Quasi-Newton Optimization For Large-scale Machine Learning

Jacob Rafati
Ph.D., Electrical Engineering and Computer Science
http://rafati.net

A collaboration with:
Dr. Roummel F. Marcia, Professor of Applied Mathematics.

University of California, Merced

Agenda

1. **Introduction to Machine Learning**

2. **Optimization Methods for Large-Scale Machine Learning**

3. **Novel Optimization Methods to Deep Learning:**

   A. Trust-Region and Line-Search Methods for Empirical Risk Minimization in Deep Learning.

   B. Methods for Improving Initialization of quasi-Newton Matrices.

   C. Quasi-Newton Optimization methods in Deep Reinforcement Learning.

4. **Conclusions and Future Work**

1. Introduction to Machine Learning

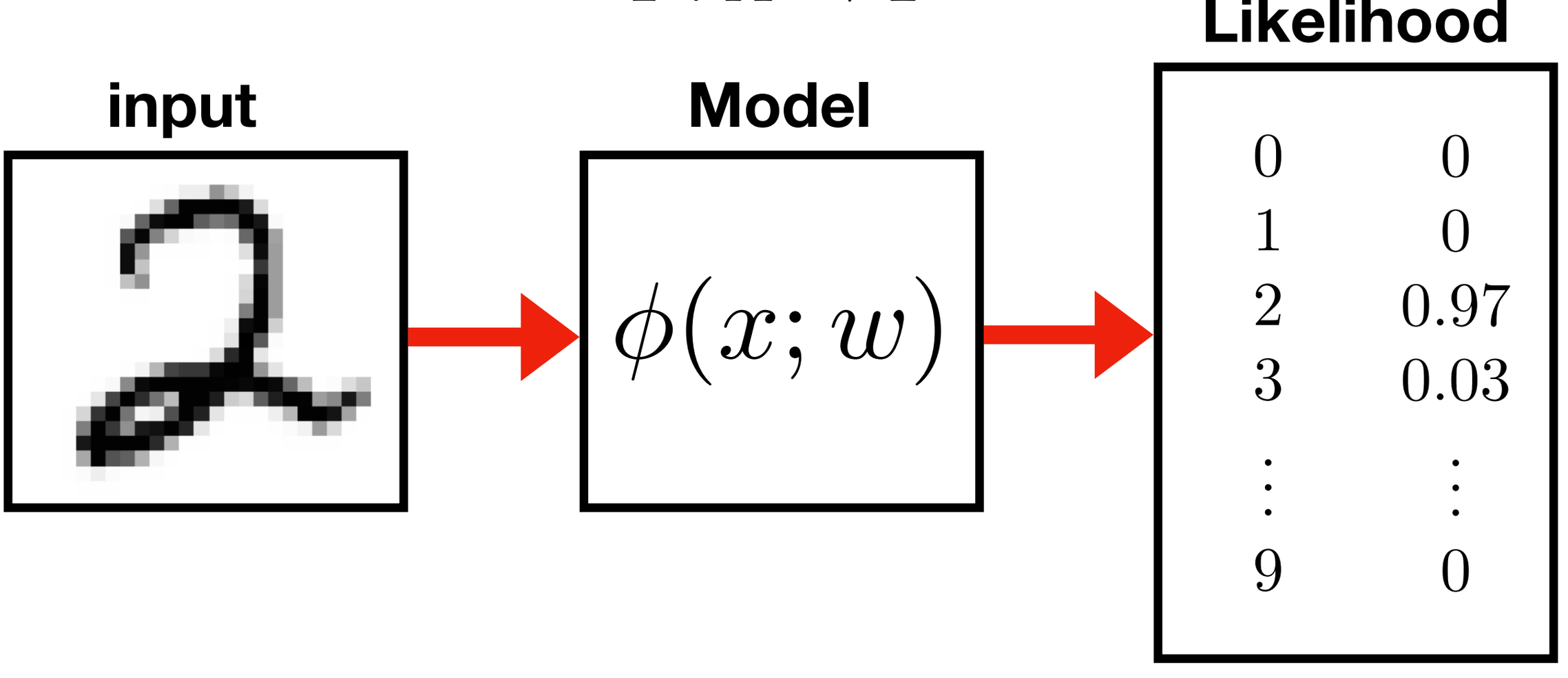
Supervised Learning

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

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

Objective: Learning a mapping from data to labels,

$$\Phi : X \rightarrow T$$



Loss Function

$$y_i = \phi(x_i; w)$$
$$y_{ij} = p(y_{ij} = C_j | x_i; w)$$

Target

$t = [0, \quad 0, \quad 1, \quad 0, \quad \ldots, \quad 0]$

Prediction

$y = [0, \quad 0, \quad 0.97, \quad 0.03, \quad \ldots, \quad 0]$

Cross-Entropy Loss:

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

Empirical Risk

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

Empirical Risk Minimization

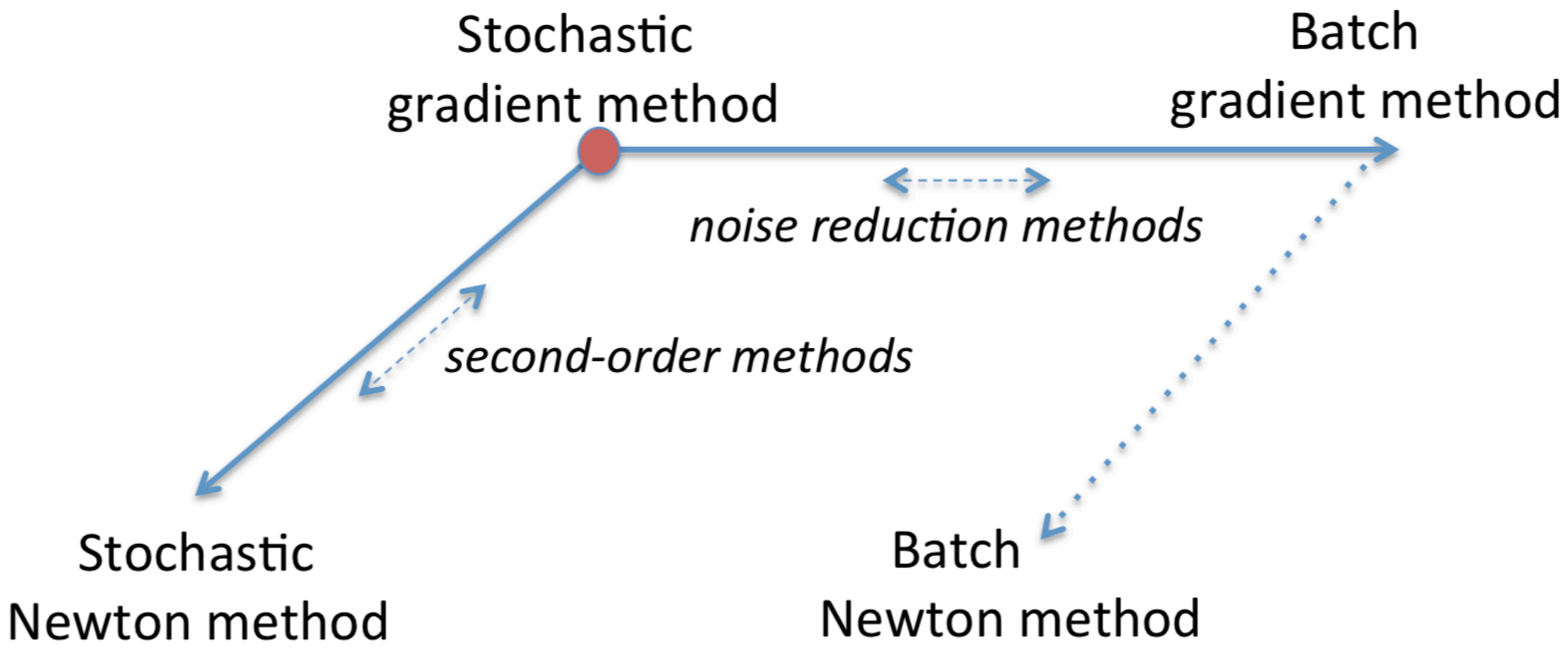
$$\min_{w \in \mathbb{R}^n} \mathcal{L}(w) \triangleq \frac{1}{N} \sum_{i=1}^{N} \ell_i(w)$$

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

- 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.

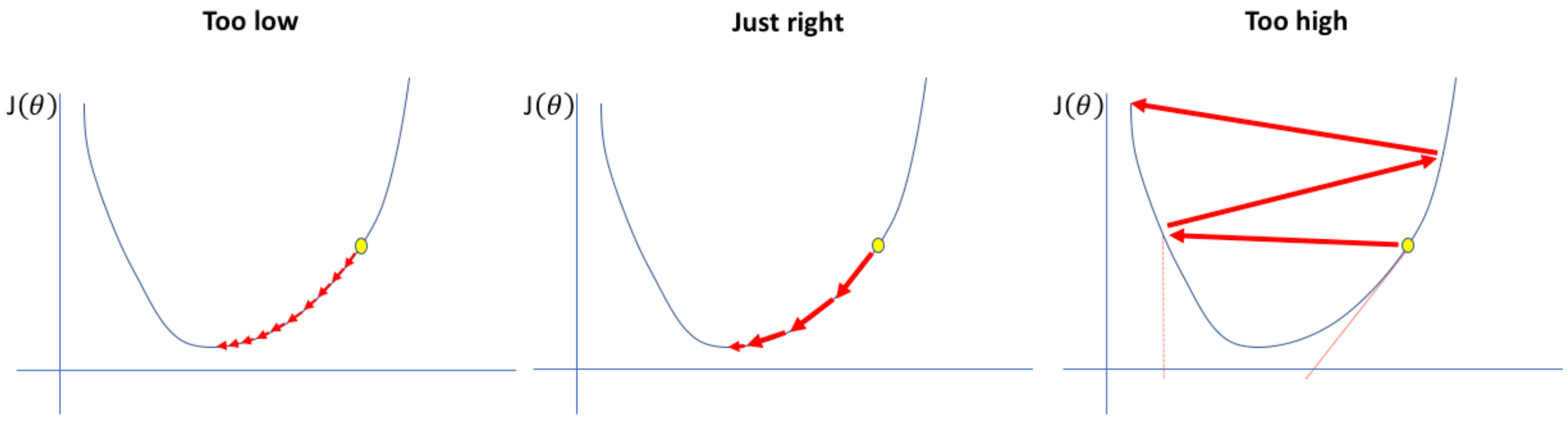
2. Optimization Methods for Large-Scale Machine Learning

Optimization Algorithms



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

Stochastic Gradient Decent



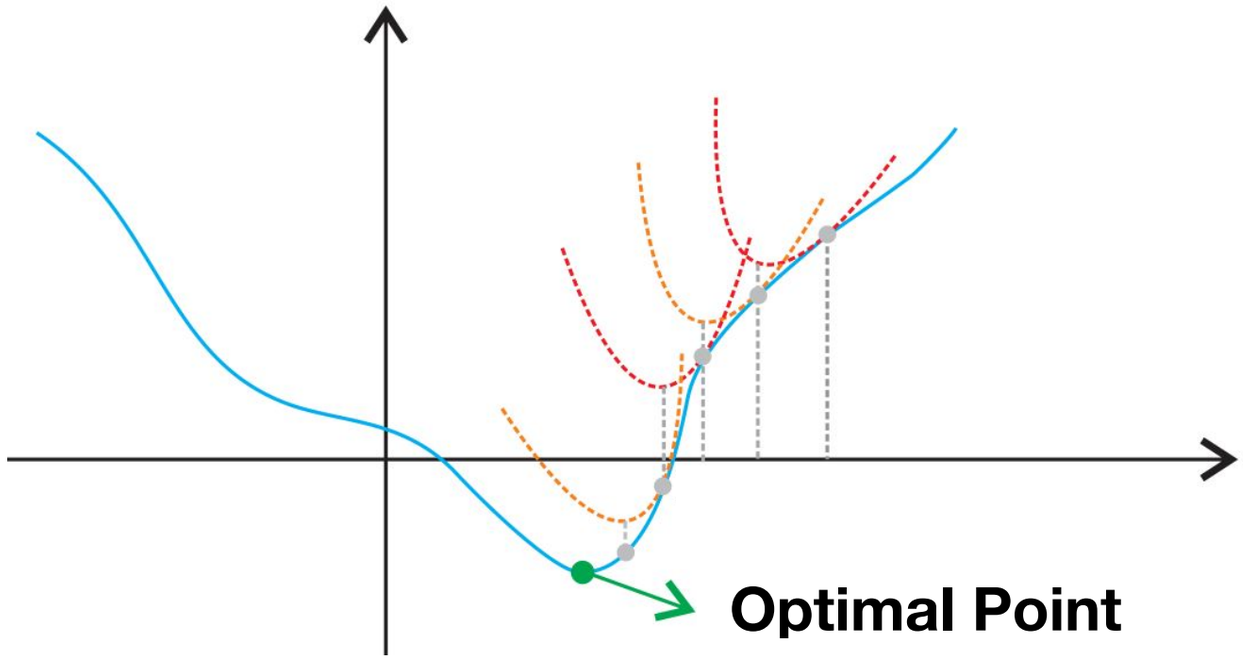
Advantages:

- Simple
- Cost per iteration is fast.

Disadvantages:

- Very sensitive to choice of learning rate.
- Very sensitive to the ill-conditioning problem and scaling.
- Requires fine-tuning many hyper-parameters.
- Can stuck in a saddle-point instead of local minimum.
- Sublinear and slow rate of convergence.

Second-Order Methods



Advantages:

- The rate of convergence is quadratic.
- They are resilient to problem ill-conditioning.
- They involve less parameter tuning.
- They are less sensitive to the choice of hyper-parameters.

Disadvantages:

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

Quasi-Newton Methods

Construct a **low-rank** update of Hessian approximation with first-order gradient informations:

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

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

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

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

- Positive definite matrices.
- 2-rank updates.

