a more isotropic optimization problem at the cost of an increase to cubic run time. Newton's method yields local quadratic convergence if f is twice differentiable (or we have suitable regularity conditions), which degrade outside of the local regions, yielding up to sublinear global convergence guarantees, depending on the alogithm.

In this paper, we explore the theoretical foundations of several Newton-type methods that achieve different global convergence guarantees, compare their performance in a classification-type problem for two loss functions on four different datasets. Finally we will propose two modifications of the algorithms to achieve an increase in runtime, by either coupling the Newton-type method with a conjugate gradient method for Hessian vector multiplication or Strassen's algorithm for fast matrix inversion.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

2 Background

2.1 Loss function and Datasets

Let $X = \begin{bmatrix} \dots x_1^\top \dots \\ \vdots \\ \dots x_i^\top \dots \\ \vdots \\ \dots x_n^\top \dots \end{bmatrix} \in \mathbb{R}^{n\times d}$ be the set of data for n datapoints with d features, i.e. $x_i \in \mathbb{R}^d$ and labels $y^\top = [y_1, ..., y_n]$

For $\sigma(x) := \frac{\exp(x)}{1+\exp(x)}$ the loss functions w.r.t. weights ω are given by

$$L_1(\omega) = -\frac{1}{n}\sum_{i=1}^{n}\Big(y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)\Big), \quad \hat{y}_i = \sigma(x_i^\top \omega) \tag{2}$$

$$L_2(\omega) = \frac{1}{n}\sum_{i=1}^{n}\log\big(1+\exp(-y_i x_i^\top \omega)\big) + r(\omega), \quad r(\omega) = \lambda\sum_{j=1}^{d}\frac{\alpha\omega_j^2}{1+\alpha\omega_j^2} \tag{3}$$

$$\tag{4}$$

which yields the two optimization problems

$$\min_\omega L_1(\omega) \tag{5}$$

$$\min_\omega L_2(\omega) \tag{6}$$

Remark 1: The 0-1 loss function for logistic regression is given by

$$-\sum_{i=1}^{N}\log\left[\mu_i^{\mathbb{I}(y_i=1)}(1-\mu_i)^{\mathbb{I}(y_i=0)}\right] = -\sum_{i=1}^{N}[y_i\log\mu_i + (1-y_i)\log(1-\mu_i)]$$

for labels $y_i \in \{0,1\}$ [4, Eq. 8.2–8.3]. If we instead use labels $\tilde{y}_i \in \{-1,+1\}$, the negative log-likelihood becomes

$$\sum_{i=1}^{N}\log\big(1+\exp(-\tilde{y}_i\,\mathbf{w}^T\mathbf{x}_i)\big)$$

[4, Eq. 8.4]. To ensure the loss functions correspond to the correct likelihood, the label encoding must match the loss form [4, Sec. 8.3.1].

The corresponding gradients of L_i are

$$\nabla L_1(x) = \frac{1}{n}X^\top(\hat{y}-y) \tag{7}$$

$$\nabla L_2(x) = -\frac{1}{n}X^\top\big(y\odot\sigma(-y\odot(X\omega))\big) + \nabla r(x) \tag{8}$$

with $\nabla r(\omega)^\top = \lambda\left[\frac{2\alpha\omega_1}{(1+\alpha\omega_1^2)^2}, \dots, \frac{2\alpha\omega_d}{(1+\alpha\omega_d^2)^2}\right]$, where $\sigma(\cdot)$ is applied elementwise, and $\odot$ denotes the entrywise multiplication of vectors.

