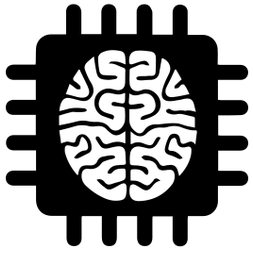
Improving L-BFGS Initialization for Trust-Region Methods in Deep Learning

Jacob Rafati
http://rafati.net
jrafatiheravi@ucmerced.edu

Ph.D. Candidate, Electrical Engineering and Computer Science
University of California, Merced



Agenda

- Introduction, Problem Statement and Motivations
- Overview on Quasi-Newton Optimization Methods
- L-BFGS Trust Region Optimization Method
- Proposed Methods for Initialization of L-BFGS
- Application in Deep Learning (Image Classification Task)

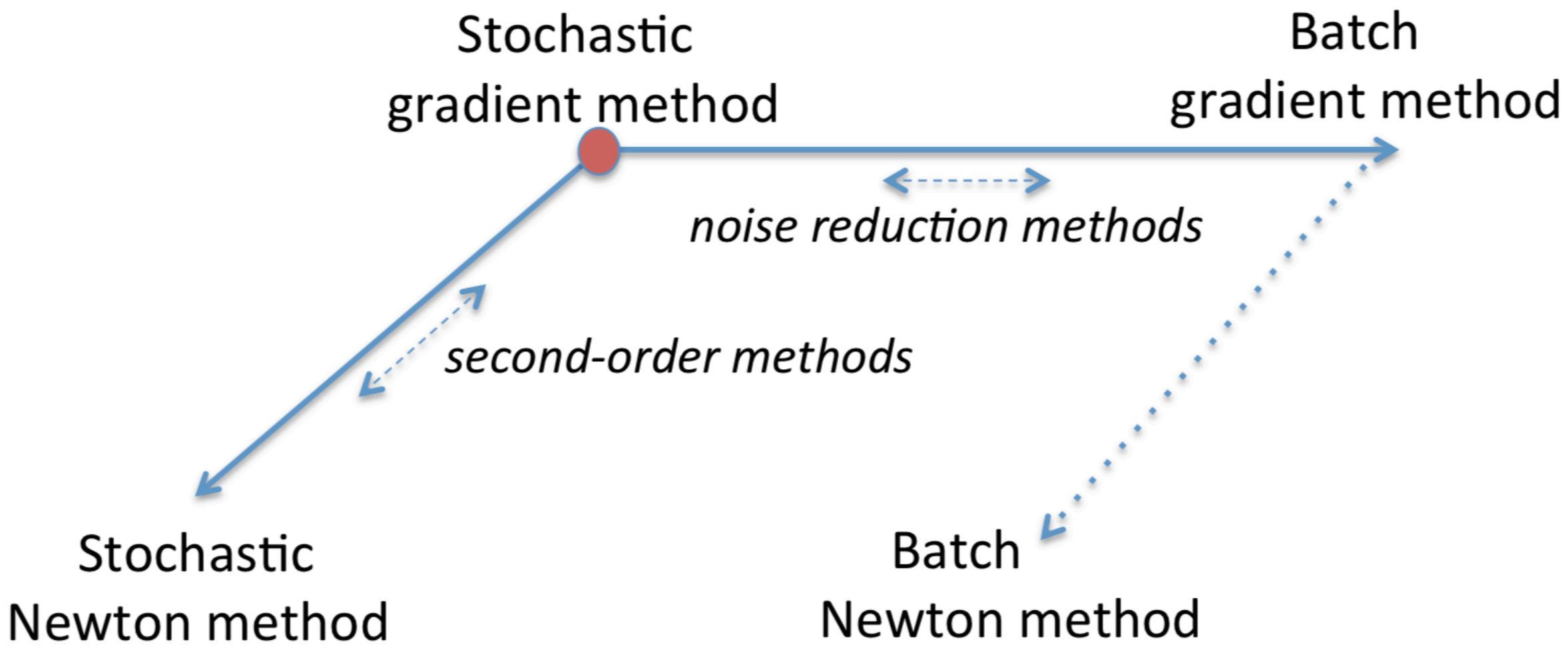
Introduction, Problem Statement and Motivations

Unconstrained Optimization Problem

$$\min_{w \in \mathbb{R}^n} \mathcal{L}(w) \triangleq \frac{1}{N} \sum_{i=1}^{N} \ell_i(w)$$

$$\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$$

Optimization Algorithms



Bottou et al., (2016) Optimization methods for large-scale machine learning, print arXiv:1606.04838

Optimization Algorithms

1. Start from a random point w_0
2. Repeat each iteration, $k = 0, 1, 2, \ldots,$
    3. Choose a search direction p_k
    4. Choose a step size α_k
    5. Update parameters $w_{k+1} \leftarrow w_k + \alpha_k p_k$
6. Until $\|\nabla \mathcal{L}\| < \epsilon$

Properties of Objective Function

$$\min_{w \in \mathbb{R}^n} \mathcal{L}(w) \triangleq \frac{1}{N} \sum_{i=1}^{N} \ell_i(w)$$

- n and N are both large in modern applications.
- $\mathcal{L}(w)$ is a non-convex and nonlinear function.
- $\nabla^2 \mathcal{L}(w)$ is ill-conditioned.
- Computing full gradient, $\nabla \mathcal{L}$ is expensive.
- Computing Hessian, $\nabla^2 \mathcal{L}$ is not practical.

Stochastic Gradient Decent

1. Sample indices $\mathcal{S}_k \subset \{1, 2, \ldots, N\}$

2. Compute *stochastic* (subsampled) gradient

$$\nabla \mathcal{L}(w_k) \approx \nabla \mathcal{L}(w_k)^{(\mathcal{S}_k)} \triangleq \frac{1}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla \ell_i(w_k)$$

3. Assign a *learning rate* α_k

4. Update parameters using $p_k = -\nabla \mathcal{L}(w_k)^{(\mathcal{S}_k)}$

$$w_{k+1} \leftarrow w_k - \alpha_k \nabla \mathcal{L}(w_k)^{(\mathcal{S}_k)}$$

H. Robbins, D. Siegmund. (1971). ”A convergence theorem for non negative almost supermartingales and some applications”. Optimizing methods in statistics.

Advantages of SGD

- SGD algorithms are very easy to implement.
- SGD requires only computing the gradient.
- SGD has a low cost-per-iteration.

Bottou et al., (2016) Optimization methods for large-scale machine learning, print arXiv:1606.04838

Disadvantages of SGD

- Very sensitive to the ill-conditioning problem and scaling.
- Requires fine-tuning many hyper-parameters.
- Unlikely exhibit acceptable performance on first try.
- requires many trials and errors.
- Can stuck in a saddle-point instead of local minimum.
- Sublinear and slow rate of convergence.

Bottou et al., (2016). Optimization methods for large-scale machine learning. print arXiv:1606.04838
J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.

Second-Order Methods

1. Sample indices $\mathcal{S}_k \subset \{1, 2, \ldots, N\}$

2. Compute *stochastic* (subsampled) gradient

$$\nabla \mathcal{L}(w_k) \approx \nabla \mathcal{L}(w_k)^{(\mathcal{S}_k)} \triangleq \frac{1}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla \ell_i(w_k)$$

3. Compute Hessian

$$\nabla^2 \mathcal{L}(w_k) \approx \nabla^2 \mathcal{L}(w_k)^{(\mathcal{S}_k)} \triangleq \frac{1}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla^2 \ell_i(w_k)$$

Second-Order Methods

4. Compute Newton's direction

$$p_k = -\nabla^2 \mathcal{L}(w_k)^{-1} \nabla \mathcal{L}(w_k)$$

5. Find proper step length

$$\alpha_k = \min_{\alpha} \mathcal{L}(w_k + \alpha p_k)$$

6. Update parameters

$$w_{k+1} \leftarrow w_k + \alpha_k p_k$$

Second-Order Methods Advantages

- The rate of convergence is super-linear (quadratic for Newton method).
- They are resilient to problem ill-conditioning.
- They involve less parameter tuning.
- They are less sensitive to the choice of hyper-parameters.

J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.

Second-Order Methods Disadvantages

- Computing the Hessian matrix is very expensive and requires massive storage.
- Computing the inverse of Hessian is not practical.