BFGS update formula:

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

With an initial matrix:

$$B_0 = \gamma_k I$$

Displacement Vector:

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

Gradients Difference Vector:

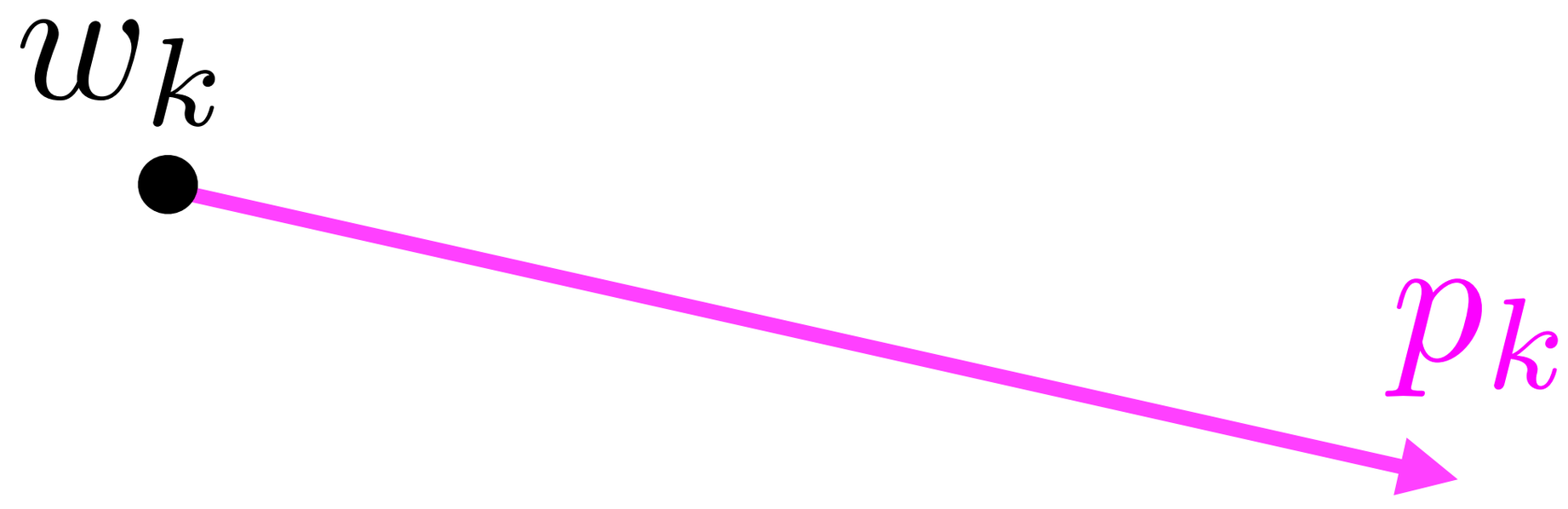
$$y_k \triangleq \nabla \mathcal{L}(w_{k+1}) - \nabla \mathcal{L}(w_k)$$

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

Compact Representation for Limited-Memory BFGS

$$B_k = \gamma_k I + \begin{bmatrix} \Psi_k & M_k & \Psi_k^T \end{bmatrix}$$

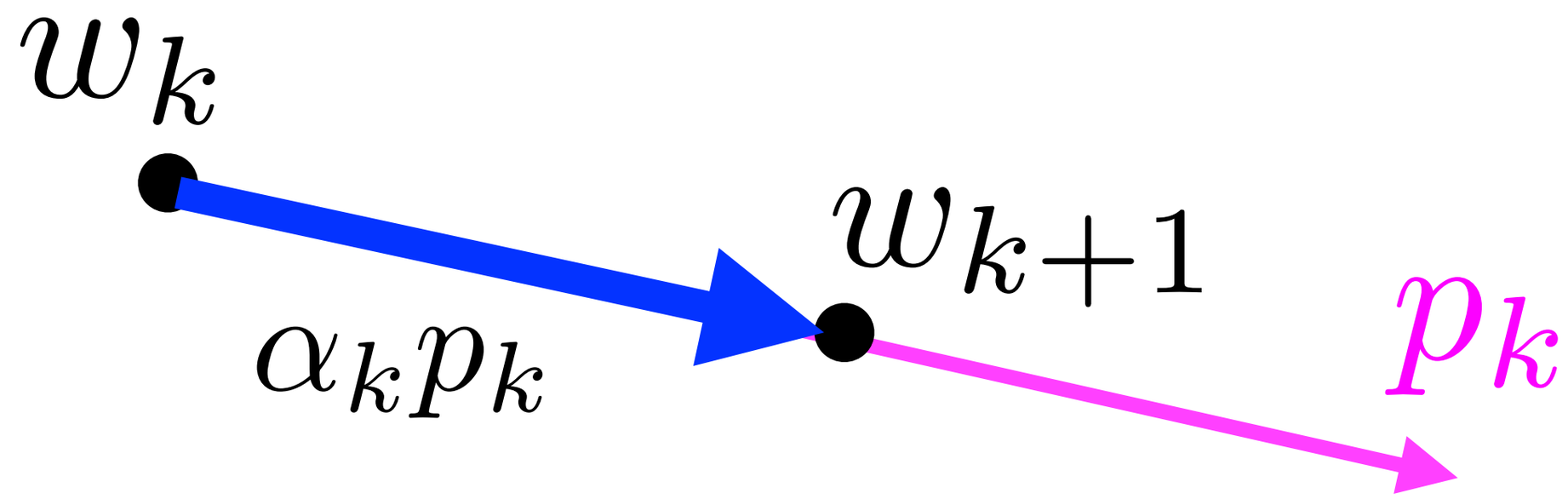
Line Search Strategy



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

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

Line Search Strategy



Next we should find a proper step size by solving

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

Instead we satisfy the sufficient decrease and curvature conditions

Wolfe conditions

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

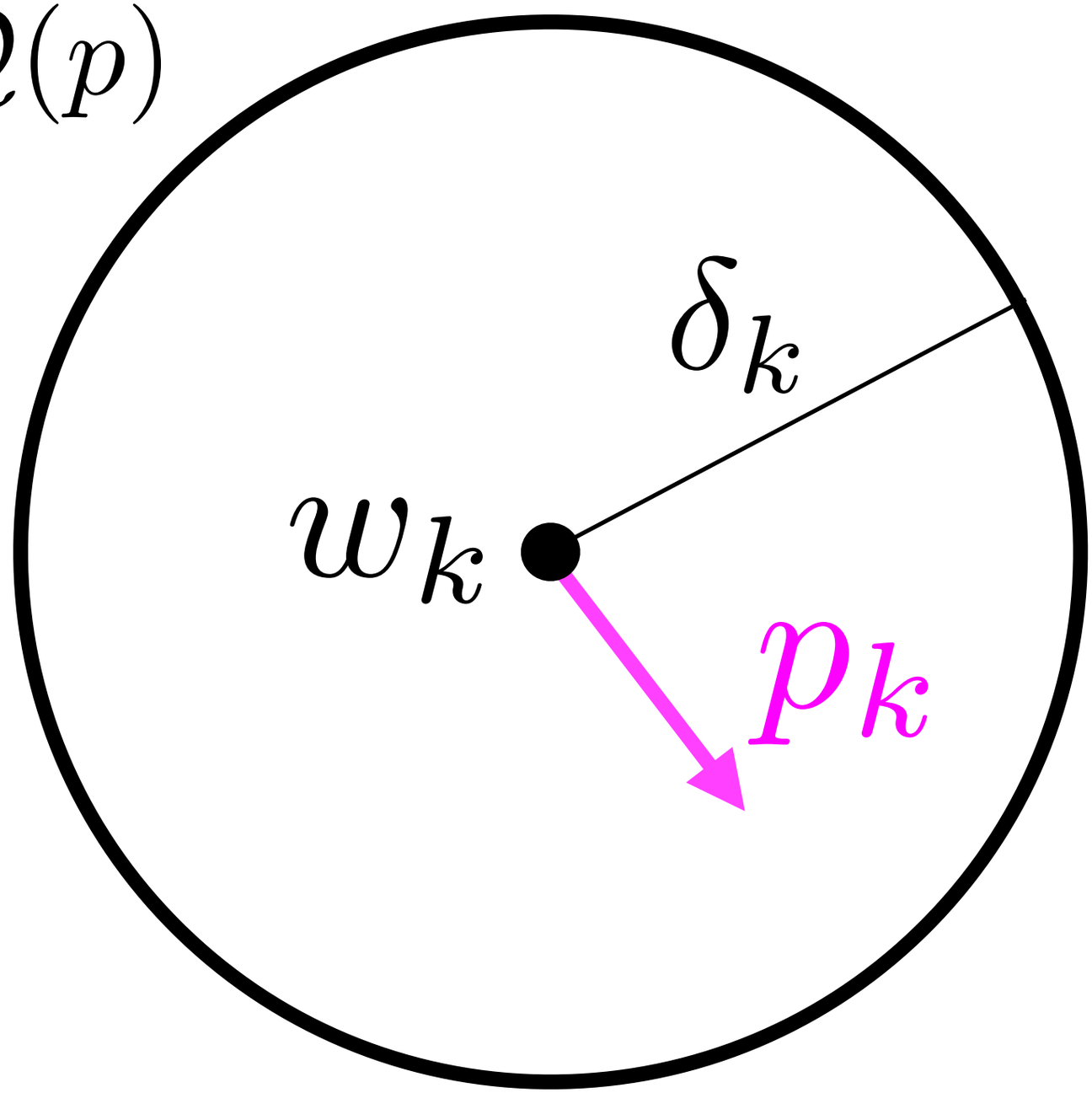
$$\nabla \mathcal{L}(w_k + \alpha_k p_k)^T p_k \geq c_2 \nabla f(w_k)^T p_k$$

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

Trust-Region Srategy

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

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



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

3. Novel Optimization Methods to Deep Learning

3A. Trust-Region and Line-Search Methods for Empirical Risk Minimization in Deep Learning

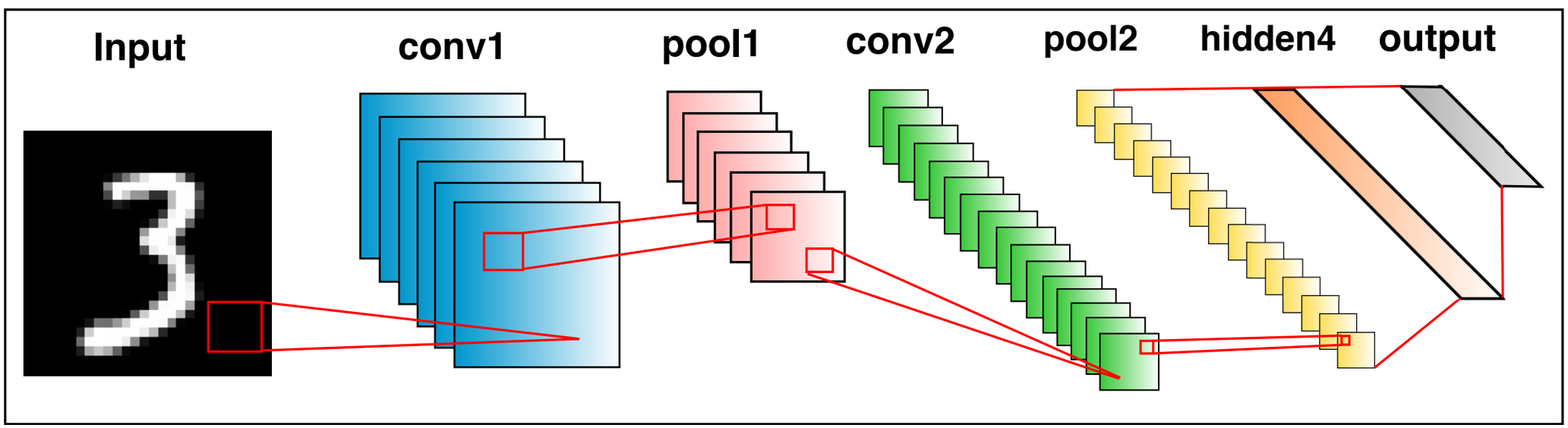
Objective:

- Implementing fast and robust stochastic quasi-Newton optimization in deep learning.