Differentiating again yields the Hessians

$$\nabla^2 L_1(\omega) = \frac{1}{n}X^\top DX \tag{9}$$

$$\nabla^2 L_2(\omega) = \frac{1}{n}X^\top DX + \nabla^2 r(\omega), \quad r(\omega) = \operatorname{diag}\left(\lambda\frac{2\alpha(1-3\alpha\omega_j^2)}{(1+\alpha\omega_j^2)^3}\right) \tag{10}$$

where the diagonal matrix D has entries

$$D_{ii} = \hat{y}_i(1-\hat{y}_i) = \sigma(-y_i x_i^\top\omega)\big(1-\sigma(-y_i x_i^\top\omega)\big), \tag{11}$$

Observation:
Since $\log(\hat{y}_i), \log(1-\hat{y}_i)$ are concave on $(0,\infty)$ it follows that $-\log(\hat{y}_i), -\log(1-\hat{y}_i)$ are convex and thus L_1 is a linear combination of convex functions (which is again convex). Meanwhile L_2 is not guaranteed to be convex due to the non-convex regularization term $r(\omega)$.

2.2 Classic Newton's Method

The classical origin of Newton's method is as an algorithm for finding the roots of functions. In this paper it is used to find the roots x^* of $\nabla(f(x))$ $s.t. \nabla(f(x^*)) = 0$ and x^* a local minimum of f. Newton's method combined with a stepsize η uses the update rule [1]:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \eta(\nabla^2 f(\mathbf{x}_k))^{-1} \nabla f(\mathbf{x}_k) \tag{12}$$

The inverse Hessian can be interpreted as transforming the gradient landscape to be more isotropic, thereby improving the conditioning of the problem.

2.3 Cubic Newton

AICN gibt sich zwar als regularized method aus, kann in Wirklichkeit aber nicht umgehen die Matrix trotzdem zur Berechnung des Faktors Alpha invertieren zu müssen. Es kämpft deshalb für singulare oder illcondiitoned matrizen mit genau denselben problemen, wie unregularisierte Methoden. Kann man das sicher nicht umgehen, dass man für das Alpha das Skalarprodukt invertieren muss

2.4 Cubic Newton

The cubic Newton method was one of the first to achieve a good complexity guarantee globally [REFERENCE TO DO: What convergence rate exactly?]. It is based on cubic regularization and uses the update rule:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\nabla^2 f(\mathbf{x}_k) + H||\mathbf{x}_{k+1} - \mathbf{x}_k||\mathbf{I})^{-1} \nabla f(\mathbf{x}_k) \tag{13}$$

2.5 Levenberg and Marquardt method

The Levenberg-Marquardt's algorithm [REFERENCE] is an early form of regularized Newton's method that modifies the Hessian. For ill conditioned (or singular) H regularization can increase the conergence (or make the problem solvable as $H + \lambda I$ is always invertible for sufficiently large $eig(H) > -\lambda, \lambda > 0$). The update rule is:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\nabla^2 f(\mathbf{x}_k) + \lambda_k \mathbf{I})^{-1} \nabla f(\mathbf{x}_k) \tag{14}$$

2.6 Regularized Newton

In their 2023 article Michenko presents a variation of Newton's method that uses the update rule [2]:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - (\nabla^2 f(\mathbf{x}_k) + \sqrt{H||\nabla f(\mathbf{x}_k)||}\mathbf{I})^{-1} \nabla f(\mathbf{x}_k) \tag{15}$$

where $H > 0$ is a constant. The convergence rate of this algorithm is $\mathcal{O}(\frac{1}{k^2})$. This method uses an adaptive variant of the Levenberg-Marquardt regularization.

2.7 Appendix

Remark 2:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

$$\implies \frac{d}{dz}\sigma(z) = \frac{d}{dz}(1+e^{-z})^{-1} = -(1+e^{-z})^{-2} \cdot (-e^{-z}) = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} \cdot \frac{e^{-z}}{1+e^{-z}} = \sigma(z)(1-\sigma(z))$$

$$L_1(\omega) = -\frac{1}{n}\sum_{i=1}^{n}\Big[\underbrace{y_i \log \hat{y}_i}_{=:A_i} + \underbrace{(1-y_i)\log(1-\hat{y}_i)}_{=:B_i}\Big]$$