J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.
Bottou et al., (2016) Optimization methods for large-scale machine learning, print arXiv:1606.04838

Quasi-Newton Methods

1. Construct a **low-rank** approximation of Hessian

$$B_k \approx \nabla^2 \mathcal{L}(w_k)$$

2. Find the search direction by Minimizing the **Quadratic Model** of the objective function

$$p_k = \underset{p \in \mathbb{R}^n}{\operatorname{argmin}} \ \mathcal{Q}_k(p) \triangleq g_k^T p + \frac{1}{2} p^T B_k p^T$$

Quasi-Newton Matrices

- Symmetric
- Easy and Fast Computation
- Satisfies Secant Condition

$$B_{k+1} s_k = y_k$$

$$s_k \triangleq w_{k+1} - w_k$$

$$y_k \triangleq \nabla \mathcal{L}(w_{k+1}) - \nabla \mathcal{L}(w_k)$$

Broyden **F**letcher **G**oldfarb **S**hanno.

J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.

Broyden **F**letcher **G**oldfarb **S**hanno.

$$B_{k+1} = B_k - \frac{1}{s_k^T B_k s_k} B_k s_k s_k^T B_k + \frac{1}{y_k^T s_k} y_k y_k^T,$$

$$s_k \triangleq w_{k+1} - w_k$$

$$y_k \triangleq \nabla \mathcal{L}(w_{k+1}) - \nabla \mathcal{L}(w_k)$$

$$B_0 = \gamma_k I$$

J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.

Quasi-Newton Methods Advantages

- The rate of convergence is super-linear.
- They are resilient to problem ill-conditioning.
- The second derivative is not required.
- They only use the gradient information to construct quasi-Newton matrices.

J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.

Quasi-Newton Methods disadvantages

- The cost of storing the gradient informations can be expensive.
- The quasi-Newton matrix can be dense.
- The quasi-Newton matrix grow in size and rank in large-scale problems.

J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.

Limited-Memory BFGS

Limited Memory Storage

$$S_k = \begin{bmatrix} s_{k-m} & \cdots & s_{k-1} \end{bmatrix} \qquad Y_k = \begin{bmatrix} y_{k-m} & \cdots & y_{k-1} \end{bmatrix}$$

L-BFGS Compact Representation

$$B_k = B_0 + \Psi_k M_k \Psi_k^T$$

$$B_0 = \gamma_k I$$

where

$$\Psi_k = \begin{bmatrix} B_0 S_k & Y_k \end{bmatrix}, \quad M_k = \begin{bmatrix} -S_k^T B_0 S_k & -L_k \\ -L_k^T & D_k \end{bmatrix}^{-1}$$

$$S_k^T Y_k = L_k + D_k + U_k$$

Limited-Memory Quasi-Newton Methods

- Low rank approximation
- Small memory of recent gradients.
- Low cost of computation of search direction.
- Linear or superlinear Convergence rate can be achieved.

J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.

Objectives

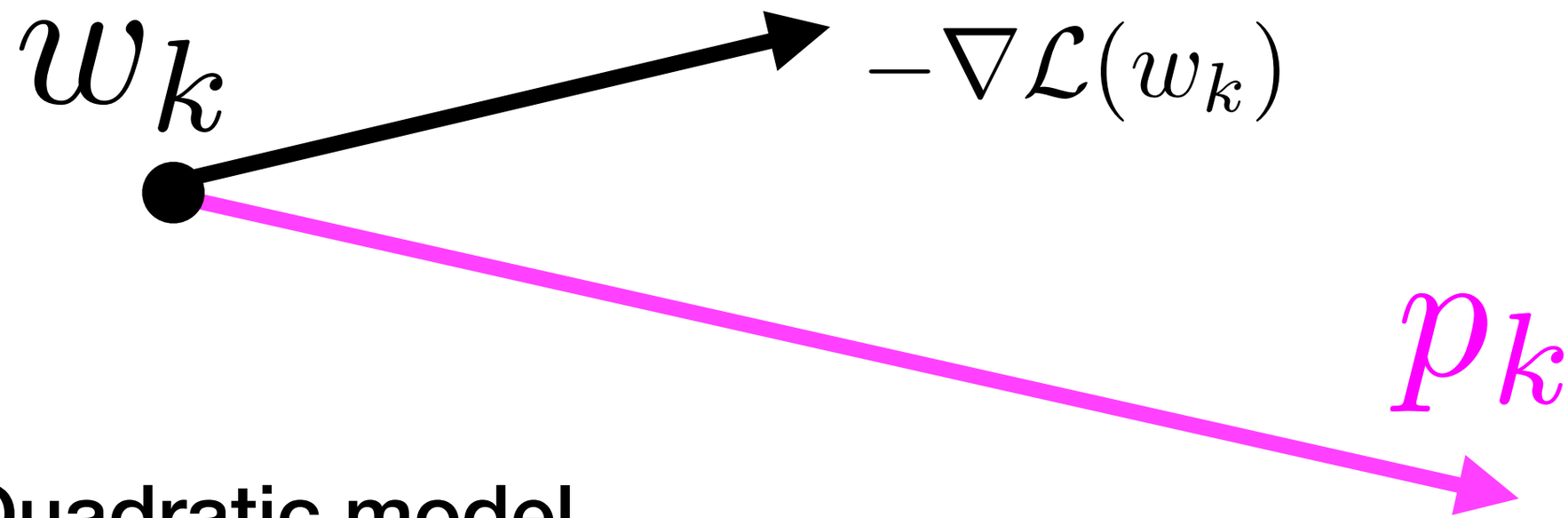
$$B_0 = \gamma_k I$$

$$B_k = B_0 + \Psi_k M_k \Psi_k^T$$

What is the best choice for initialization?

Overview on Quasi-Newton Optimization Strategies

Line Search Method



Quadratic model

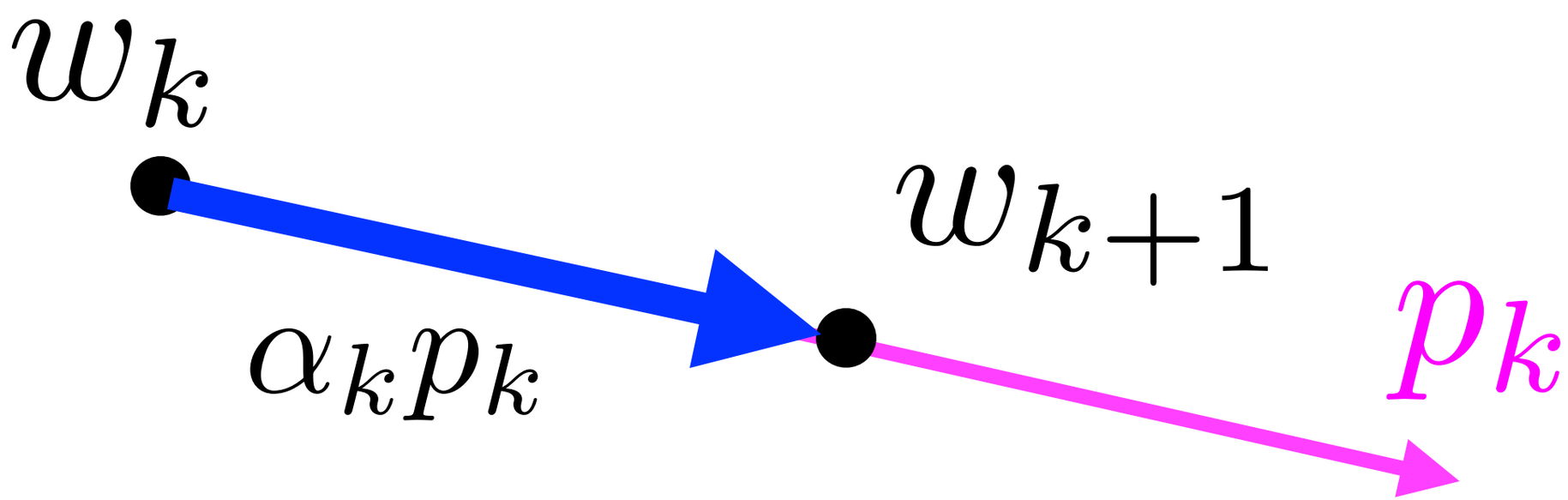
$$p_k = \underset{p \in \mathbb{R}^n}{\operatorname{argmin}} \; \mathcal{Q}_k(p) \triangleq g_k^T p + \frac{1}{2} p^T B_k p^T$$

if B_k is positive definite:

$$p_k = B_k^{-1} g_k$$

J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.