Contribution:

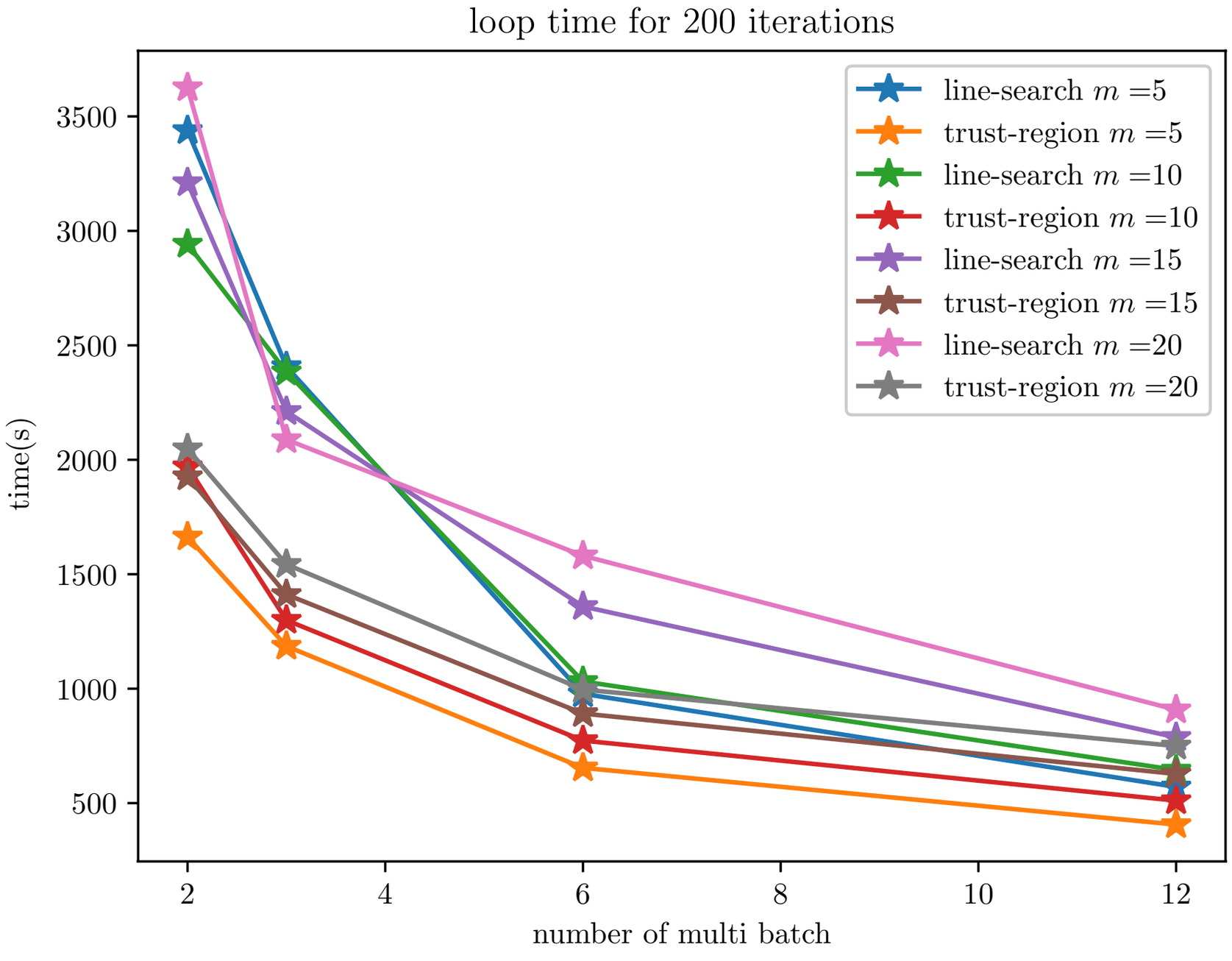
- Multi-batch Stochastic L-BFGS in Line Search.
- Multi-Batch Stochastic L-BFGS in Trust-Region.

Experiment on MNIST



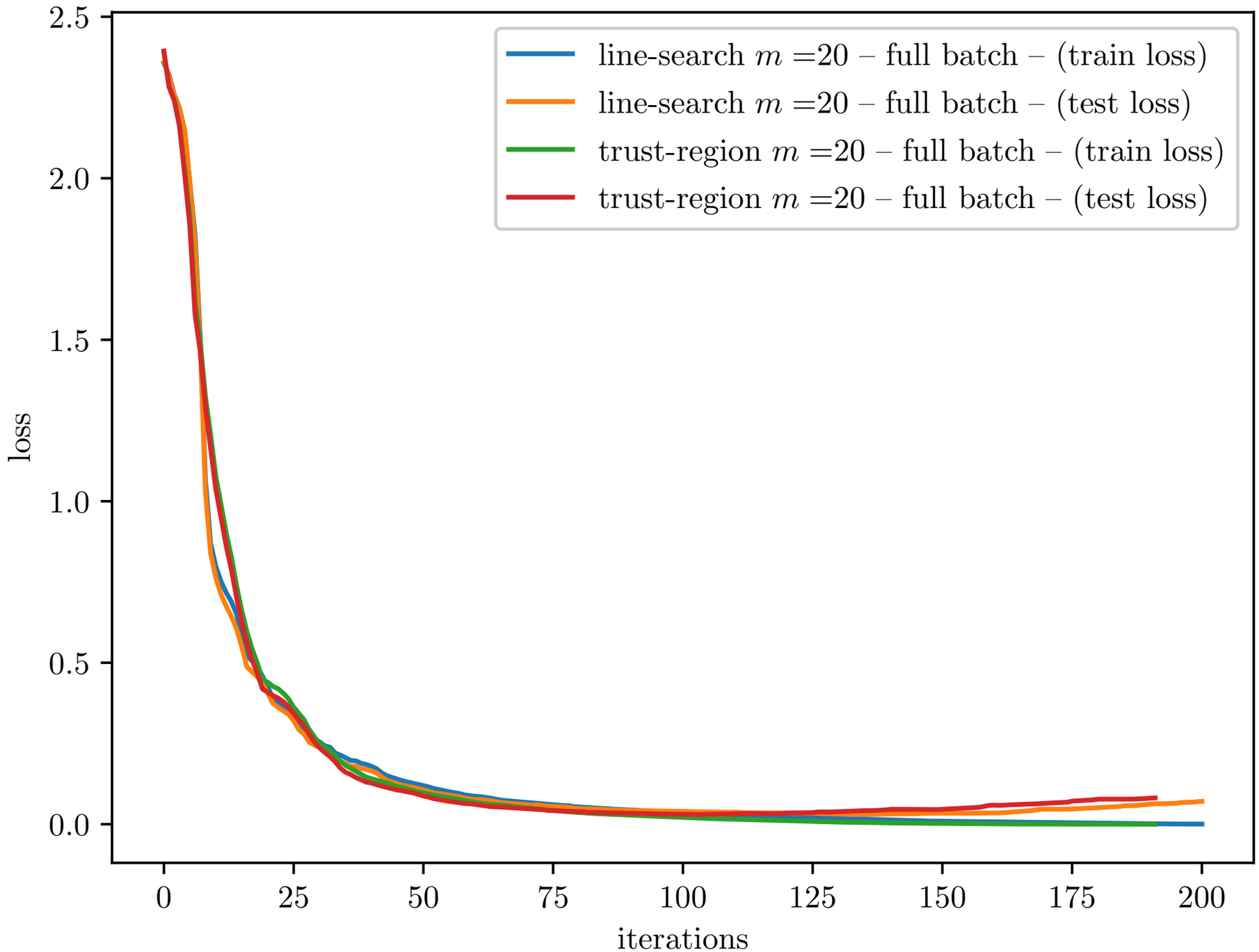
Jacob Rafati, and Roummel F. Marcia (2018). Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning. (ICMLA 2018), Orlando, Florida.

Trust-region vs Line-Search Training Time - MNIST



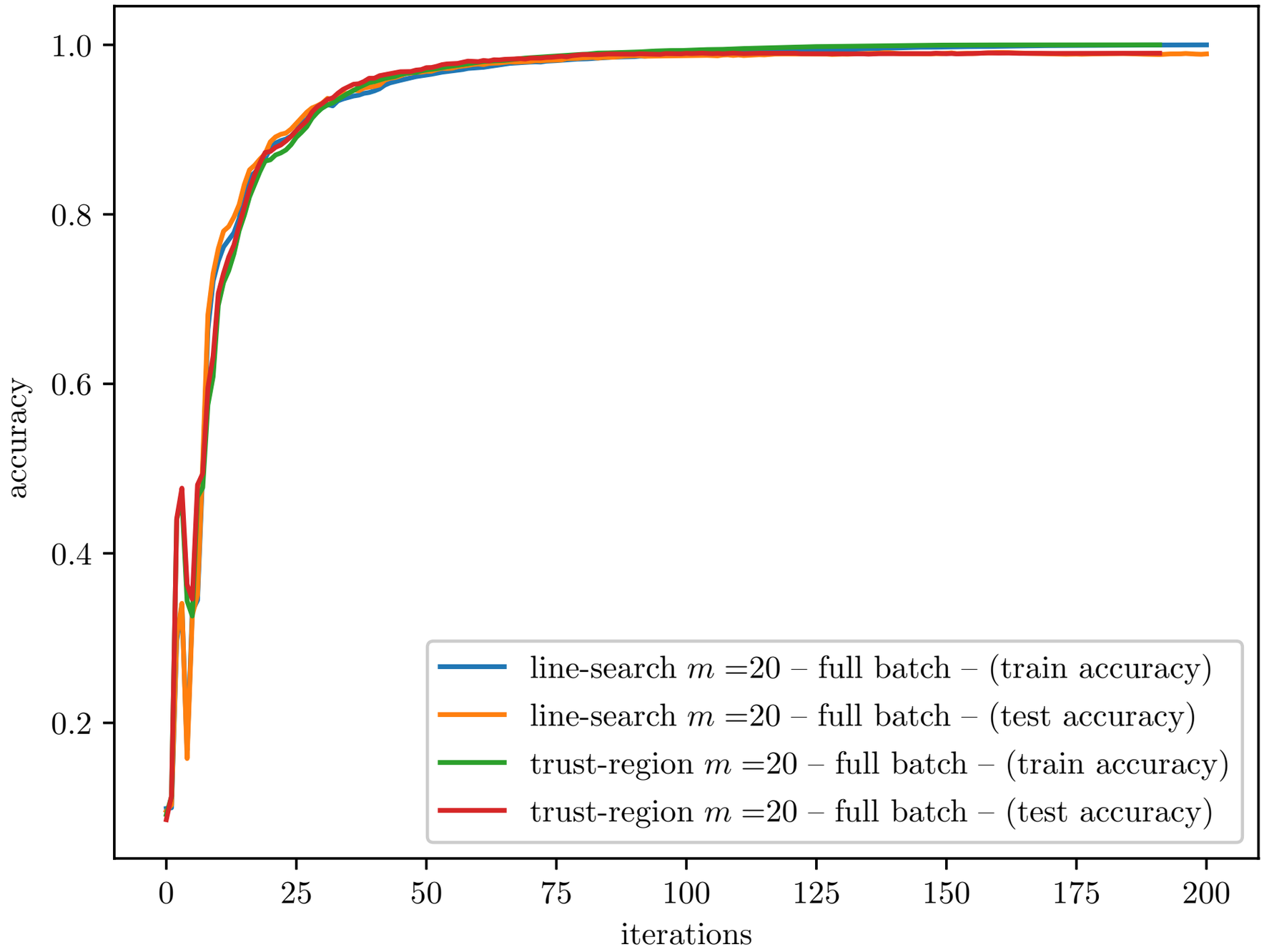
Rafati et al. (2018). Trust-Region Minimization Algorithm for Training Responses. In 26th European Signal Processing Conference, Rome, Italy.

Trust-region vs Line-Search Training/Test Loss - MNIST



Rafati et al. (2018). Trust-Region Minimization Algorithm for Training Responses. In 26th European Signal Processing Conference, Rome, Italy.

Trust-region vs Line-Search Training/Test Accuracy - MNIST

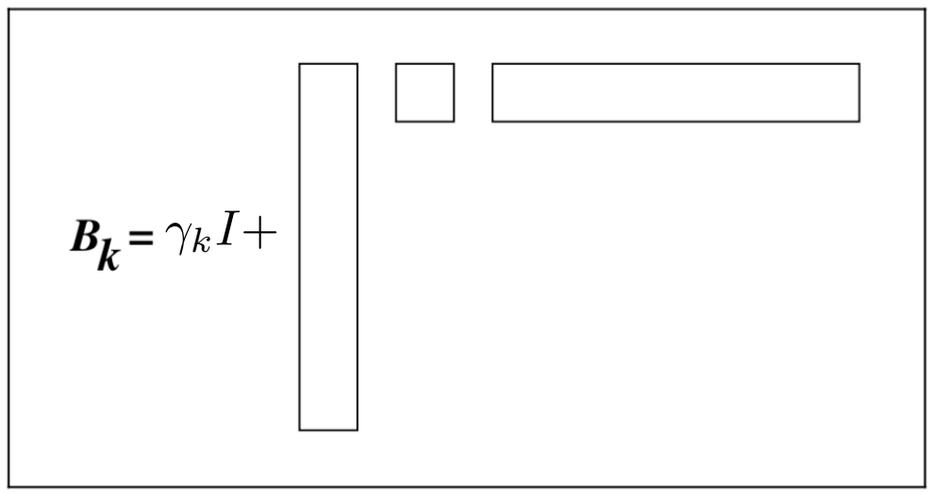


Rafati et al. (2018). Trust-Region Minimization Algorithm for Training Responses. In 26th European Signal Processing Conference, Rome, Italy.

3B. Methods for Improving Initialization of Quasi-Newton Matrices

Initialization Methods for L-BFGS

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



How can we improve the performance by choosing a proper initialization

$$B_0 = \gamma_k I, \quad \gamma_k > 0$$

Initialization Method I

$$B_0 = \gamma_k I, \quad \gamma_k > 0$$

Method I is used in literature. This method finds the best initial positive matrix that best represent the Hessian, i.e. spectral estimate of Hessian.

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

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

Initialization Methods II,III

Consider the compact representation of L-BFGS

$$B_k - \gamma_k I = \Psi_k M_k \Psi_k^T$$

We should bound γk in order to avoid a false curvature condition in $\Psi_k M_k \Psi_k^T$.

By solving a general Eigenvalue problem

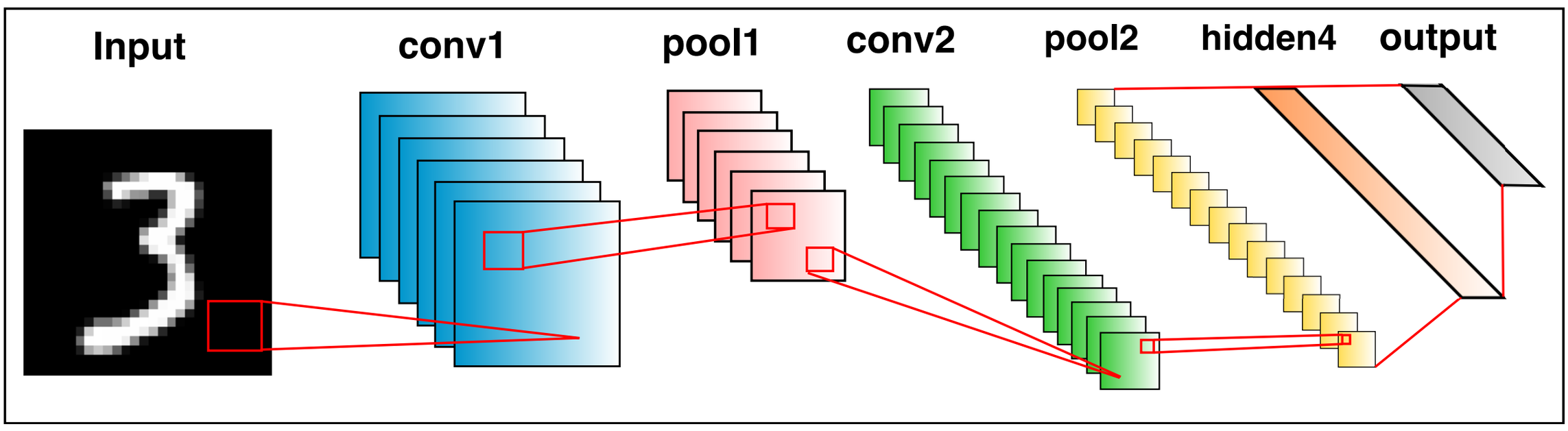
$$Az = \lambda Bz$$

$$\gamma_k < \lambda_{\min}$$

Jacob Rafati, and Roummel F. Marcia (2018). Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning. (ICMLA 2018), Orlando, Florida.