$$\hat{y}_i = \sigma(x_i^\top \omega) = \frac{1}{1+e^{-x_i^\top \omega}}$$

and applying Remark 2 to $\hat{y}$ we get, that

$$
\begin{aligned}
\frac{\partial}{\partial\omega}A_i &= \frac{\partial}{\partial\omega}\big(-y_i\log\hat{y}_i\big) = -y_i\frac{1}{\hat{y}_i}\hat{y}_i(1-\hat{y}_i)x_i = -y_i(1-\hat{y}_i)x_i \\
\frac{\partial}{\partial\omega}B_i &= \frac{\partial}{\partial\omega}\big(-(1-y_i)\log(1-\hat{y}_i)\big) = (1-y_i)\frac{1}{1-\hat{y}_i}\hat{y}_i(1-\hat{y}_i)x_i = (1-y_i)\hat{y}_i x_i \\
\frac{\partial}{\partial\omega}A + \frac{\partial}{\partial\omega}B &= -y_i(1-\hat{y}_i)x_i + (1-y_i)\hat{y}_i x_i = \big(-y_i + y_i\hat{y}_i + \hat{y}_i - y_i\hat{y}_i\big)x_i \\
&= (-y_i + \hat{y}_i)x_i = (\hat{y}_i - y_i)x_i \\
\implies \nabla L_1(\omega) &= \frac{1}{n}\sum_{i=1}^{n}\frac{\partial}{\partial\omega}A_i + \frac{\partial}{\partial\omega}B_i = \frac{1}{n}\sum_{i=1}^{n}\big[\hat{y}_i - y_i\big]x_i = \frac{1}{n}X^\top(\hat{y}-y)
\end{aligned}
$$

For the Hessian it then follows

$$
\begin{aligned}
\nabla^2_\omega L_1(\omega) &= \nabla_\omega\frac{1}{n}X^\top(\hat{y}-y) = \frac{1}{n}X^\top = \nabla_\omega(\hat{y}-y) = \frac{1}{n}X^\top\nabla_\omega\hat{y} \\
\frac{\partial}{\partial\omega}\big(\hat{y}_i x_i\big) &= \hat{y}_i(1-\hat{y}_i)x_i x_i^\top \\
\implies \frac{\partial\hat{y}}{\partial\omega} &= \operatorname{diag}\big(\sigma(X\omega)\odot(1-\sigma(X\omega))\big)X \\
\implies \nabla^2 L_1(\omega) &= \frac{1}{n}X^\top\operatorname{diag}(\hat{y}\odot(1-\hat{y}))X \\
\implies \nabla^2 L_1(\omega) &= \frac{1}{n}X^\top D X \\
D &= \operatorname{diag}(\hat{y}_i(1-\hat{y}_i)).
\end{aligned}
$$

For L_2 we have

$$
L_2(\omega) = \frac{1}{n}\sum_{i=1}^{n}\underbrace{\log\big(1+\exp(-y_i x_i^\top\omega)\big)}_{f_i(\omega)} + \underbrace{\lambda\sum_{j=1}^{d}\frac{\alpha\omega_j^2}{1+\alpha\omega_j^2}}_{r(\omega)}
$$

For the gradient we then get

$$
\begin{aligned}
\frac{\partial}{\partial\omega_j}r(\omega) &= 2\lambda\alpha\frac{\omega_j}{(1+\alpha\omega_j^2)^2} \implies \nabla r(\omega) = 2\lambda\alpha\frac{\omega}{(1+\alpha\omega^2)^2} \\
\nabla f_i(\omega) &= \frac{\partial}{\partial\omega}\log(1+e^{-y_i x_i^\top\omega}) \\
&= \underbrace{\frac{1}{1+e^{y_i x_i^\top\omega}}}_{\sigma(-y_i x_i^\top\omega)}\cdot(-y_i x_i) = \sigma(-y_i x_i^\top\omega)\cdot(-y_i x_i) = -y_i x_i\,\sigma(-y_i x_i^\top\omega) \\
\nabla f(\omega) &= -\frac{1}{n}\sum_{i=1}^{n}y_i x_i\sigma(-y_i x_i^\top\omega) = -\frac{1}{n}X^\top\big(y\odot\sigma(-y\odot(X\omega))\big) \\
\nabla L_2(\omega) &= \nabla f(\omega) + \nabla r(\omega) \\
&= -\frac{1}{n}X^\top\big(y\odot\sigma(-y\odot(X\omega))\big) + 2\lambda\alpha\frac{\omega}{(1+\alpha\omega^2)^2}
\end{aligned}
$$