Line Search Method



Wolfe conditions

$$\mathcal{L}(w_k + \alpha_k p_k) \leq \mathcal{L}(w_k) + c_1 \alpha_k \nabla \mathcal{L}(w_k)^T p_k$$

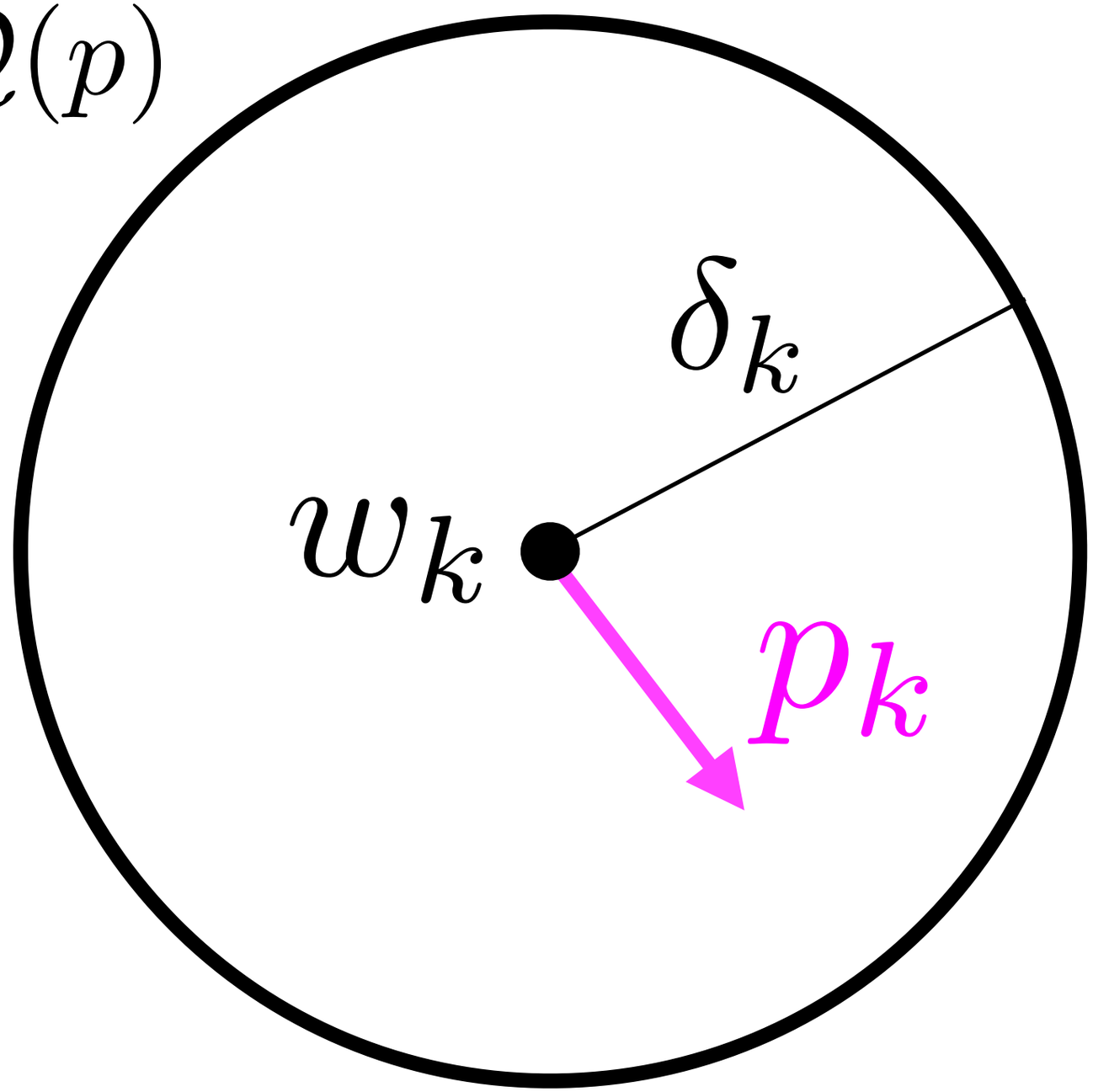
$$\nabla \mathcal{L}(w_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f(w_k)^T p_k$$

J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.

Trust Region Method

$$p_k = \arg\min_{p \in \mathbb{R}^n} \mathcal{Q}(p)$$

$$\text{s.t. } \|p\|_2 \leq \delta_k$$



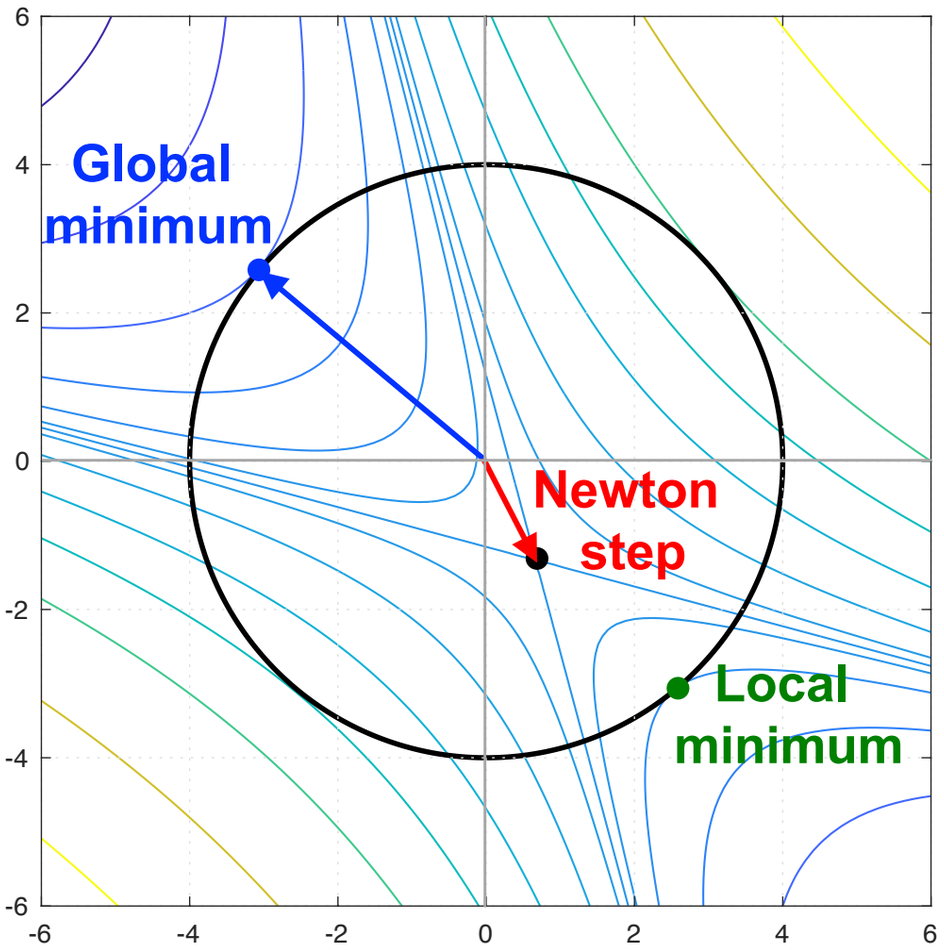
Toint et al. (2000), Trust Region Methods, SIAM.

Trust Region Method

Theorem. Let δ be a positive constant. A vector p^* is a global solution of the trust-region subproblem if and only if $\|p^*\|_2 \leq \delta$ and there exists a unique $\sigma^* \geq 0$ such that $B + \sigma^* I$ is positive semidefinite and

$$(B + \sigma^* I)p^* = -g \quad \text{and} \quad \sigma^*(\delta - \|p^*\|_2) = 0.$$

Moreover, if $B + \sigma^* I$ is positive definite, then the global minimizer is unique.



J. J. Mor´e and D. C. Sorensen, (1984) Newton's method, in Studies in Mathematics, Volume 24. Studies in Numerical Analysis, Math. Assoc., pp. 29–82.