Experiment on MNIST

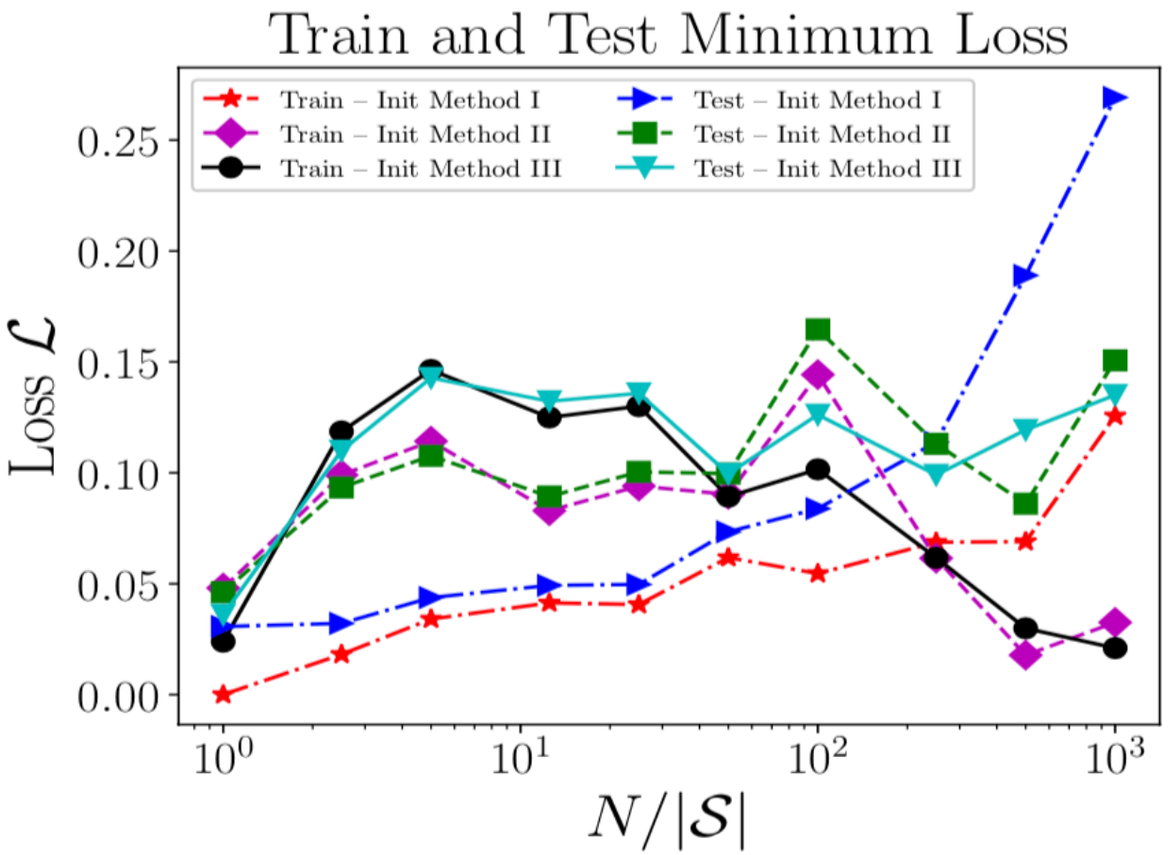
Initialization	Source	Formula
Method I	Solve the optimization problem: $\gamma_k = \arg\min_\gamma \|B_0^{-1} y_{k-1} - s_{k-1}\|_2^2$	$\gamma_k = \max\left\{1, \frac{y_{k-1}^T y_{k-1}}{s_{k-1}^T y_{k-1}}\right\}$
Method II	Solve the generalized eigenvalue problem: $(L_k + D_k + L_k^T) z = \lambda S_k^T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



Jacob Rafati, and Roummel F. Marcia (2018). Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning. (ICMLA 2018), Orlando, Florida.

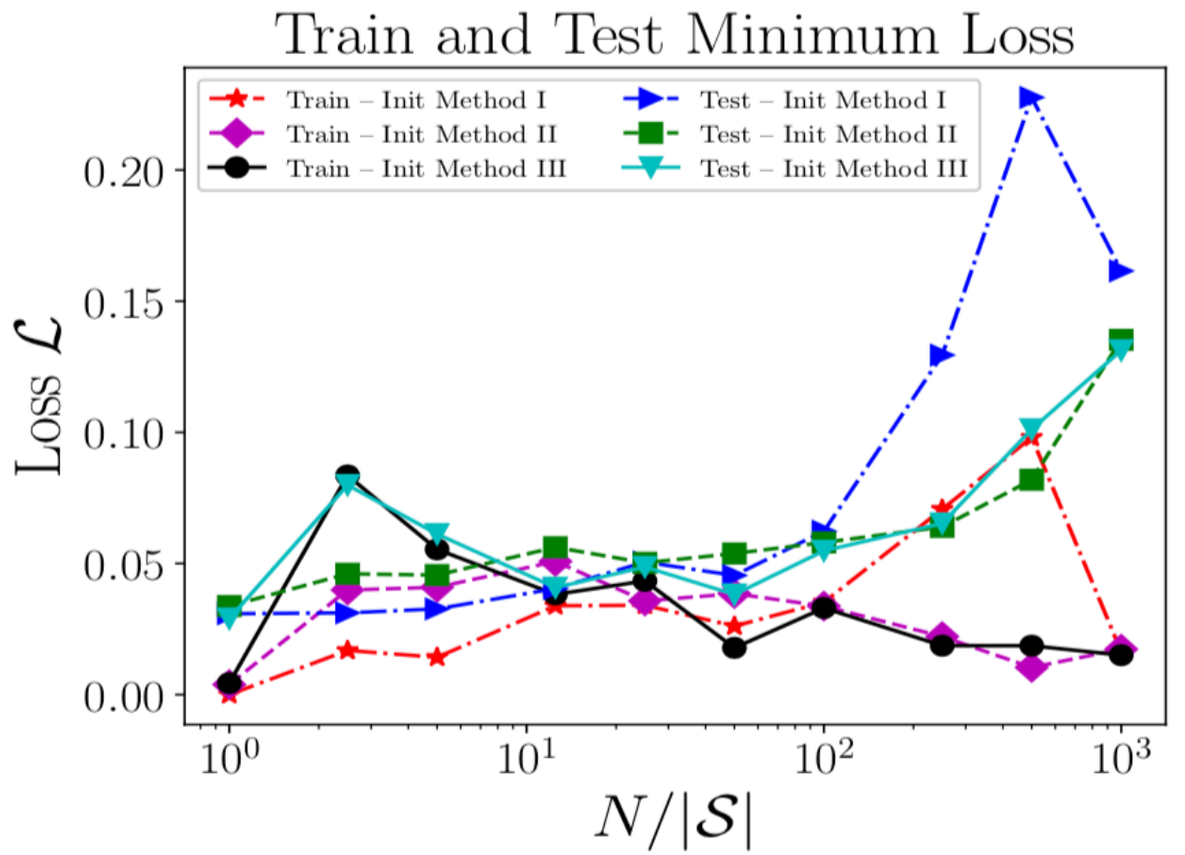
Training/Test Loss

$m = 10$



Full ⟷ **Stochastic**

$m = 20$

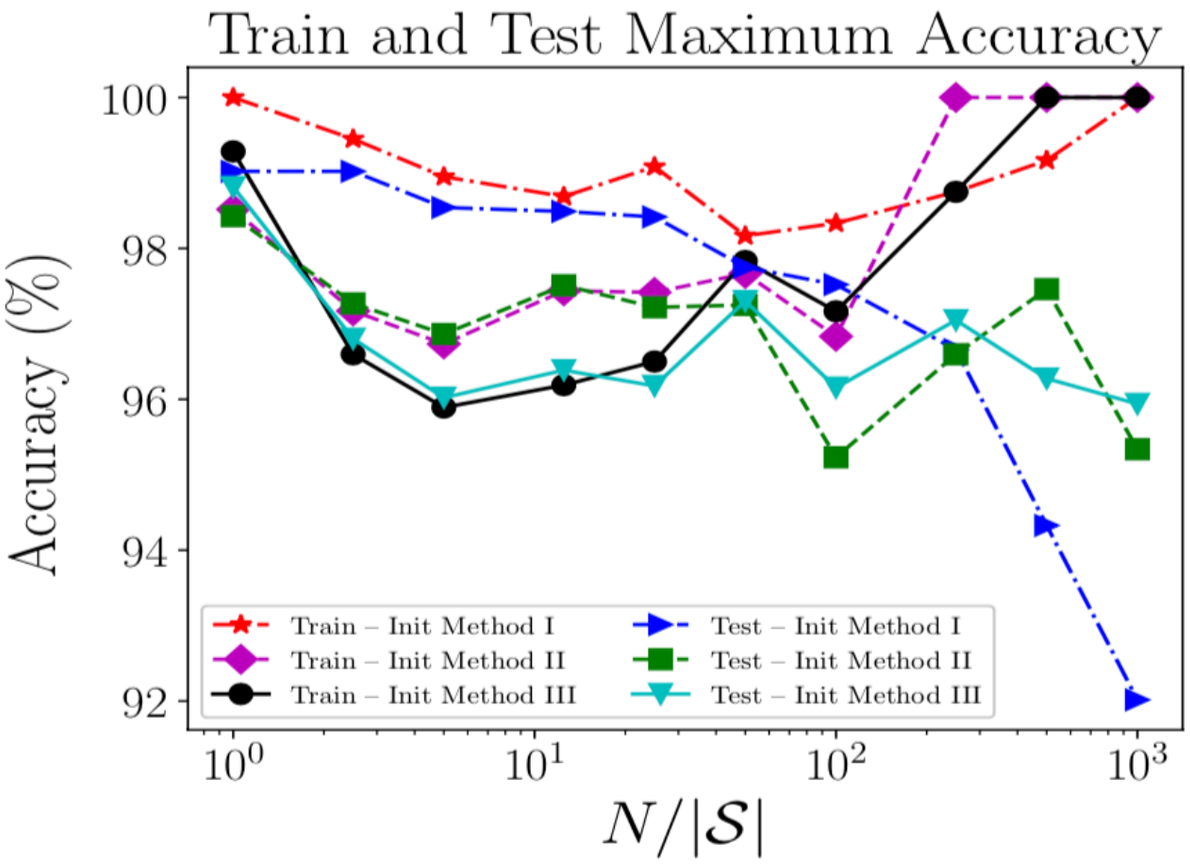


Full ⟷ **Stochastic**

Jacob Rafati, and Roummel F. Marcia (2018). Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning. (ICMLA 2018), Orlando, Florida.