For the Hessians we first observe two remarks:

Remark 3: By chain rule we have

$$
\begin{aligned}
z_i(\omega) &:= -y_i x_i^\top\omega \\
\implies \nabla_\omega z_i(\omega) &= -y_i x_i \\
\implies \nabla_\omega\sigma(z_i(\omega)) &= \sigma'(z_i(\omega))\nabla_\omega z_i(\omega) \\
&= \sigma\big(-y_i x_i^\top\omega\big)\big(1-\sigma\big(-y_i x_i^\top\omega\big)\big)\big(-y_i x_i\big)
\end{aligned}
$$

From the gradient we have

$$\nabla^2_\omega f(\omega) = \nabla_\omega \left(-\frac{1}{n} X^\top \big(y \odot \sigma(-y \odot (X\omega))\big) \right) = -\frac{1}{n} X^\top \nabla_\omega \big(y \odot \sigma(-y \odot (X\omega))\big)$$

Now notice, that

$$y \odot \sigma(-y \odot (X\omega)) = \begin{pmatrix} y_1 \sigma(-y_1 x_1^\top \omega) \\ y_2 \sigma(-y_2 x_2^\top \omega) \\ \vdots \\ y_n \sigma(-y_n x_n^\top \omega) \end{pmatrix}$$

and applying Remark 3 yields

$$\begin{aligned} \nabla_\omega \sigma(-y_i x_i^\top \omega) &= \sigma(-y_i x_i^\top \omega)\big(1 - \sigma(-y_i x_i^\top \omega)\big)(-y_i x_i) \\ \implies \nabla_\omega \big(y_i\, \sigma(-y_i x_i^\top \omega)\big) &= -\underbrace{y_i^2}_{=1 \text{ by Remark 1}} \sigma(-y_i x_i^\top \omega)\big(1 - \sigma(-y_i x_i^\top \omega)\big) x_i = -\sigma(-y_i x_i^\top \omega)\big(1 - \sigma(-y_i x_i^\top \omega)\big) x_i \end{aligned}$$

$$\begin{aligned} \implies \nabla_\omega \big(y \odot \sigma(-y \odot (X\omega))\big) &= -\begin{pmatrix} \overbrace{\sigma(-y_1 x_1^\top \omega)\big(1 - \sigma(-y_1 x_1^\top \omega)\big)}^{=D_{1,1}}\, x_1 \\ \vdots \\ \underbrace{\sigma(-y_n x_n^\top \omega)\big(1 - \sigma(-y_n x_n^\top \omega)\big)}_{D_{n,n}}\, x_n \end{pmatrix} \\ &= -\begin{bmatrix} D_{1,1} & 0 & \cdots & 0 \\ 0 & D_{2,2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{n,n} \end{bmatrix} \begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,d} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n,1} & x_{n,2} & \cdots & x_{n,d} \end{bmatrix} \\ &= -\begin{bmatrix} D_{1,1}\, x_{1,1} & D_{1,1}\, x_{1,2} & \cdots & D_{1,1}\, x_{1,d} \\ D_{2,2}\, x_{2,1} & D_{2,2}\, x_{2,2} & \cdots & D_{2,2}\, x_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ D_{n,n}\, x_{n,1} & D_{n,n}\, x_{n,2} & \cdots & D_{n,n}\, x_{n,d} \end{bmatrix} = -\begin{bmatrix} D_{1,1}\, x_1^\top \\ D_{2,2}\, x_2^\top \\ \vdots \\ D_{n,n}\, x_n^\top \end{bmatrix} = -DX \end{aligned}$$