L-BFGS Trust Region Optimization Method

L-BFGS in Trust Region

$$B_k = B_0 + \Psi_k M_k \Psi_k^T$$

Eigen-decomposition

$$B_k = P \begin{bmatrix} \Lambda + \gamma_k I & 0 \\ 0 & \gamma_k I \end{bmatrix} P^T$$

Sherman-Morrison-Woodbury Formula

$$p_k^* = -\frac{1}{\tau^*} \left[I - \Psi_k (\tau^* M_k^{-1} + \Psi_k^T \Psi_k)^{-1} \Psi_k^T \right] g_k,$$

$$\tau^* = \gamma_k + \sigma^*$$

L. Adhikari et al. (2017) “Limited-memory trust-region methods for sparse relaxation,” in Proc.SPIE, vol. 10394.
Brust et al, (2017). “On solving L-SR1 trust-region subproblems,” Computational Optimization and Applications, vol. 66, pp. 245–266.

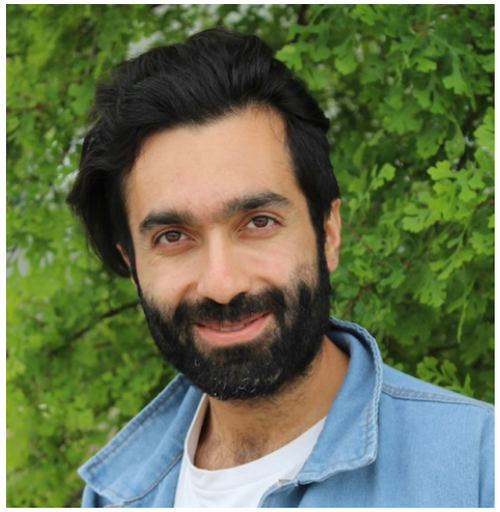
L-BFGS in Trust Region vs. Line-Search

Trust-Region Minimization Algorithm for Training Responses (TRMinATR): The Rise of Machine Learning Techniques

Jacob Rafati
Electrical Engineering and Computer Science
University of California, Merced
Merced, CA 95343 USA

Omar DeGuchy
Applied Mathematics
University of California, Merced
Merced, CA 95343 USA

Roummel F. Marcia
Applied Mathematics
University of California, Merced
Merced, CA 95343 USA



26th European Signal Processing Conference, Rome, Italy. September 2018.

Proposed Methods for Initialization of L-BFGS

Initialization Method I

$$B_0 = \gamma_k I$$

$$B_k = B_0 + \Psi_k M_k \Psi_k^T$$

Spectral estimate of Hessian

$$\gamma_k = \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}}$$

$$\gamma_k = \arg\min_{\gamma} \|B_0^{-1} y_{k-1} - s_{k-1}\|_2^2, \quad B_0 = \gamma I$$

J. Nocedal and S. J. Wright. (2006). Numerical Optimization. 2nd ed. New York. Springer.

Initialization Method II

Consider a quadratic function

$$\mathcal{L}(w) = \frac{1}{2} w^T H w + g^T w \qquad \nabla^2 \mathcal{L}(w) = H$$

We have

$$H S_k = Y_k$$

Therefore

$$S_k^T H S_k = S_k^T Y_k$$

Erway et al. (2018). “Trust-Region Algorithms for Training Responses: Machine Learning Methods Using Indefinite Hessian Approximations,” ArXiv e-prints.

Initialization Method II

Since

$$B_k = B_0 + \Psi_k M_k \Psi_k^T \qquad B_0 = \gamma_k I$$

$$\Psi_k = \begin{bmatrix} B_0 S_k & Y_k \end{bmatrix}, \quad M_k = \begin{bmatrix} -S_k^T B_0 S_k & -L_k \\ -L_k^T & D_k \end{bmatrix}^{-1}$$

Secant Condition

$$B_k S_k^T = Y_k$$

We have

$$S_k^T H S_k - \gamma_k S_k^T S_k = S_k^T \Psi_k M_k \Psi_k^T S_k$$

Initialization Method II

$$S_k^T H S_k - \gamma_k S_k^T S_k = S_k^T \Psi_k M_k \Psi_k^T S_k$$

General Eigen-Value Problem

$$(L_k + D_k + L_k^T) z = \lambda S_k^T S_k z$$

Upper bound for initial value to avoid false curvature information

$$\gamma_k \in (0, \lambda_{\min})$$

Initialization Method III

$$B_0 = \gamma_k I \qquad\qquad B_k = B_0 + \Psi_k M_k \Psi_k^T$$

$$S_k^T H S_k - \gamma_k S_k^T S_k = S_k^T \Psi_k M_k \Psi_k^T S_k$$

Note that compact representation matrices contains γ_k

$$\Psi_k = \begin{bmatrix} B_0 S_k & Y_k \end{bmatrix}, \quad M_k = \begin{bmatrix} -S_k^T B_0 S_k & -L_k \\ -L_k^T & D_k \end{bmatrix}^{-1}$$

Initialization Method III

General Eigen-Value Problem

$$A^* z = \lambda B^* z$$

Upper bound for initial values to avoid false curvature information

$$\gamma_k \in (0, \lambda_{\min})$$

$$A^* = L_k + D_k + L_k^T - S_k^T Y_k \tilde{D} Y_k^T S_k - \gamma_{k-1}^2 (S_k^T S_k \tilde{A} S_k^T S_k),$$
$$B^* = S_k^T S_k + S_k^T S_k \tilde{B} Y_k^T S_k^T + S_k^T Y_k \tilde{B}^T S_k^T S_k.$$