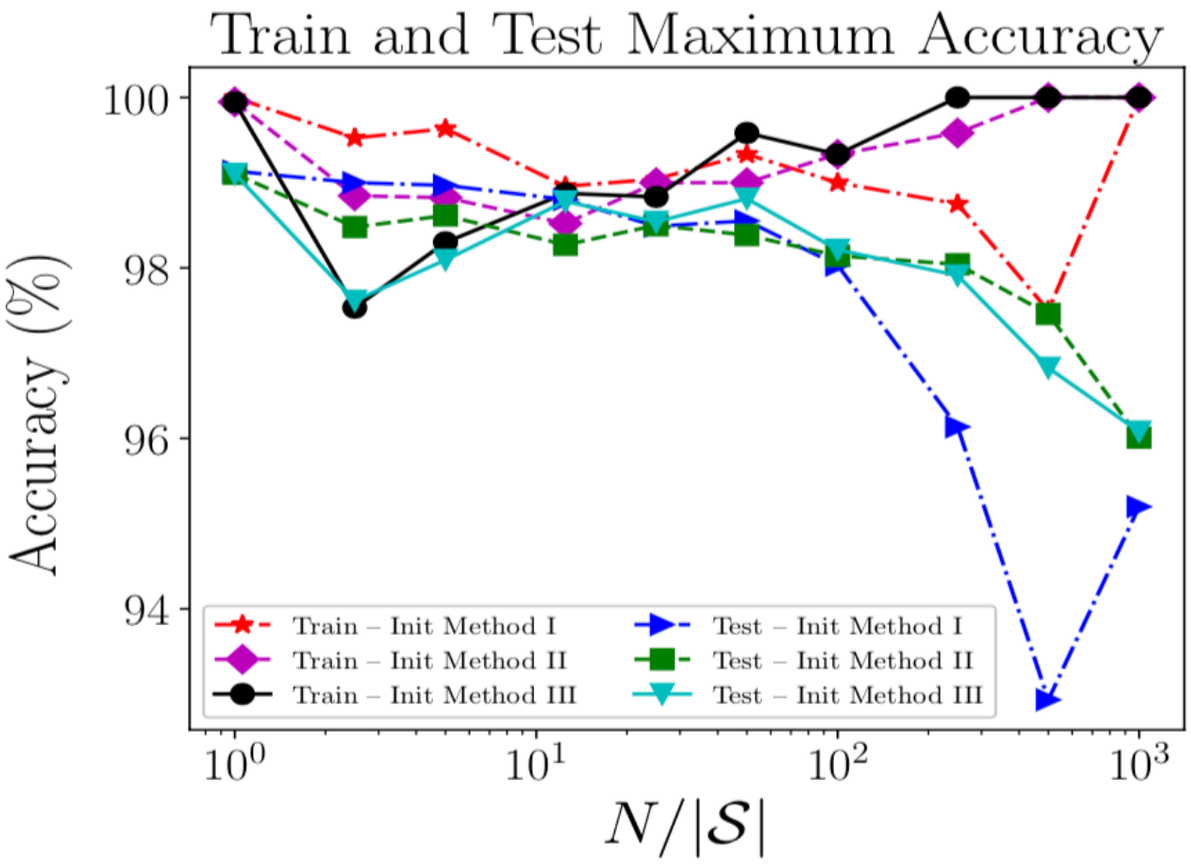
Training/Test Accuracy

$m = 10$



Full ⟷ **Stochastic**

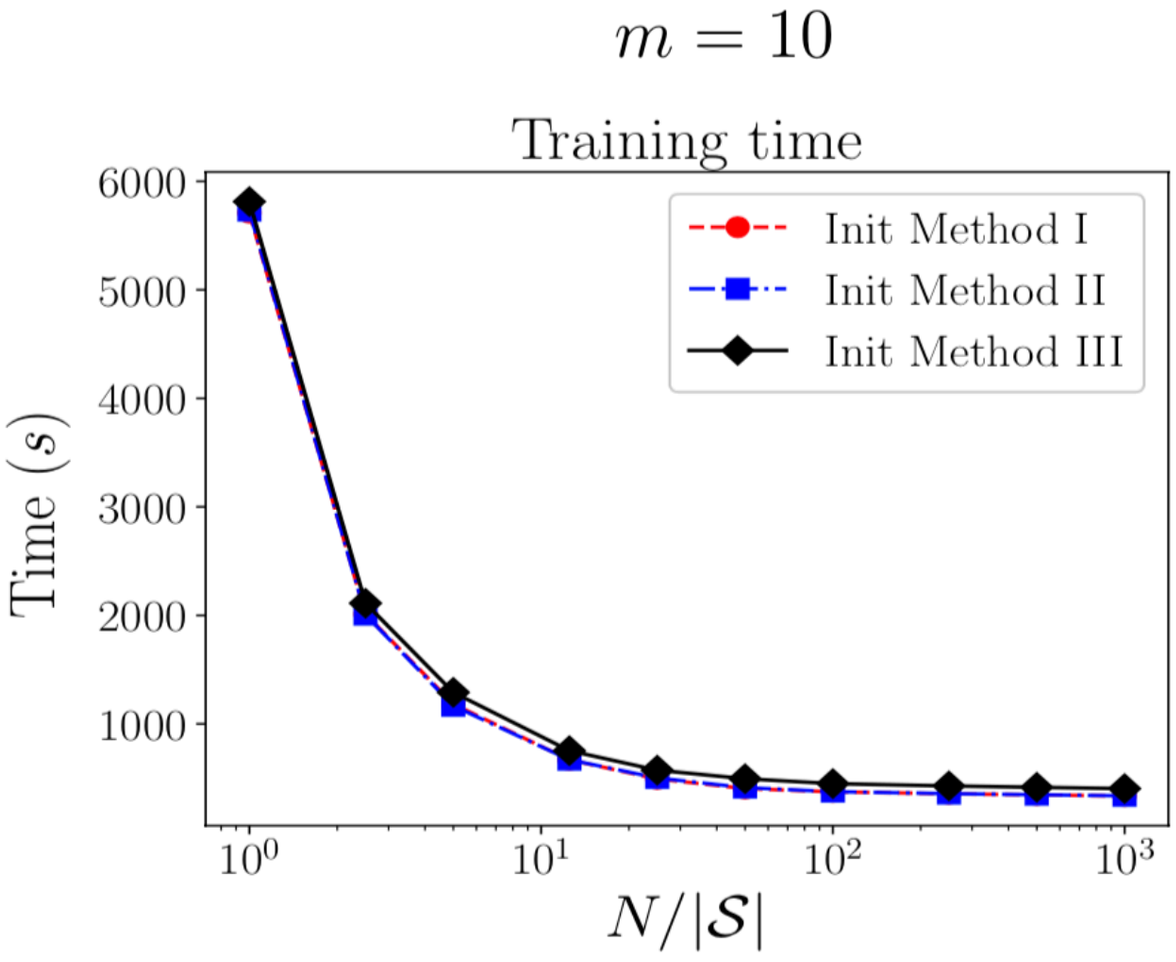
$m = 20$



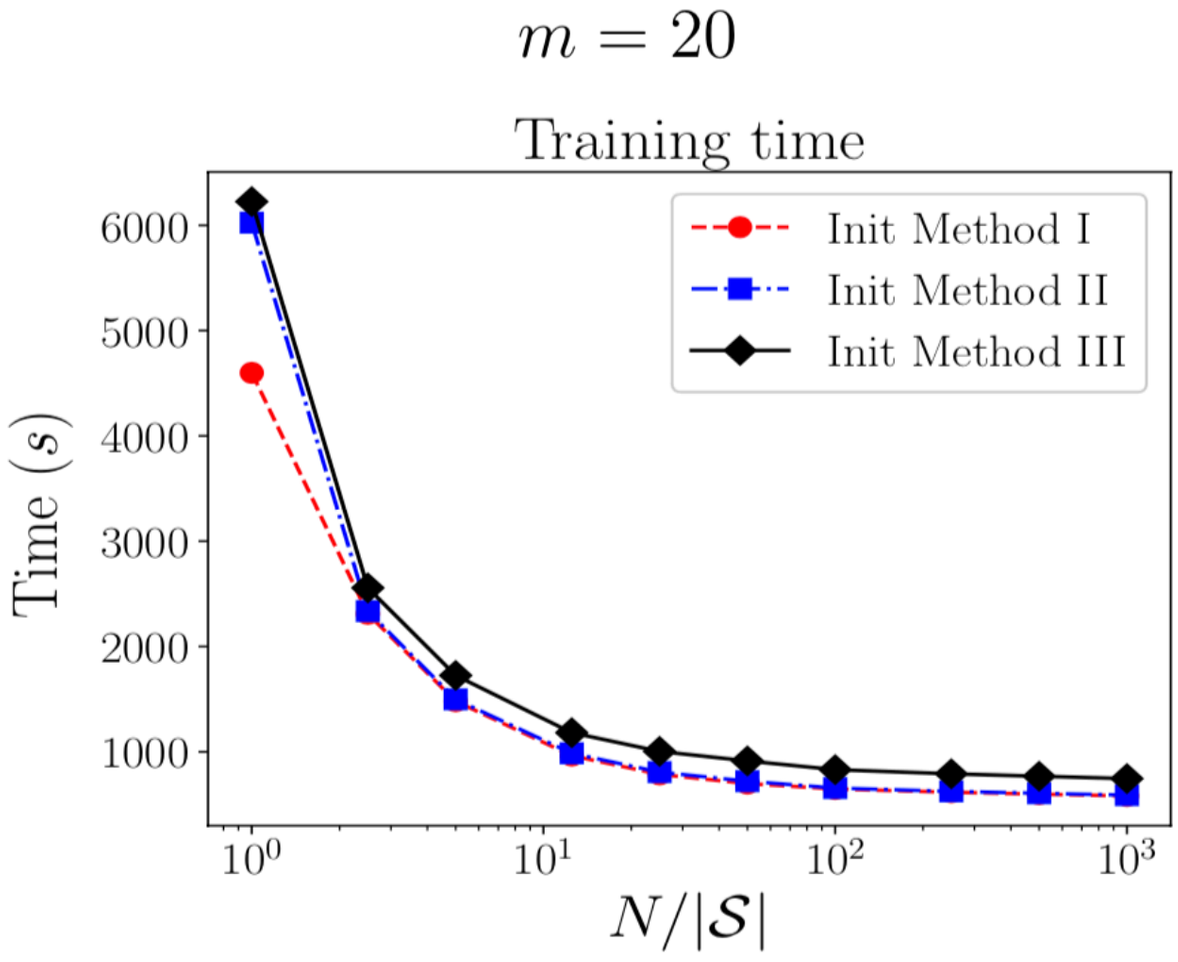
Full ⟷ **Stochastic**

Jacob Rafati, and Roummel F. Marcia (2018). Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning. (ICMLA 2018), Orlando, Florida.

Training Time



Full ⟷ **Stochastic**

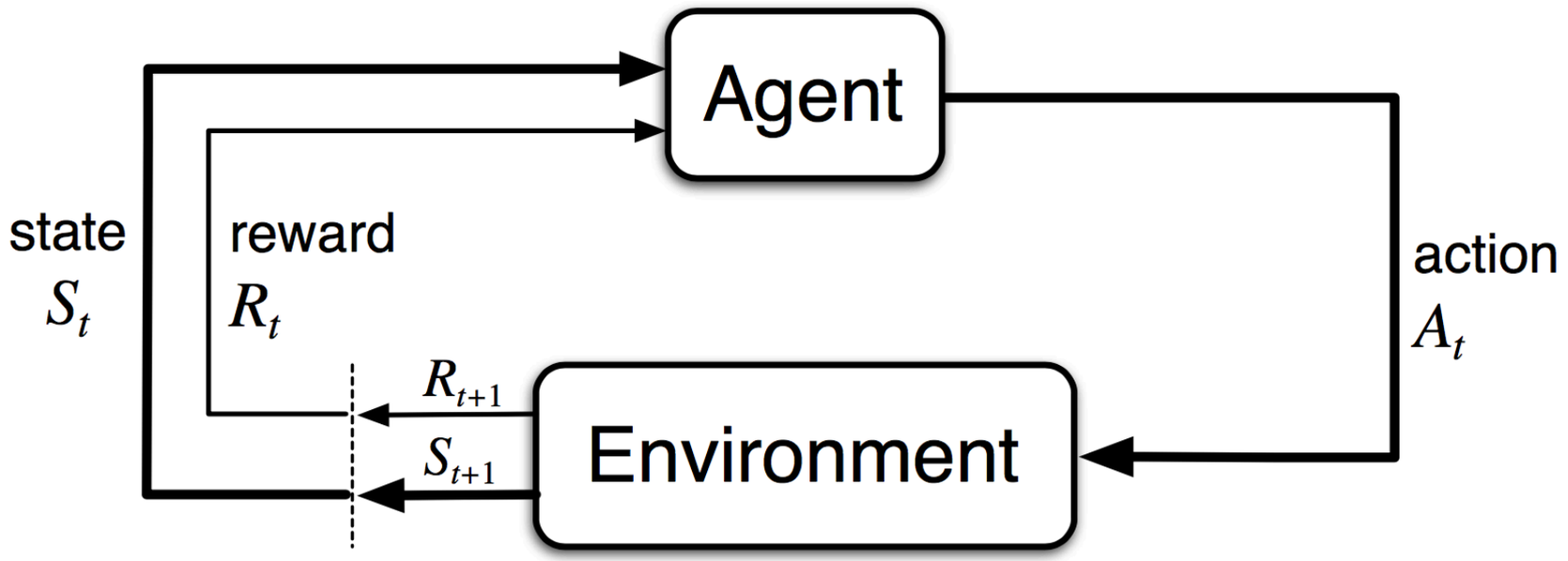


Full ⟷ **Stochastic**

Jacob Rafati, and Roummel F. Marcia (2018). Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning. (ICMLA 2018), Orlando, Florida.

3C. Quasi-Newton Optimization Methods in Deep RL

Reinforcement Learning

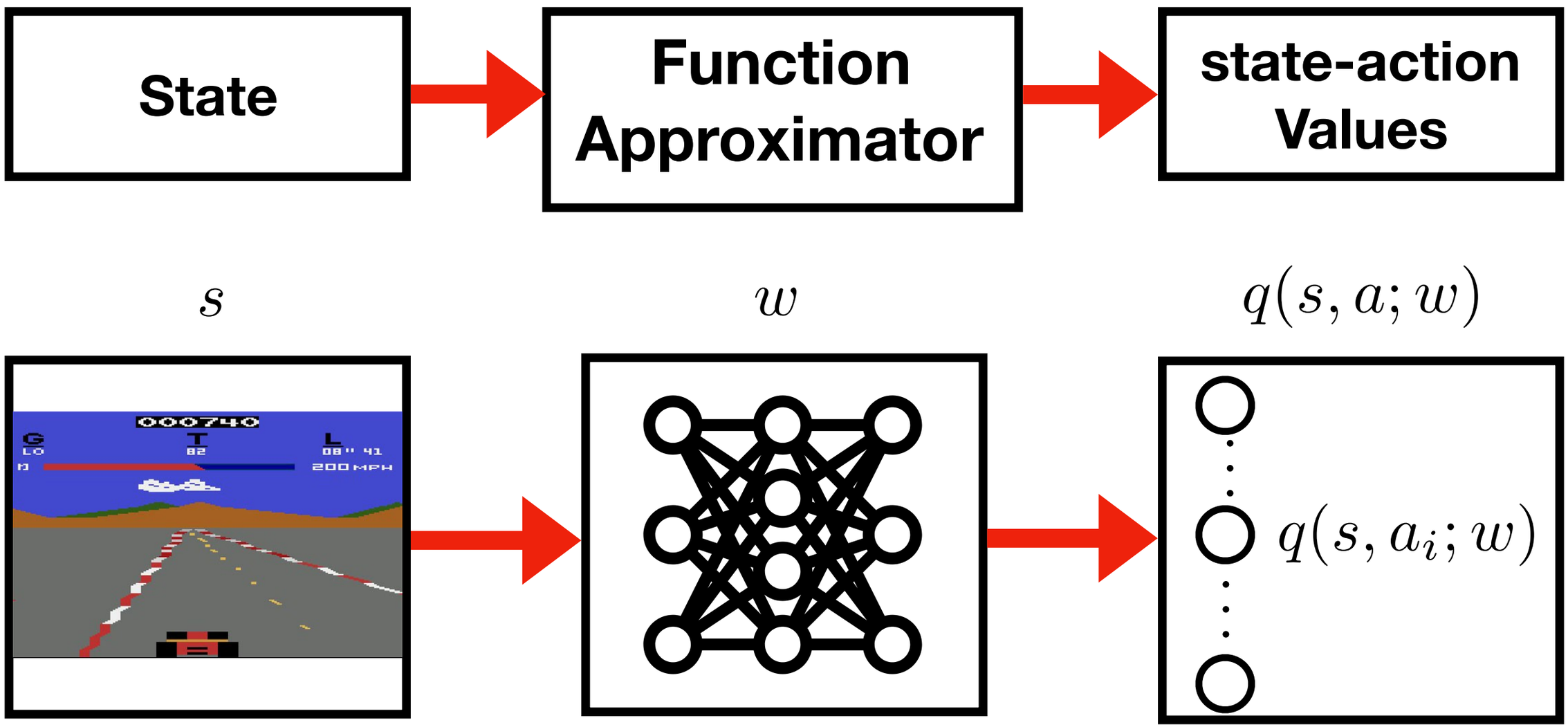


Reinforcement learning (RL) is learning how to map situations (**states**) to decisions (**actions**) to **maximize future rewards.**