where we factored out the x_i in the last step to rewrite it as matrix-vector product. Deriving the entire expression we conclude:

$$\begin{aligned} \nabla^2 f(\omega) &= -\frac{1}{n} X^\top \nabla_\omega \big(y \odot \sigma(-y \odot (X\omega))\big) = \frac{1}{n} X^\top D X \\ D_{ii} &= \sigma(-y_i x_i^\top \omega)\big(1 - \sigma(-y_i x_i^\top \omega)\big) \end{aligned}$$

The hessian of the non-convex regularization term is derived by

$$\begin{aligned} \nabla^2_\omega r(\omega) &= \nabla_\omega \left(2\lambda\alpha \frac{\omega_j}{(1+\alpha\omega_j^2)^2} \right) \\ \frac{\partial^2}{\partial \omega_j^2} r(\omega) &= 2\lambda\alpha \frac{\partial}{\partial \omega_j} \left(\frac{\omega_j}{(1+\alpha\omega_j^2)^2} \right) = 2\lambda\alpha \frac{(1+\alpha\omega_j^2)^2 - 4\alpha\omega_j^2(1+\alpha\omega_j^2)}{(1+\alpha\omega_j^2)^4} = 2\lambda\alpha \frac{1 - 3\alpha\omega_j^2}{(1+\alpha\omega_j^2)^3} \\ \implies \nabla^2 r(\omega) &= \operatorname{diag}\left(2\lambda\alpha \frac{1 - 3\alpha\omega_j^2}{(1+\alpha\omega_j^2)^3} \right)_{j=1,\dots,d} \end{aligned}$$

Combining the steps we derive the Hessian

$$\begin{aligned} \nabla^2 L_2(\omega) &= \nabla^2 f(\omega) + \nabla^2 r(\omega) = \frac{1}{n} X^\top D X + \operatorname{diag}\left(2\lambda\alpha \frac{1 - 3\alpha\omega^2}{(1+\alpha\omega^2)^3} \right) \\ D_{ii} &= \sigma(-y_i x_i^\top \omega)\big(1 - \sigma(-y_i x_i^\top \omega)\big) \end{aligned}$$

2.8 Inexact Newton Method

Given that Newton has cubic complexity we now outline how we aim to reduce the runtime by extending CG and MINRES methods to the Newton-type methods described in our paper. In order for the modified algorithms to inherit the convergence guarantees of the algorithms we want to approximate p s.t.

$$\|Hp + \nabla f\| \leq \epsilon \text{ (absolute tolerance)} < \epsilon = 10^{-8}$$

Since $H_{1,2} = \nabla^2 L_{1,2}$ are clearly symmetric (as both $X^\top DX$ and $\nabla^2 r(x)$ are) we can apply the conjugate gradient method if the H is positive definite or have to fall back on MINRES if it is not pd. Positive definiteness depends on the data matrix and the regularizer curvature. [TODO: runtime for MINRES and CG]

Every iteration of Vanilla Newton takes O(n^3) per iteration because inversion of the Hessian costs O(n^3).

for symmetric applying CG to newton drops the effort for conversion down to

$$O(k \cdot n^2) = O(\sqrt{\kappa} \log{(1/\epsilon)} \cdot n^2)$$

where $\kappa(H) = \frac{\lambda_{max}(H)}{\lambda_{max}(H)}$

Precondition with SSOR to reduce condition number.

References

[1] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, 2nd edition, 2006.

[2] Konstantin Mishchenko. Regularized newton method with global convergence. *SIAM Journal on Optimization*, 33(3):1440–1462, 2023.