Applications in Deep Learning

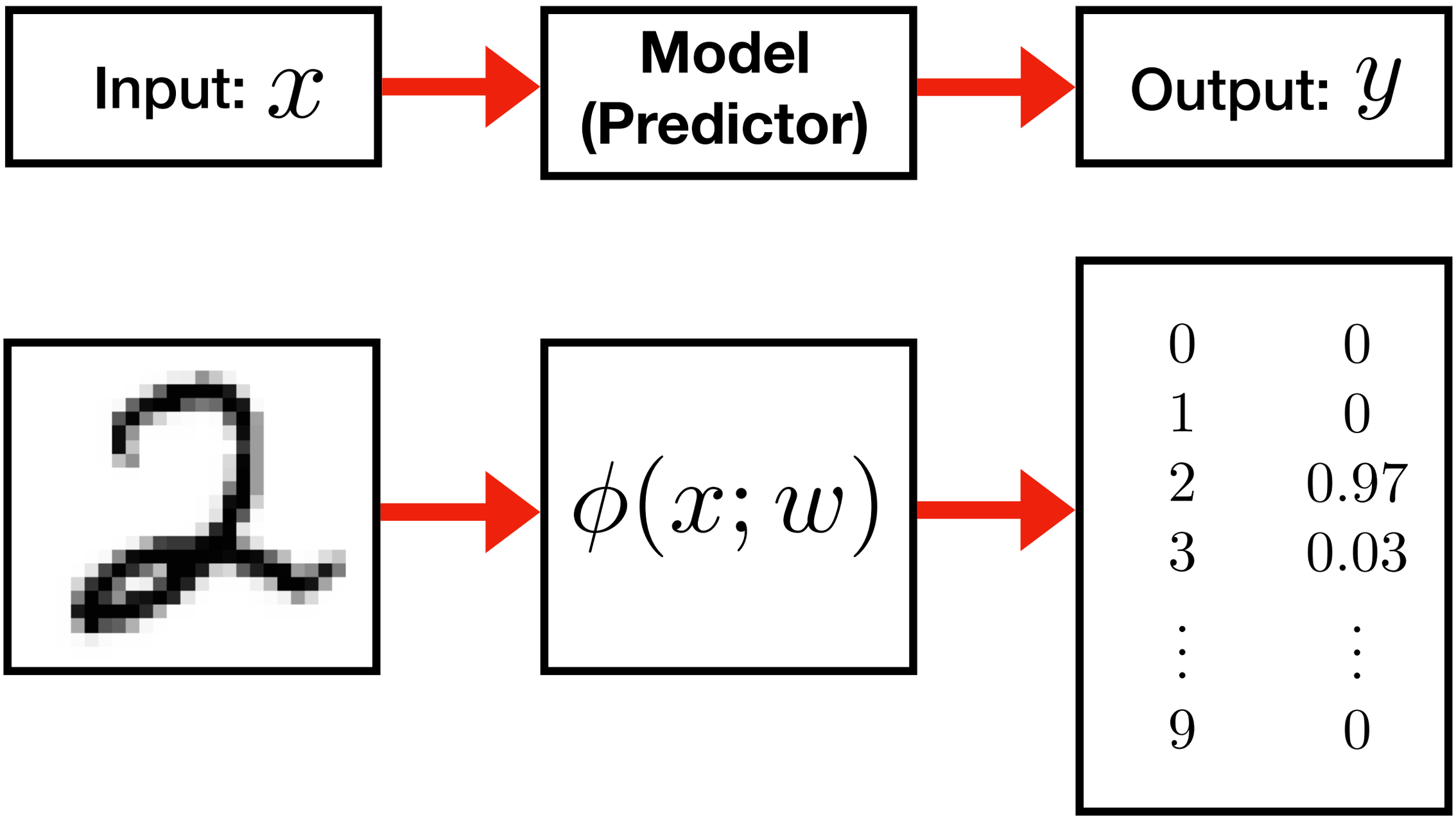
Supervised Learning

Features $X = \{x_1, x_2, \ldots, x_i, \ldots, x_N\}$

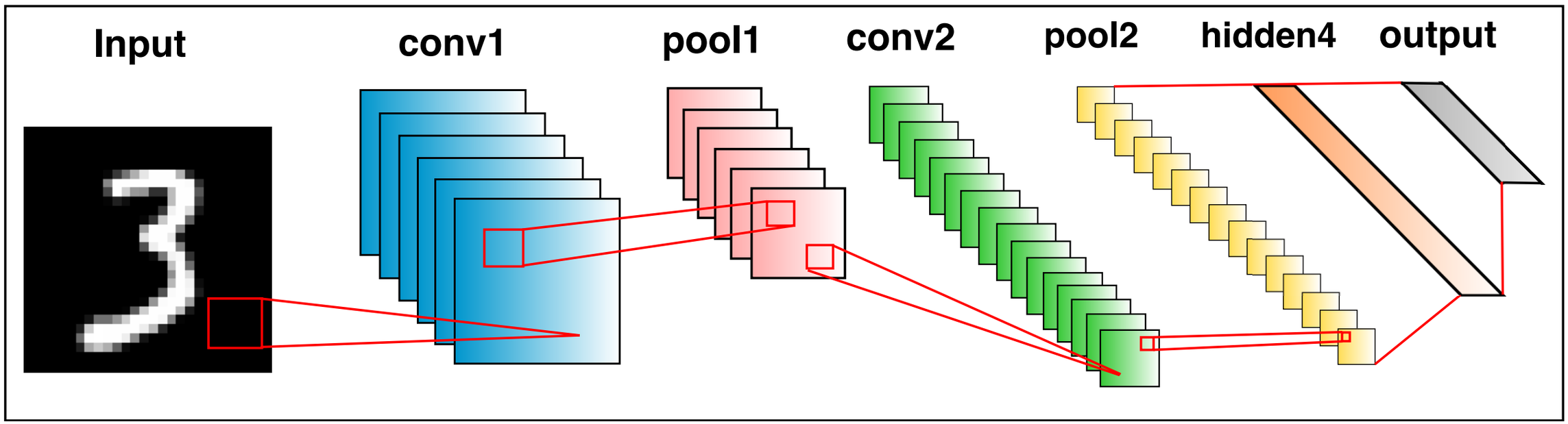
Labels $T = \{t_1, t_2, \ldots, t_i, \ldots, t_N\}$

$$\Phi : X \rightarrow T$$

Supervised Learning



Convolutional Neural Network



$$\phi(x; w)$$

$$n = 413,080$$

Lecun et al. (1998) “Gradient-basedlearning applied to document recognition,” Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324.

Loss Function

Target $t = \begin{bmatrix} 0, & 0, & 1, & 0, & \dots, & 0 \end{bmatrix}$

Output $y = \begin{bmatrix} 0, & 0, & 0.97, & 0.03, & \dots, & 0 \end{bmatrix}$

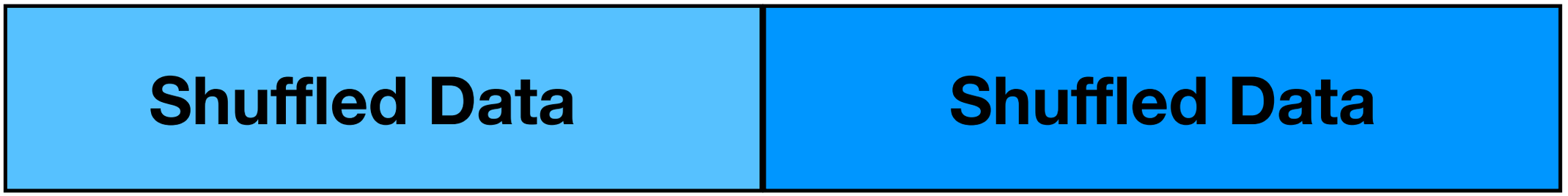
Cross-entropy

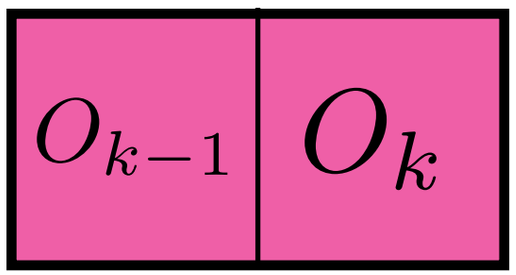
$$\ell(t, y) = -t.\log(y) - (1 - t).\log(1 - y)$$

Empirical Risk

$$\mathcal{L}(w) = \frac{1}{N} \sum_{i=1}^{N} \ell(t_i, \phi(x_i, w))$$

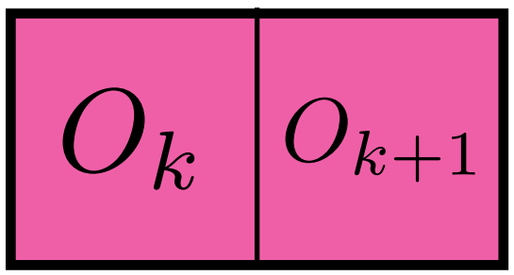
Multi-Batch L-BFGS



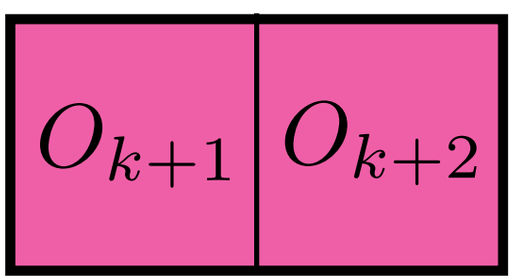


$\mathcal{S}_k$

$O_k = \mathcal{S}_k \cap \mathcal{S}_{k+1}$



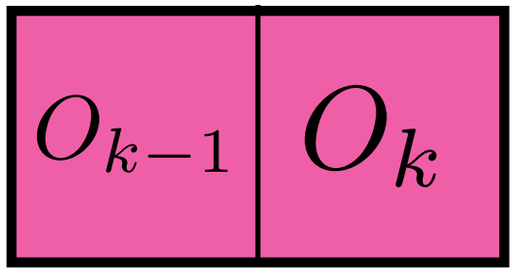
$\mathcal{S}_{k+1}$



$\mathcal{S}_{k+2}$

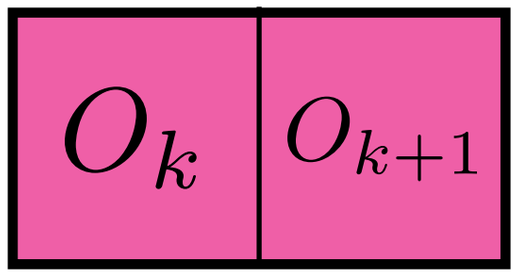
Berahas et al. (2016). “A multi-batch L-BFGS method for machine learning,” in Advances in Neural Information Processing Systems 29, pp. 1055–1063.

Computing gradients



$\mathcal{S}_k$

$O_k = \mathcal{S}_k \cap \mathcal{S}_{k+1}$



$\mathcal{S}_{k+1}$

$$g_k = \nabla \mathcal{L}(w_k)^{(\mathcal{S}_k)} = \frac{1}{|\mathcal{S}_k|} \sum_{i \in J_k} \nabla \mathcal{L}_i(w_k)$$

$$y_k = \nabla \mathcal{L}(w_{k+1})^{(O_k)} - \nabla \mathcal{L}(w_k)^{(O_k)}$$

Berahas et al. (2016). “A multi-batch L-BFGS method for machine learning,” in Advances in Neural Information Processing Systems 29, pp. 1055–1063.