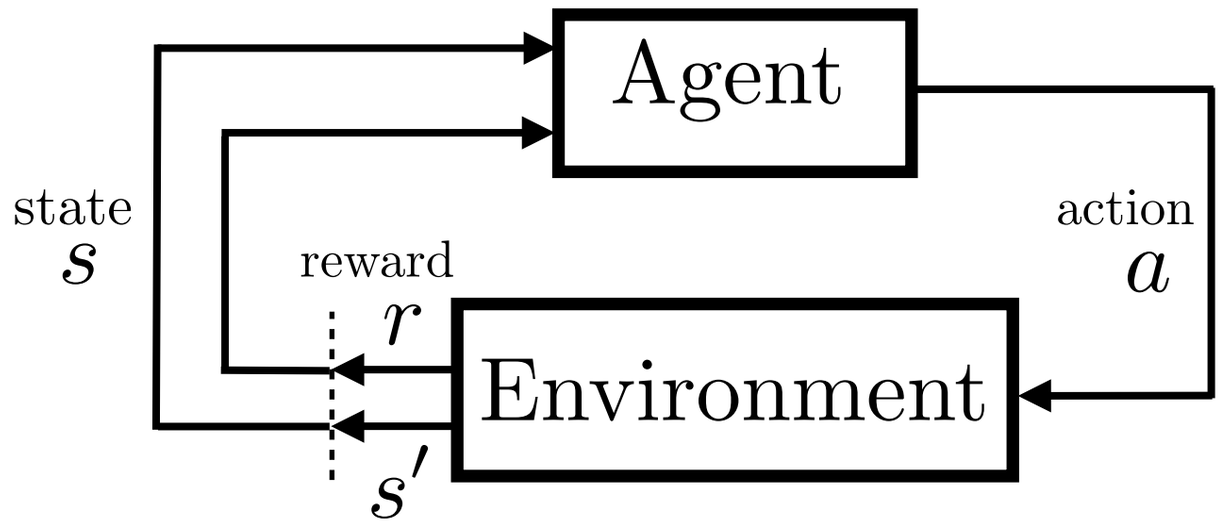
Sutton and Barto (2017). Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, USA, 2nd edition.

Generalization

$$Q(s, a) \approx q(s, a; w)$$



Optimization in Deep RL

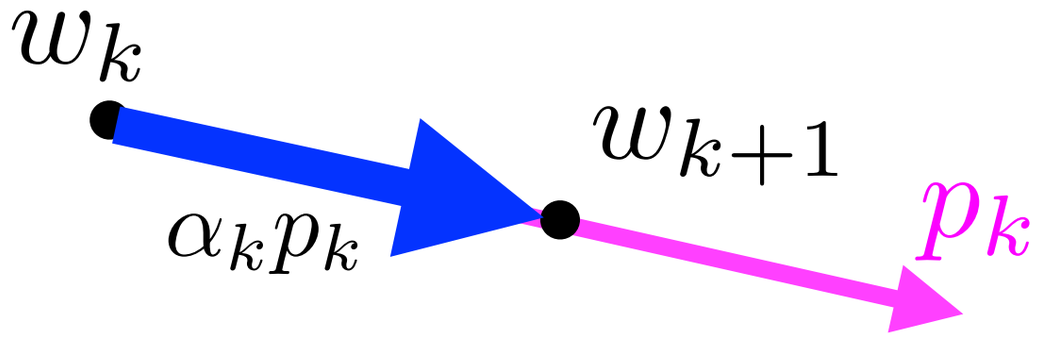


$$\min_{w \in \mathbb{R}} \mathcal{L}(w) \triangleq \frac{1}{N} \sum_{e \in \mathcal{D}} \left(r + \gamma \max_{a'} q(s', a'; w) - q(s, a; w) \right)^2$$

$\mathcal{D} = \{(s, a, s', r)\}$ is Agent's Experiences Memory.

Sutton and Barto (2017). Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, USA, 2nd edition.

Line Search Method



Search step, the minimizer of quadratic model

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

There is a formula to update inverse of BFGS matrices, $H_k = B_k^{-1}$

$$H_{k+1} = \left(I - \frac{y_k s_k^T}{y_k^T s_k}\right) H_k \left(I - \frac{s_k y_k^T}{y_k s_k^T}\right) + \frac{y_k y_k^T}{y_k s_k^T}$$

Instead we satisfy the sufficient decrease and curvature conditions known as

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.

Convergence Analysis

Theorem. Under these assumptions:

$\mathcal{L}(w)$ is strongly convex, and twice differentiable.

$\forall w,\ \exists \lambda, \Lambda > 0$ such that $\lambda I \preceq \nabla^2 \mathcal{L}(w) \mathcal{L} \preceq \Lambda I,$ i.e. Hessian is bounded.

$\forall w,\ \exists \eta > 0$ such that $\|\nabla \mathcal{L}(w)\|^2 \leq \eta^2,$ i.e. Gradient does not explode.

$\exists \lambda', \Lambda' > 0$ *such that* $\lambda' I \preceq H_k \preceq \Lambda' I.$

If we bound the step size, $\alpha < \frac{1}{2\lambda\lambda'}$, we can compute an upper-bound for loss

$$\mathcal{L}(w_k) - \mathcal{L}(w^*) \leq (1 - 2\alpha\lambda\lambda')^k [\mathcal{L}(w_0) - \mathcal{L}(w^*)]$$
$$+ [1 - (1 - 2\alpha\lambda\lambda')^k] \frac{\alpha^2 \Lambda'^2 \Lambda \eta^2}{4\lambda'\lambda}$$

i.e. the loss function will converge to a neighborhood of the optimal loss.

Jacob Rafati, and Roummel F. Marcia (2019). Quasi-Newton Optimization in Deep Q-Learning for Playing ATARI Games. arXiv e-print (arXiv:1811.02693).

Value Optimality

Theorem.

If α_k satisfies

$$\left|1 - \alpha_k \nabla Q_k^T H_k \nabla Q_k + \frac{\alpha_k^2}{2} \nabla Q_k^T H_k \nabla^2 Q_k H_k \nabla \mathcal{L}_k\right| < 1,$$

then

$$\|Q_{k+1} - Q^*\|_\infty < \|Q_k - Q^*\|_\infty.$$

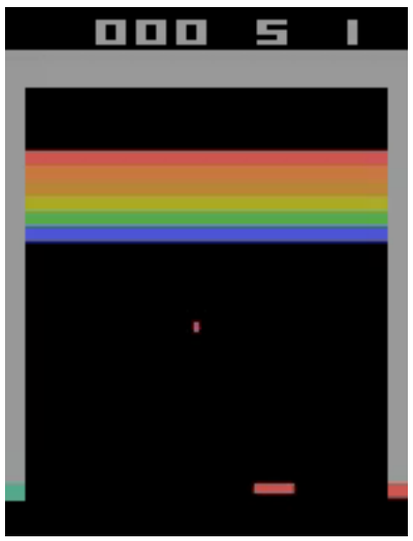
By applying another condition on step size,
the value function theoretically converges to optimal values.

Jacob Rafati, and Roummel F. Marcia (2019). Quasi-Newton Optimization in Deep Q-Learning for Playing ATARI Games. arXiv e-print (arXiv:1811.02693).

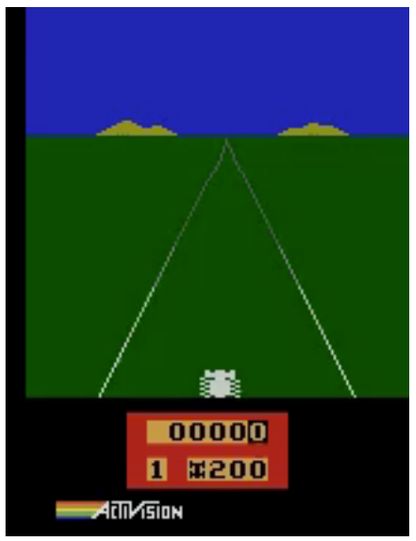
Experiments - ATARI 2600

Beam Rider

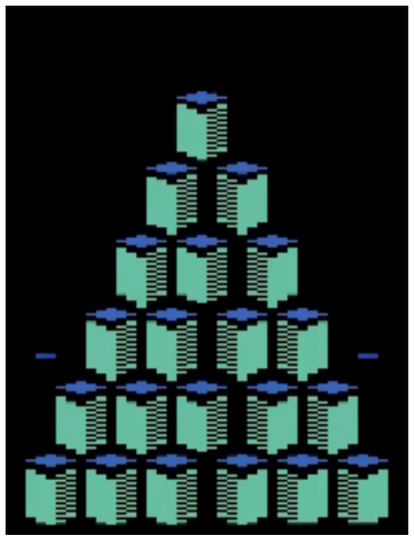
Breakout



Enduro

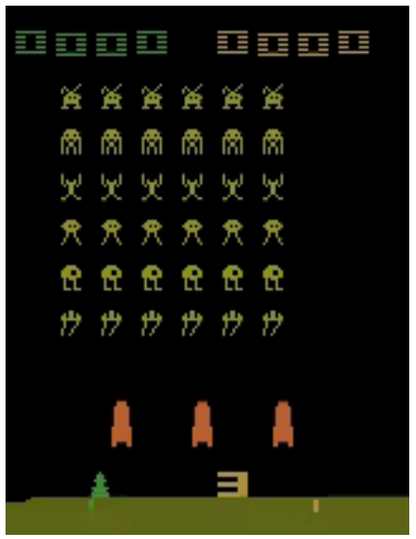


Qbert



Seaquest

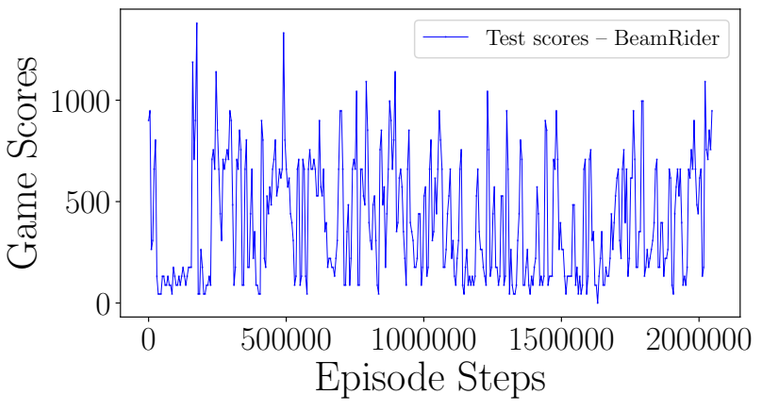
Space Invadors



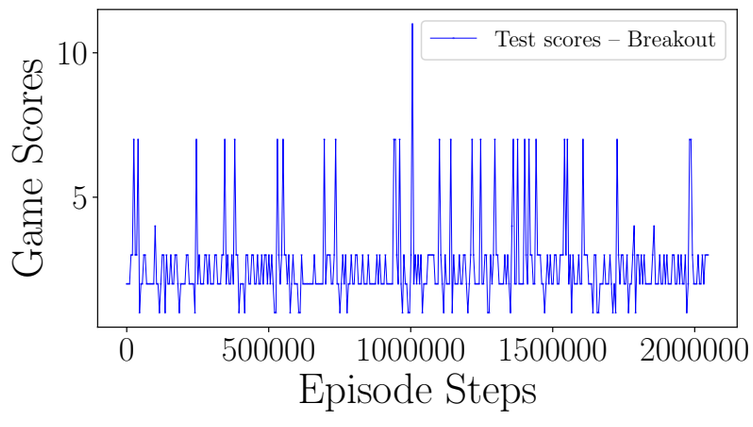
We used one NVIDIA Tesla K40 GPU with 12GB GDDR5 RAM on MERCED clusters.