[3] Slavomír Hanzely, Dmitry Kamzolov, Dmitry Pasechnyuk, Alexander Gasnikov, Peter Richtárik, and Martin Takác. A damped newton method achieves global $(o)(\frac{1}{k^2})$ and local quadratic convergence rate. *Advances in Neural Information Processing Systems*, 35:25320–25334, 2022.

[4] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, Cambridge, MA, 2012.

2.9 Experiments Setup

All algorithms were implemented in Python 3.13.0 using the NumPy and SciPy libraries. Experiments were conducted on a machine equipped with an Apple M4 Pro processor running the Sequoia 15.5 operating system. Visualizations and plots were generated using a Jupyter Notebook named global-convergence.py.

The source code can be found on github (https://github.com/ben133290/global-convergence-newton).

2.10 PLOTS AND STUFF (please copy and paste to final locaton in report)

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

Table 1: Run time and test accuracy for each algorithm on each dataset and loss type

Dataset	Loss Type	Method	Mean Execution Time (s)	Mean Test Accuracy
a9a	Binary CE Loss	Gradient Descent	0.76568969	0.78
		Classic Newton	failed	failed
		Adaptive Newton	1.93920263	0.84
		Adaptive Newton+	1.886729	0.84
		Globally Convergent Newton	1.22799778	0.85
		Cubic Regularized Newton	1.38458006	0.84
	Non-convex CE Loss	Gradient Descent	0.7895395	0.78
		Classic Newton	1.30171601	0.85
		Adaptive Newton	failed	failed
		Adaptive Newton+	2.03450656	0.82
		Globally Convergent Newton	1.27804756	0.85
		Cubic Regularized Newton	1.48027492	0.84
ijcnn1	Binary CE Loss	Gradient Descent	0.11042674	0.88
		Classic Newton	0.18028998	0.92
		Adaptive Newton	0.27533038	0.92
		Adaptive Newton+	0.31017598	0.92
		Globally Convergent Newton	0.1776003	0.90
		Cubic Regularized Newton	0.21398926	0.90
	Non-convex CE Loss	Gradient Descent	0.11616317	0.90
		Classic Newton	failed	failed
		Adaptive Newton	0.26090709	0.92
		Adaptive Newton+	0.2853574	0.92
		Globally Convergent Newton	0.17406511	0.90
		Cubic Regularized Newton	0.20171062	0.90
covtype	Binary CE Loss	Adaptive Newton	20.22513978	0.75
		Adaptive Newton+	20.77353032	0.75
		Global Regularized Newton	12.83550604	0.74
		Cubic Regularized Newton	14.80531335	0.69
	Non-convex CE Loss	Adaptive Newton	31.30845594	0.75
		Adaptive Newton+	21.32005628	0.75
		Global Regularized Newton	13.47647985	0.74
		Cubic Regularized Newton	14.6580193	0.69

Table 2: Average execution time to reach convergence criterion for different methods. (Gradient Descent failed)

Global Regularized Newton	Adaptive Newton	Adaptive Newton+	Cubic Regularized Newton	Classic Newton
2.26345611	0.35479093	0.25507712	8.79509473	0.11621308

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? **[TODO]**
   (b) Did you describe the limitations of your work? **[TODO]**
   (c) Did you discuss any potential negative societal impacts of your work? **[TODO]**
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? **[TODO]**
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? **[TODO]**
   (b) Did you include complete proofs of all theoretical results? **[TODO]**
3. If you ran experiments...

(a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? **[TODO]**
(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? **[TODO]**
(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? **[TODO]**
(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? **[TODO]**

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? **[TODO]**
   (b) Did you mention the license of the assets? **[TODO]**
   (c) Did you include any new assets either in the supplemental material or as a URL? **[TODO]**
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? **[TODO]**
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? **[TODO]**
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? **[TODO]**
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? **[TODO]**
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? **[TODO]**

A Appendix

Optionally include extra information (complete proofs, additional experiments and plots) in the appendix. This section will often be part of the supplemental material.