Experiment

Initialization	Source	Formula
Method I	Solve the optimization problem: $\gamma_k = \arg\min_\gamma \|B_0^{-1} y_{k-1} - s_{k-1}\|_2^2$	$\gamma_k = \max\left\{1, \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}}\right\}$
Method II	Solve the generalized eigenvalue problem: $(L_k + D_k + L_k^T)z = \lambda S_k^T S_k z$	$\gamma_k = \begin{cases} \max\{1, 0.9\lambda_{\min}\} & \text{if } \lambda_{\min} > 0, \\ \text{Use Method I} & \text{if } \lambda_{\min} \leq 0. \end{cases}$
Method III	Solve the generalized eigenvalue problem: $A^* z = B^* \lambda z$	$\gamma_k = \begin{cases} \max\{1, 0.9\lambda_{\min}\} & \text{if } \lambda_{\min} > 0, \\ \text{Use Method I} & \text{if } \lambda_{\min} \leq 0. \end{cases}$

Trust-Region Algorithm

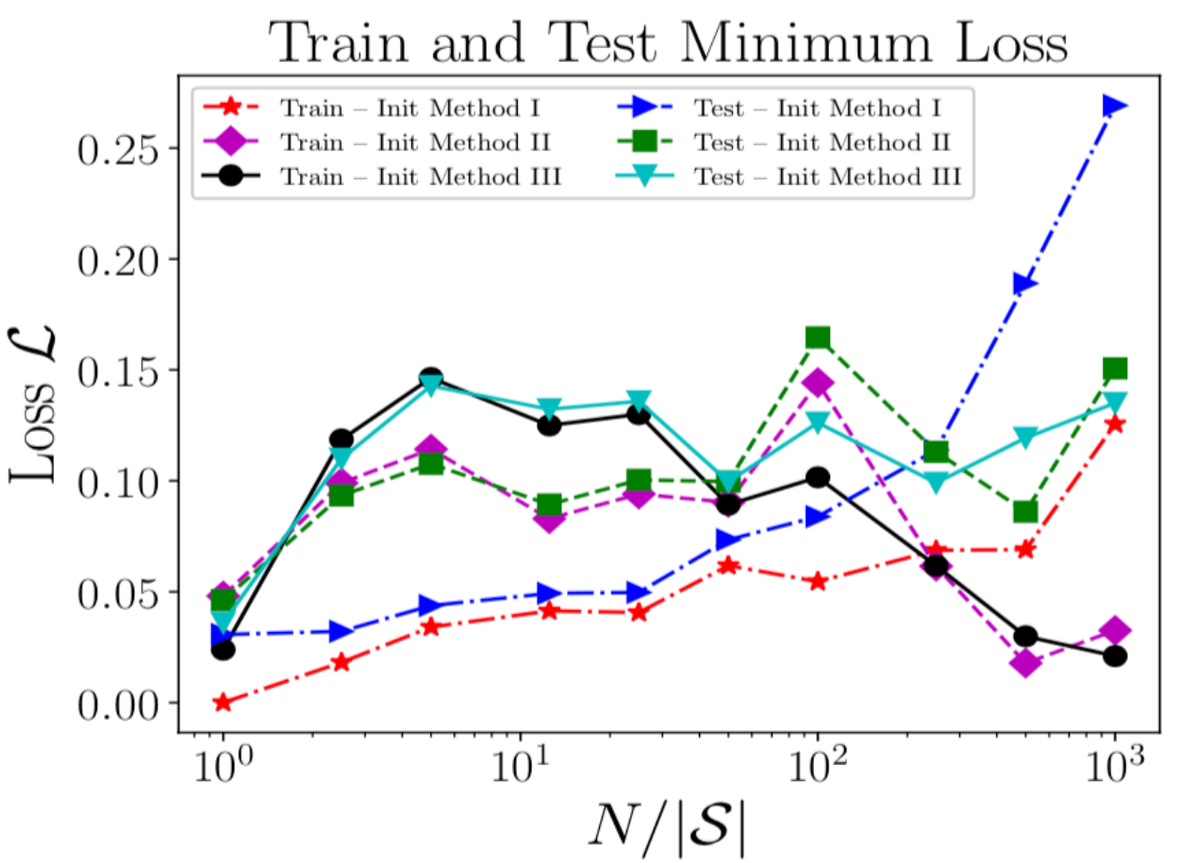
Algorithm Limited-memory BFGS trust-region method.

Input: starting point w_0, tolerance $\epsilon > 0$, δ_0, $\eta < \frac{1}{4}$
Choose initialization **Method I**, **Method II**, or **Method III**

for $k = 0, 1, 2, \ldots$ **do**
  Compute $g_k = \nabla \mathcal{L}(w_k)$
  Compute γ_k to initialize $B_0 = \gamma_k I$
  $\gamma_k \leftarrow \max\{\gamma_k, 1\}$
  Compute Ψ_k and M_k
  Form B_k orthonormal matrices
  Compute search step p_k by solving TR subproblem
  Compute $s_k = p_k$ and $y_k = \nabla \mathcal{L}(w_k + p_k) - \nabla \mathcal{L}(w_k)$
  if $s_k^T y_k > 0$ **then**
    Store $\{s_k, y_k\}$ in storage S_{k+1} and Y_{k+1}
    Discard $\{s_{k-m}, y_{k-m}\}$ from storage if $k > m$
  end if
  $\rho_k \leftarrow (\mathcal{L}(w_k) - \mathcal{L}(w_k + p_k))/(\mathbf{Q}_k(0) - \mathbf{Q}_k(p_k))$
  if $\rho_k > \eta$ **then**
    $w_{k+1} = w_k + p_k$
  else
    $w_{k+1} = w_k$
  end if
  Update trust-region radius δ_{k+1}
  if $\|g\|_2 < \epsilon$ or k reached to maximum episodes **then**
    break
  end if
end for