Test Scores

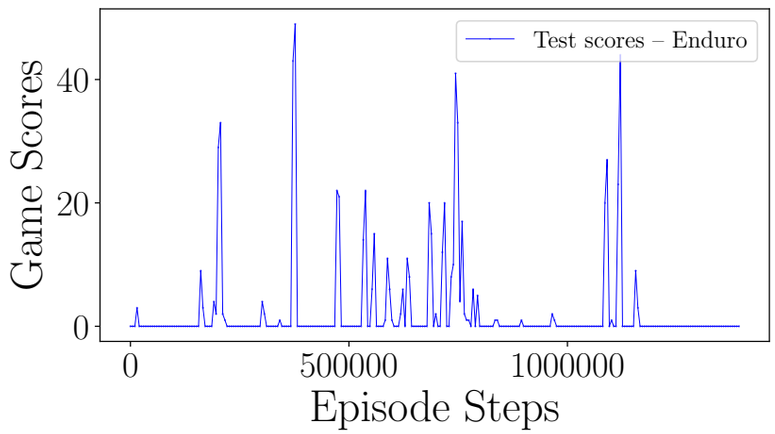
Beam Rider



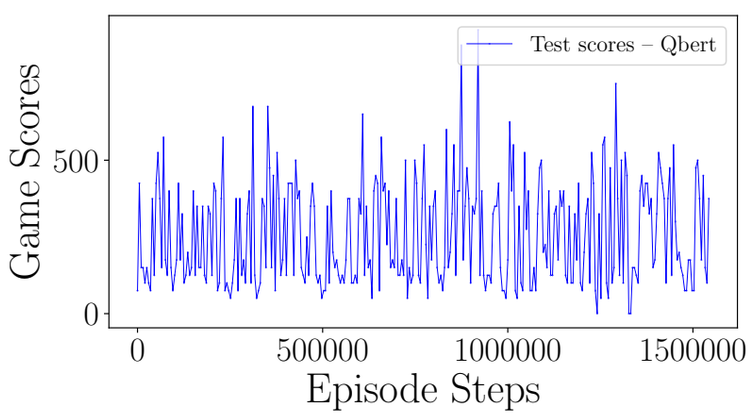
Breakout



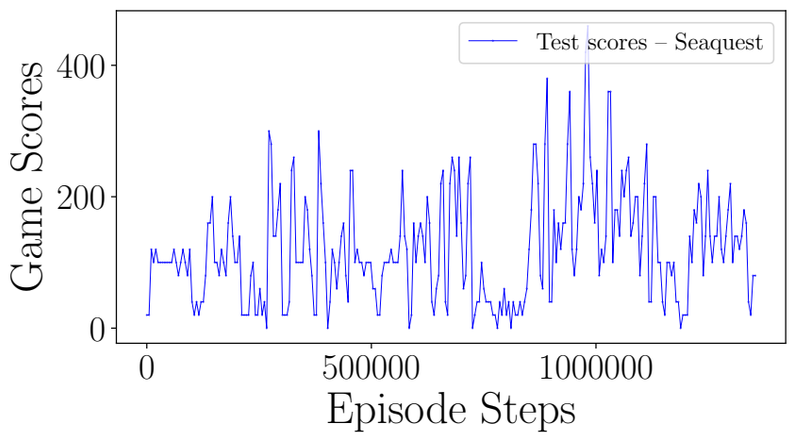
Enduro



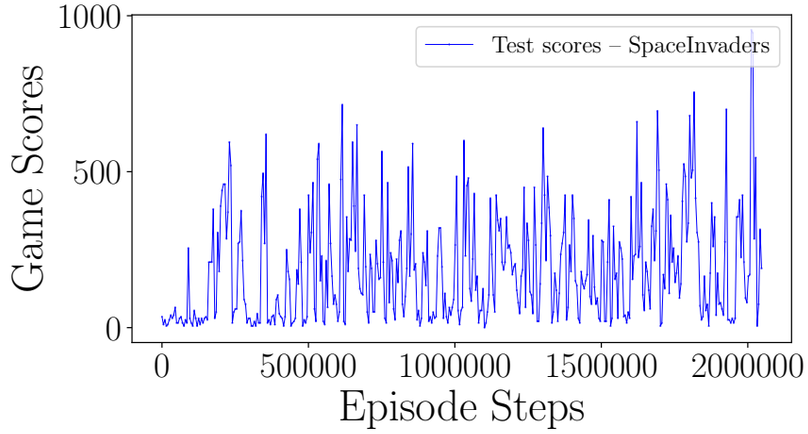
Qbert



Seaquest

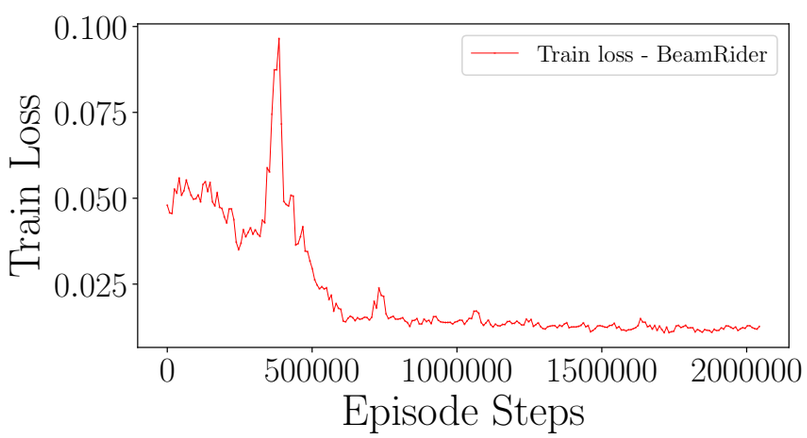


Space Invadors

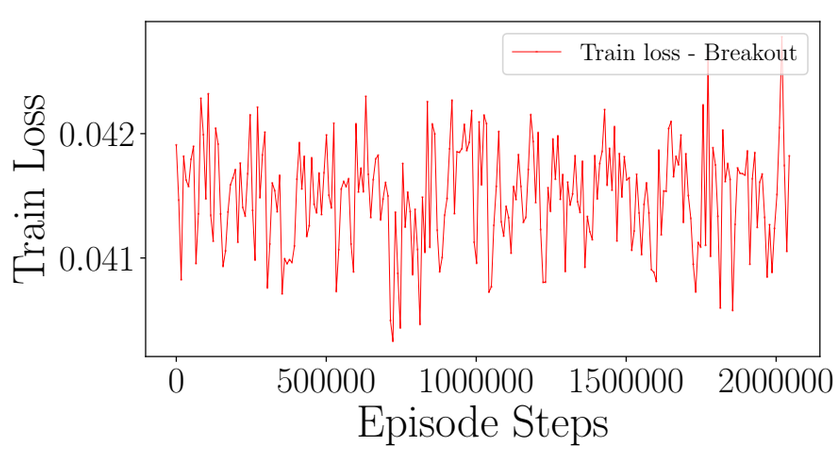


Train Loss

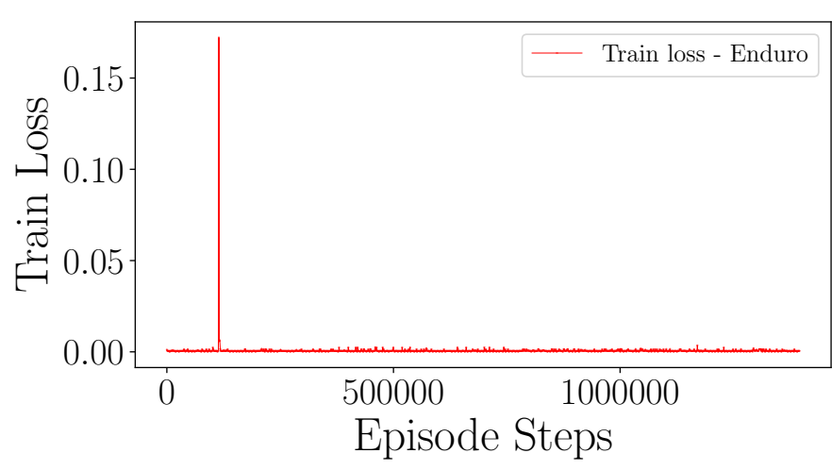
Beam Rider



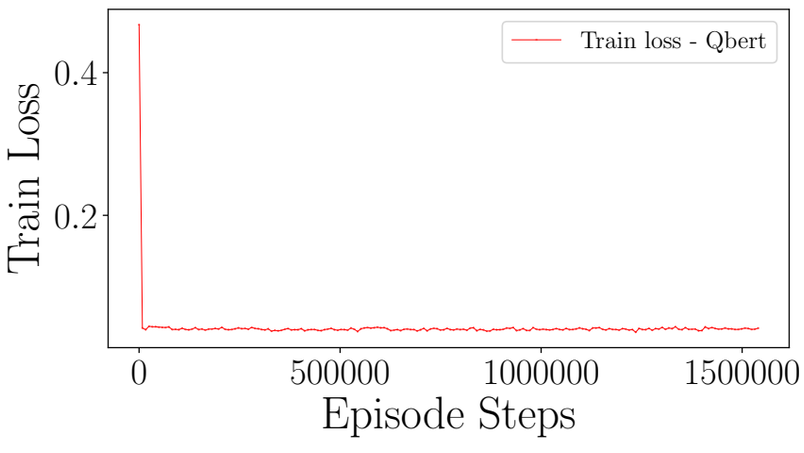
Breakout



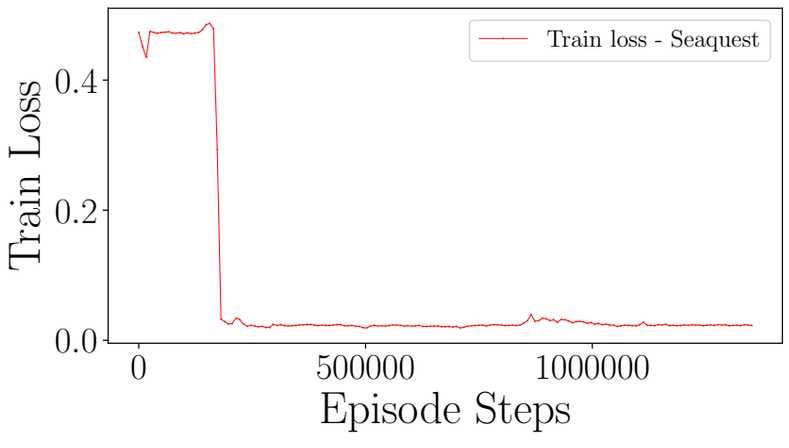
Enduro



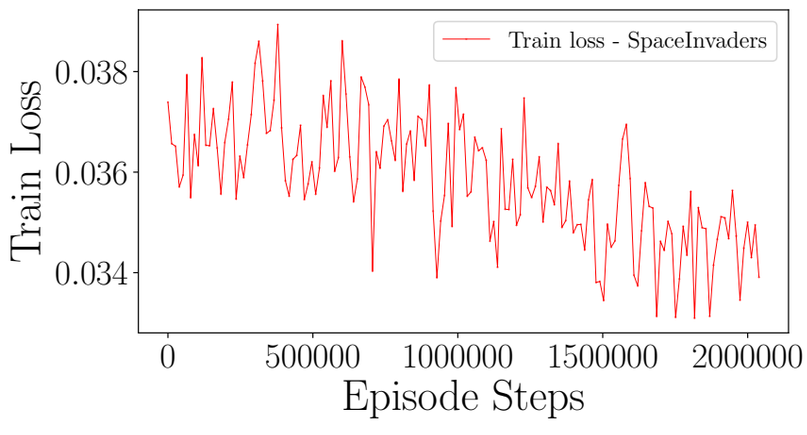
Qbert



Seaquest

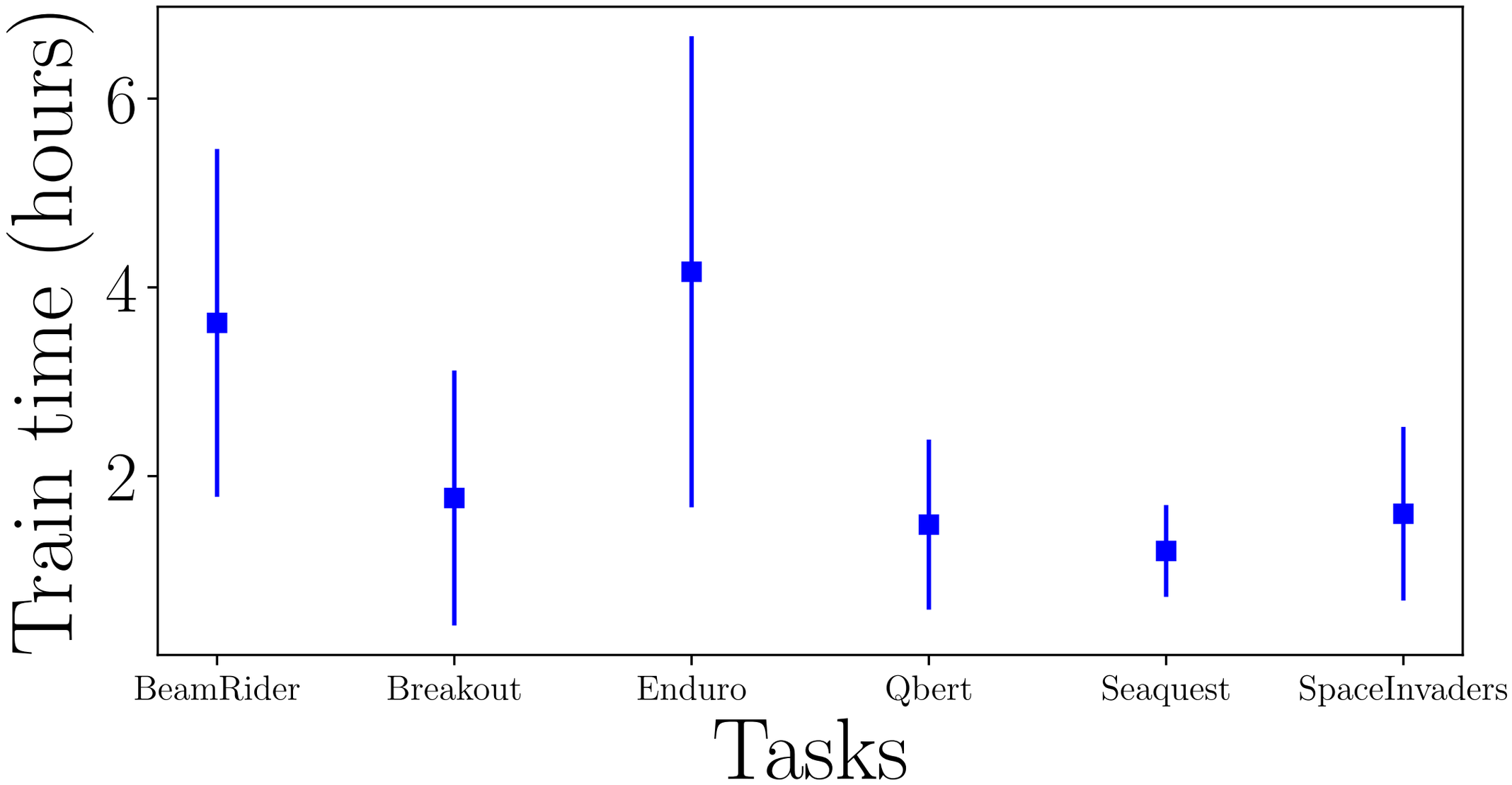


Space Invadors



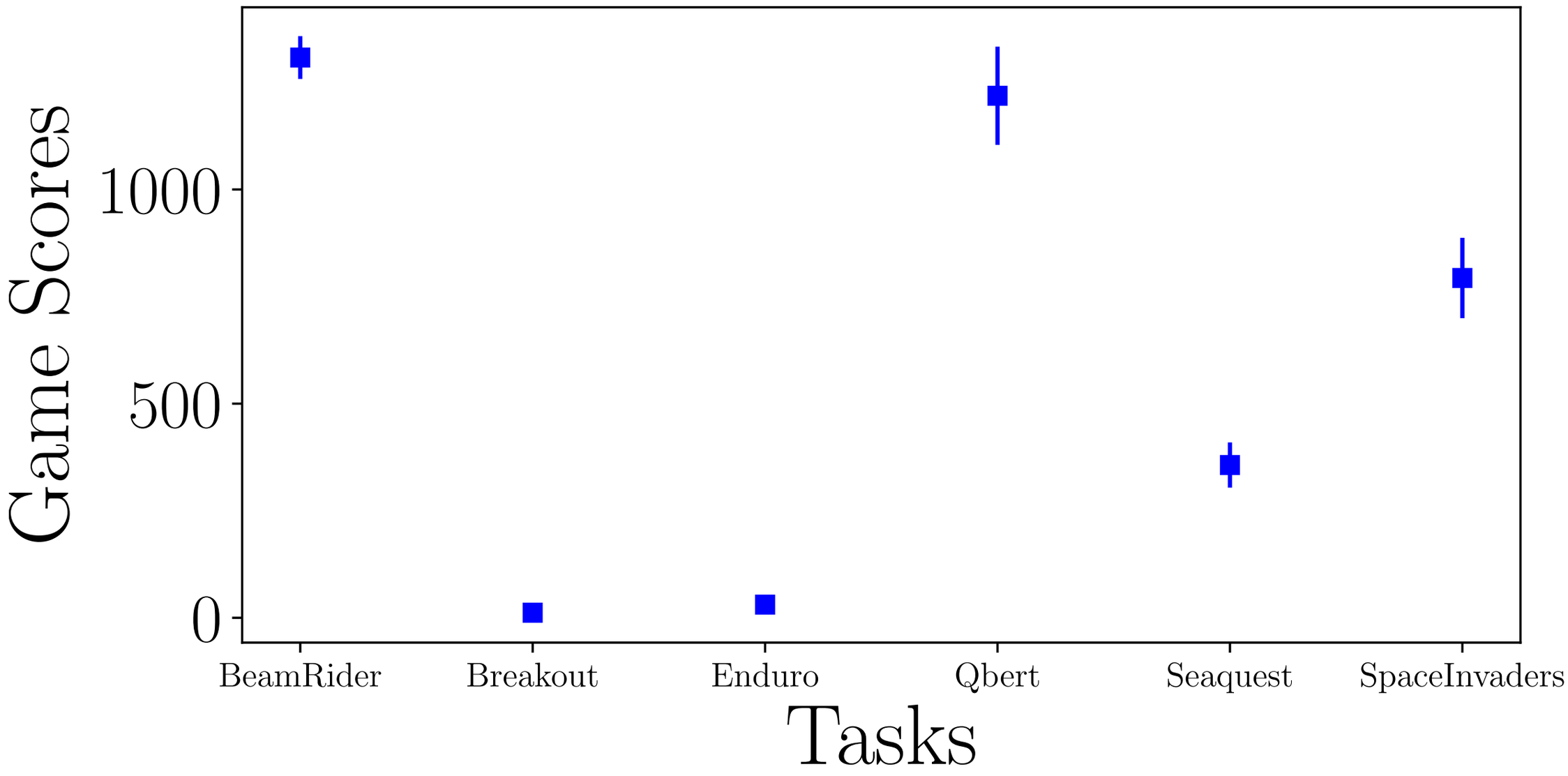
Jacob Rafati, and Roummel F. Marcia (2019). Quasi-Newton Optimization in Deep Q-Learning for Playing ATARI Games. arXiv e-print (arXiv:1811.02693).

Training Time for different batch & L-BFGS Memory Size



Jacob Rafati, and Roummel F. Marcia (2019). Quasi-Newton Optimization in Deep Q-Learning for Playing ATARI Games. arXiv e-print (arXiv:1811.02693).

Max Score for different batch & L-BFGS Memory Size



Jacob Rafati, and Roummel F. Marcia (2019). Quasi-Newton Optimization in Deep Q-Learning for Playing ATARI Games. arXiv e-print (arXiv:1811.02693).

Best Game Scores

Method	BR	BO	EO	Q*B	SQ	SI
Random	354	1.2	0	157	110	179
Human	7456	31	368	18900	28010	3690
Sarsa (Bellemare et al., 2013)	996	5.2	129	614	665	271
Contingency (Bellemare et al., 2012)	1743	6	159	960	723	268
HNeat Pixel (Hausknecht et al., 2014)	1332	4	91	1325	800	1145
DQN (Mnih et al., 2013)	4092	168	470	1952	1705	581
TRPO, Single path (Schulman et al., 2015)	1425	10	534	1973	1908	568
TRPO, Vine (Schulman et al., 2015)	859	34	431	7732	7788	450
Our method	1380	18	49	1525	600	955

Bellemare et al. (2012). Investigating contingency awareness using ATARI 2600 games. AAAI.
Bellemare et al. (2013). The arcade learning environment: An evaluation platform for general agents. Journal of Artificial Intelligence Research, 47:253–279.
Hausknecht et al. (2014). A neuroevolution approach to general ATARI game playing. IEEE Transactions on Computational Intelligence and AI in Games, 6(4):355–366.
Schulman et al. (2015). Trust region policy optimization. ICML.
Mnih, et al. (2015). Human-level control through deep reinforcement learning. Nature, 518(7540):529–533.
Jacob Rafati, and Roummel F. Marcia (2019). Quasi-Newton Optimization in Deep Q-Learning for Playing ATARI Games. arXiv e-print (arXiv:1811.02693)

Conclusions

- We proposed and demonstrated optimization methods based on the limited memory quasi-Newton method known as L-BFGS. We considered both line-search and trust-region frameworks.