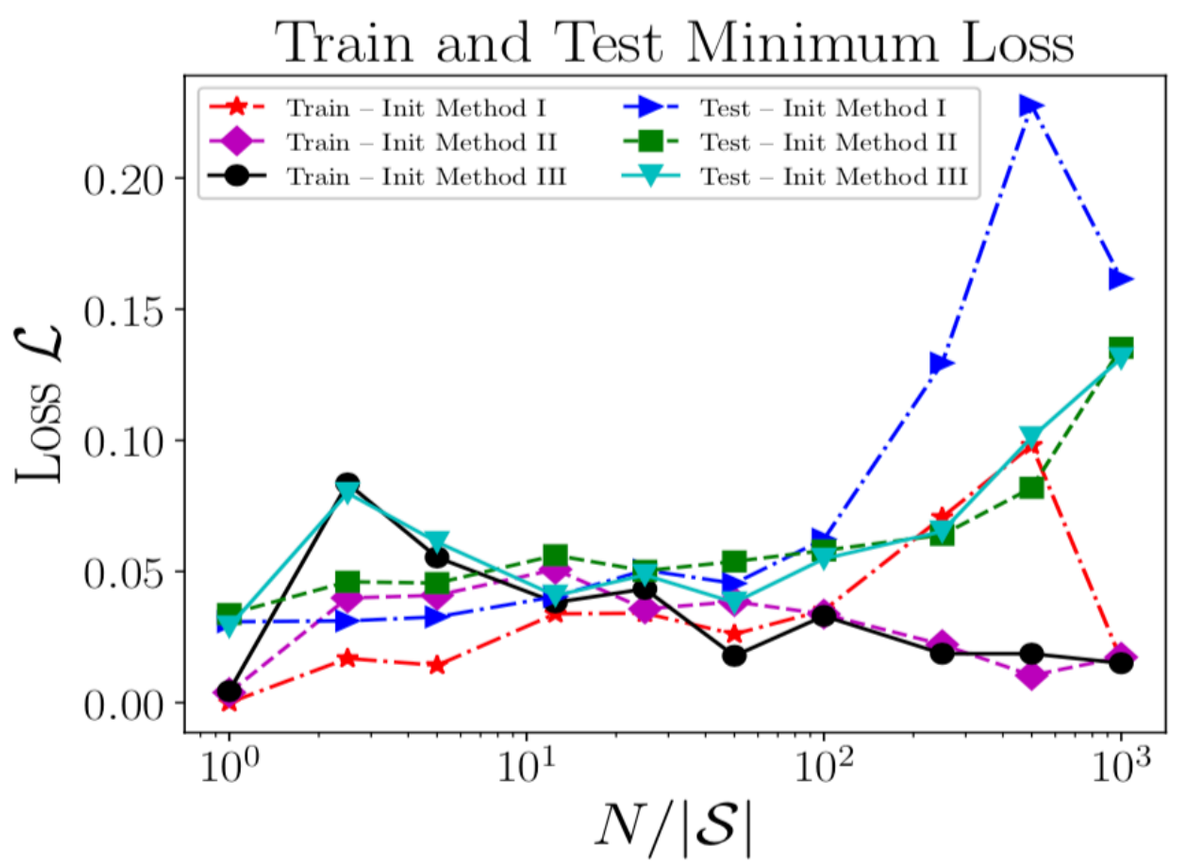
Results - Loss

$m = 10$



Full ⟷ **Stochastic**

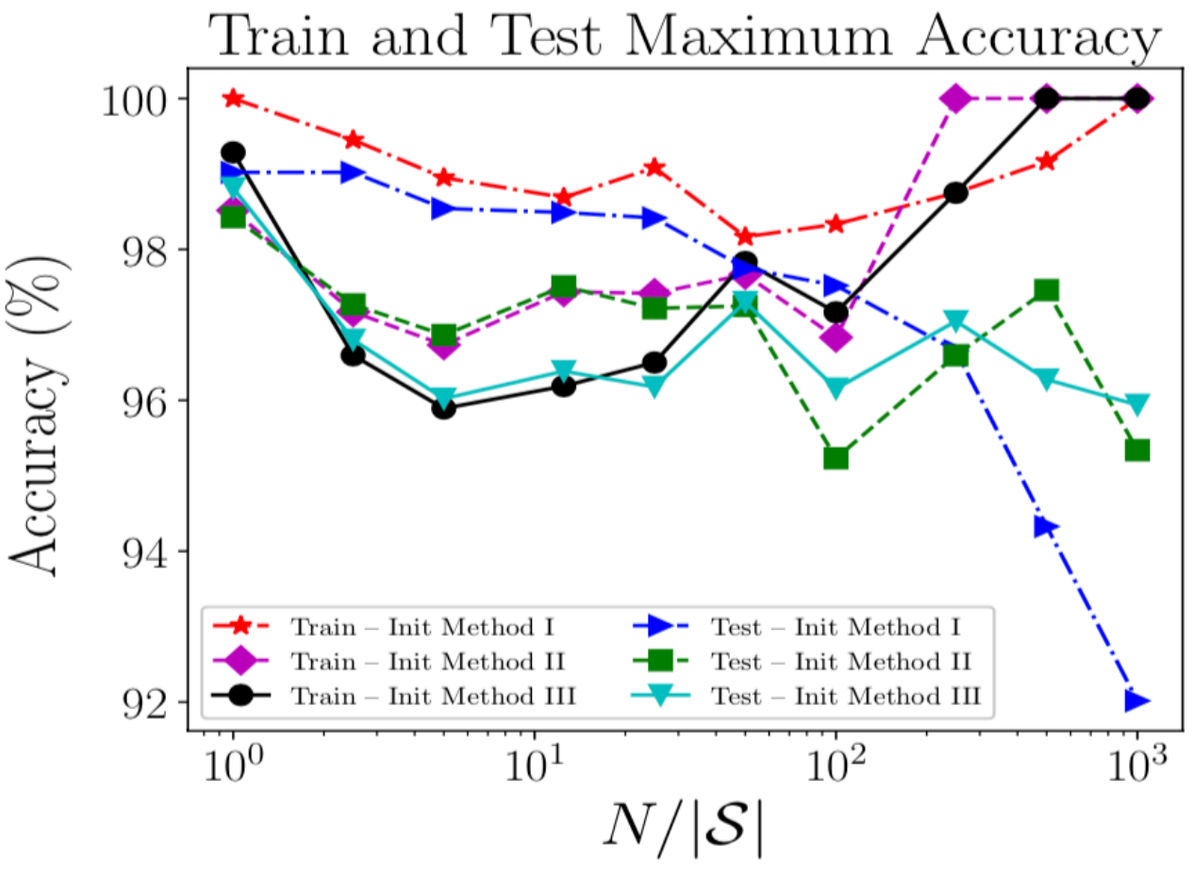
$m = 20$



Full ⟷ **Stochastic**

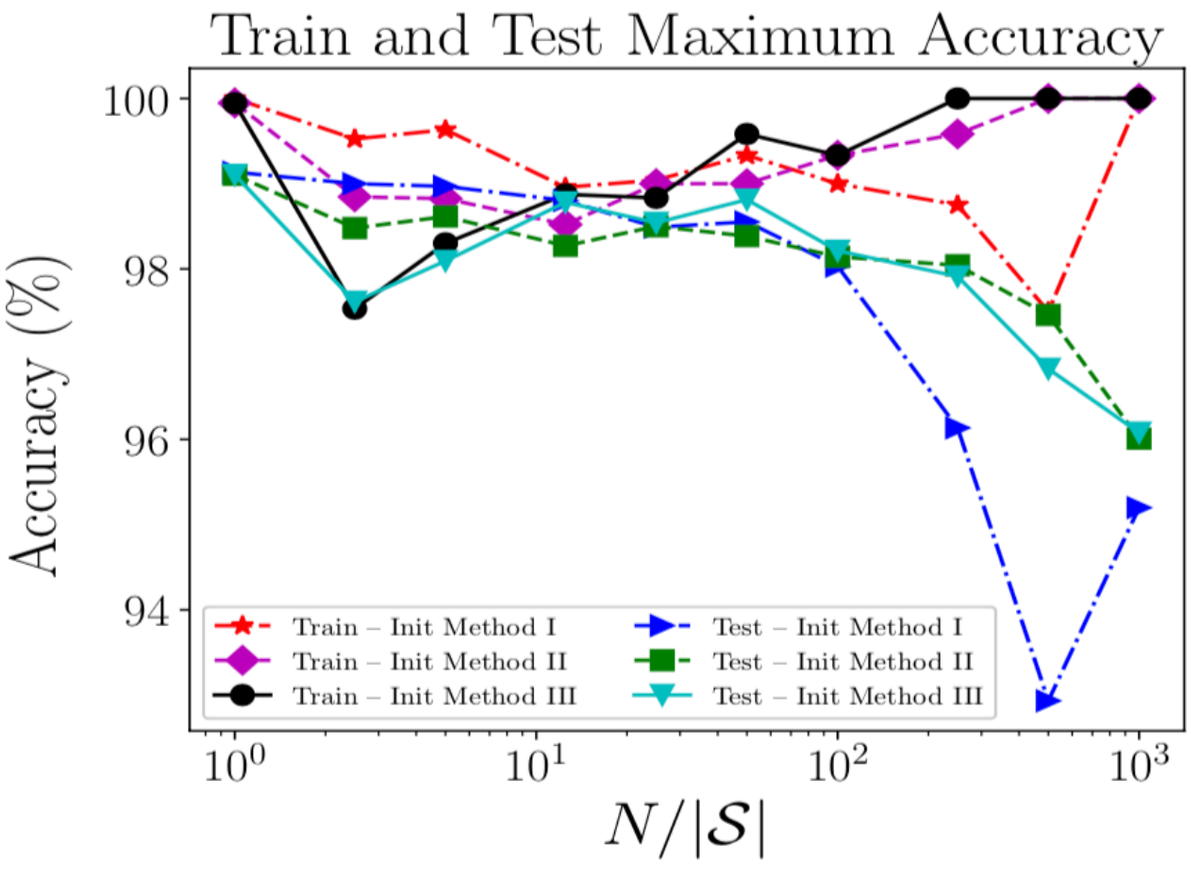
Results - Accuracy

$m = 10$



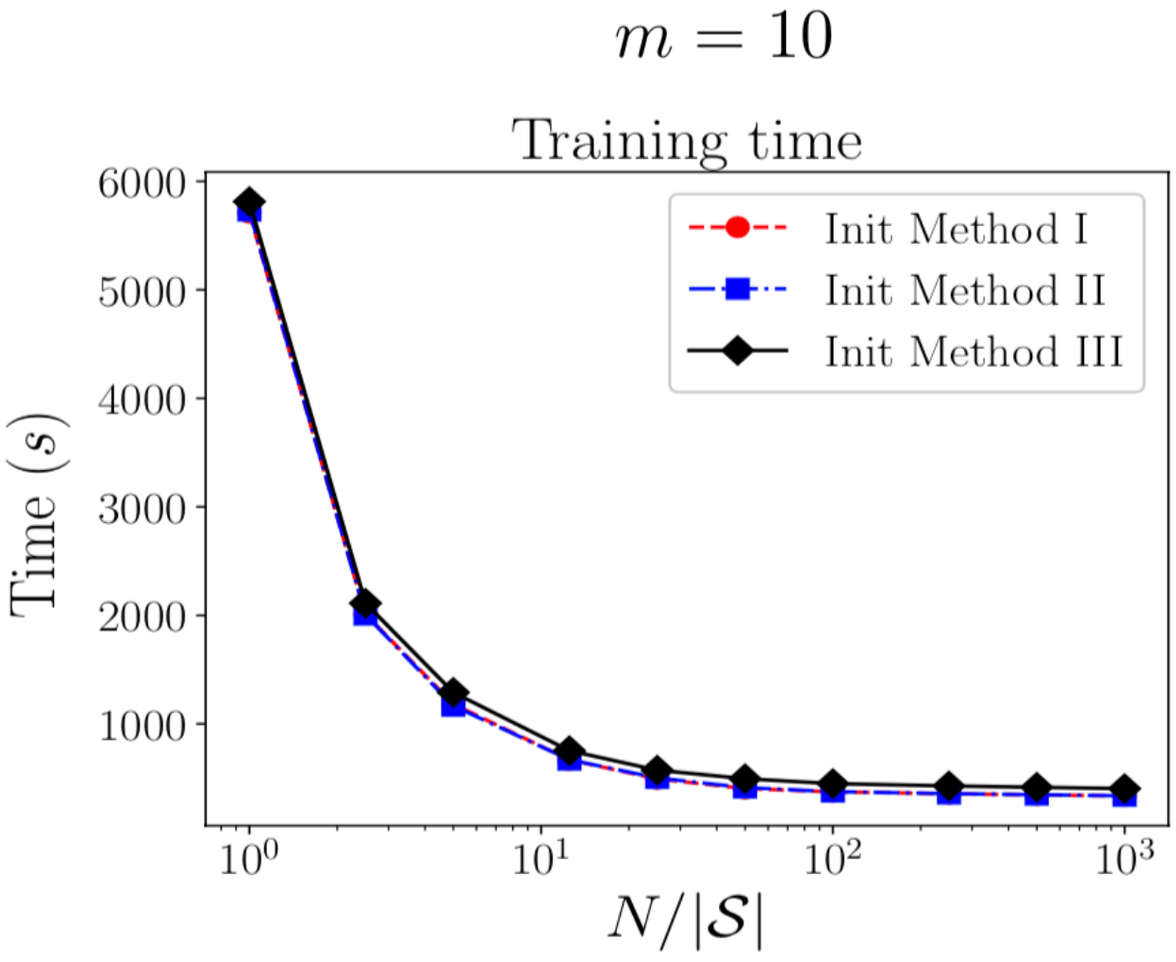
Full ⟷ **Stochastic**

$m = 20$

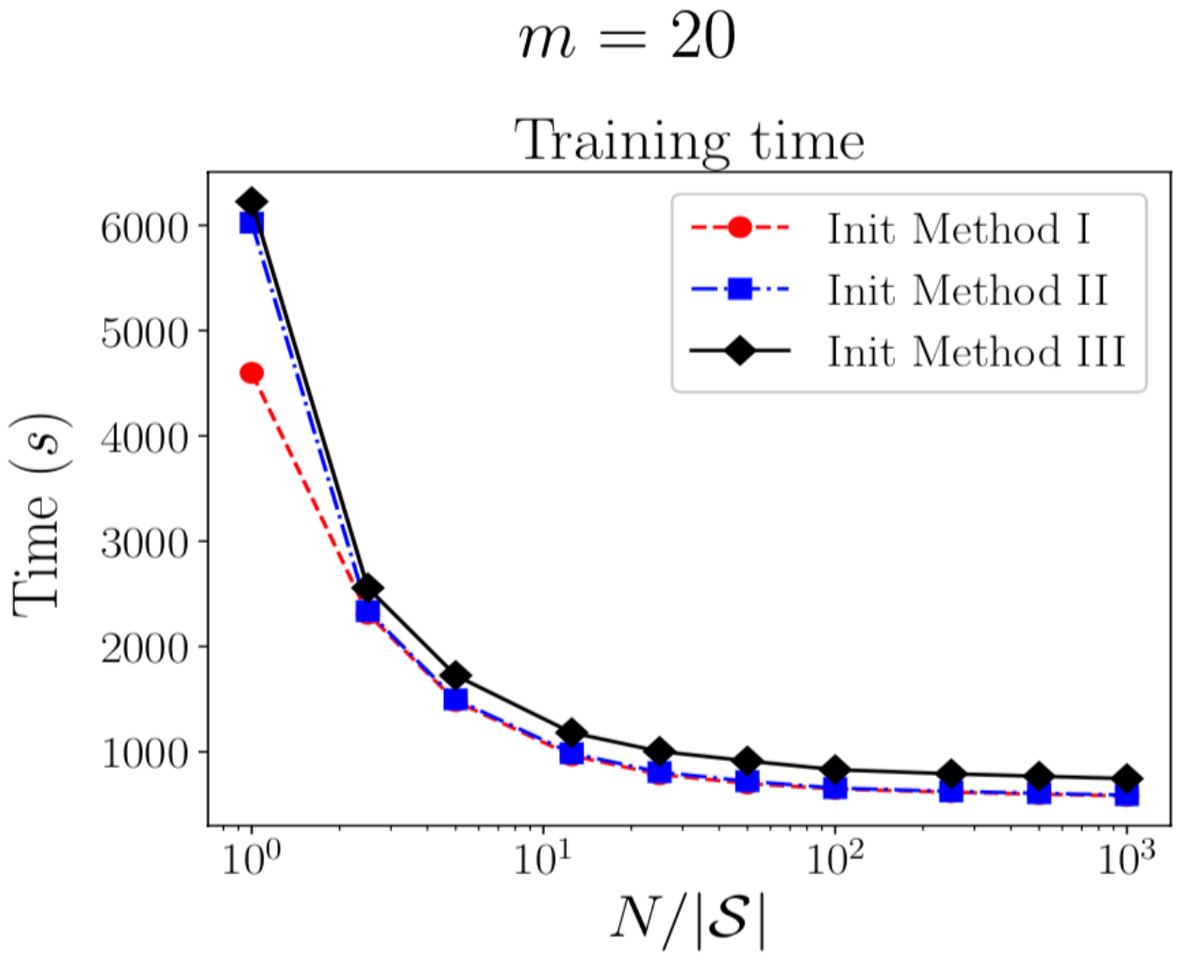


Full ⟷ **Stochastic**

Results - Training Time



Full ⟷ **Stochastic**

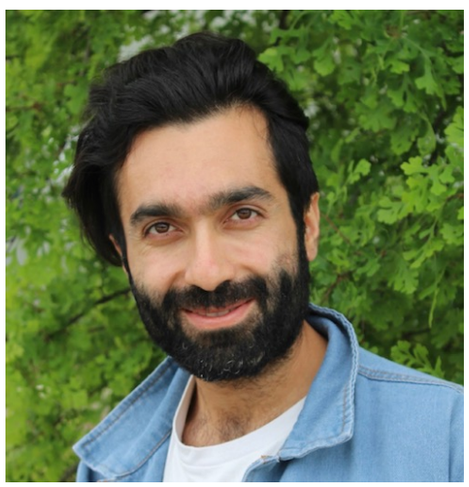


Full ⟷ **Stochastic**

Acknowledgement

Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning

Jacob Rafati
Electrical Engineering and Computer Science
University of California, Merced
Merced, CA 95340 USA
jrafatiheravi@ucmerced.edu



Roummel F. Marcia
Applied Mathematics
University of California, Merced
Merced, CA 95340 USA
rmarcia@ucmerced.edu

This research is supported by NSF Grants CMMI Award 1333326 , IIS 1741490 and ACI 1429783.

Paper, Code and Slides:

http://rafati.net

jrafatiheravi@ucmerced.edu