- We investigated three methods for initializing L-BFGS matrices in a trust-region optimization framework in order to avoid false curvature conditions.
- We proposed and implemented a novel optimization method based on line search limited-memory BFGS for deep reinforcement learning framework.

Future Work

- What if we use full BFGS instead of limited-memory BFGS?
- The quasi-Newton methods with indefinite matrices, like Symmetric Rank 1 (SR1) or Full Broyden Class (FBC), might lead to better convergence properties, since the true Hessain is indefinite. We will study these methods, in the future.
- We will examine these optimization methods on larger machine learning problems, such as image recognition, healthcare, deep RL.
- We will study the Hessian-free optimization methods for deep learning and deep RL within conjugate gradient framework.

Publications

- Jacob Rafati, and Roummel F. Marcia (2019). Deep Reinforcement Learning via L-BFGS Optimization. arXiv e-print (arXiv:1811.02693).

- Jacob Rafati, Omar DeGuchy and Roummel F. Marcia (2018). Trust-Region Minimization Algorithm for Training Responses (TRMinATR): The Rise of Machine Learning Techniques. In 26th European Signal Processing Conference, Rome, Italy.

- Jacob Rafati, and Roummel F. Marcia (2018). Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning. In 17th IEEE International Conference on Machine Learning and Applications (ICMLA 2018), Orlando, Florida.

Acknowledgement

- Special thanks to my collaborators, Dr. Roummel F. Marcia and Omar DeGuchy.

- My research collaboration with Roummel F. Marcia is supported by NSF Grants CMMI 1333326, and IIS 1741490.

- I used MERCED clusters for some of the numerical computations, supported by the NSF Grant No. ACI-1429783.

Thank you

Papers, Code and Slides:

http://rafati.net

Supplementary Slides

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:

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

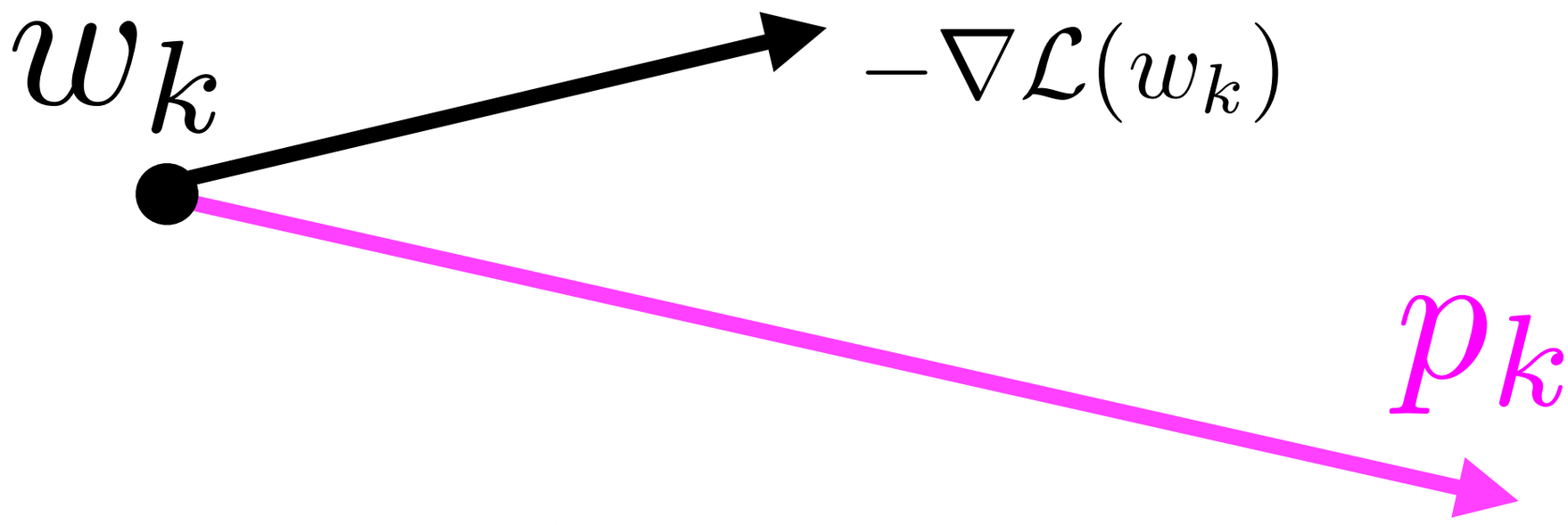
where:

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

LDU decomposition:

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

Line Search Strategy



Quadratic model of objective function

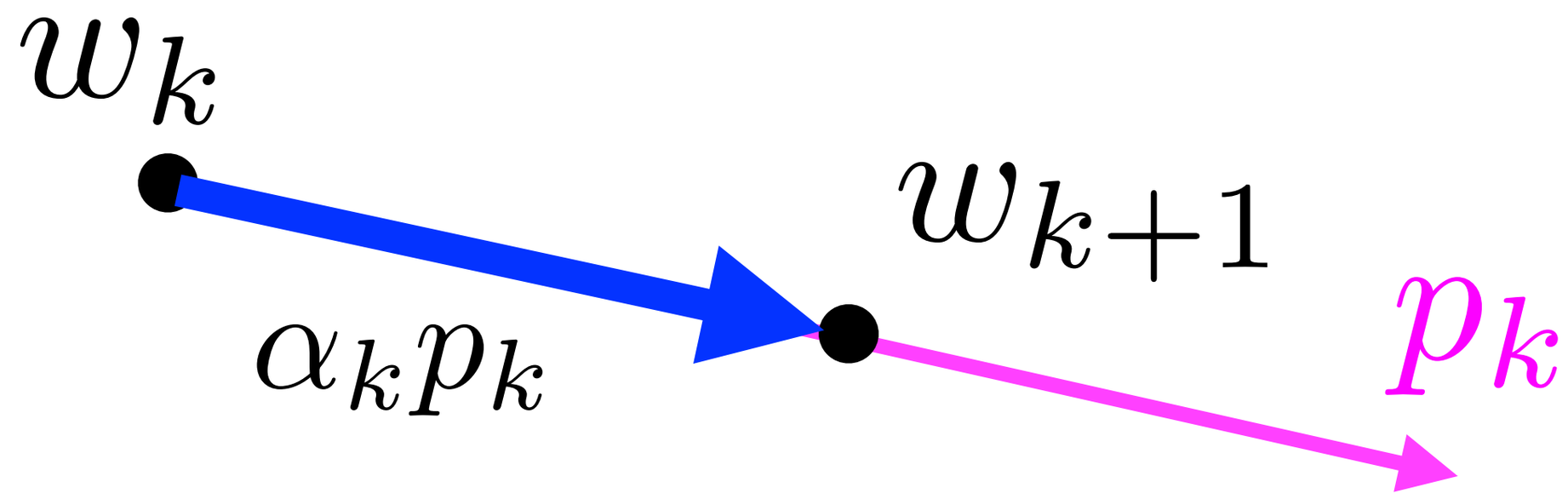
$$p_k = \underset{p \in \mathbb{R}^n}{\operatorname{argmin}} \; \mathcal{Q}_k(p) \triangleq g_k^T p + \frac{1}{2} p^T B_k p^T$$

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 Strategy



Next we should find a proper step size by solving

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

Instead we satisfy the sufficient decrease and curvature conditions

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.

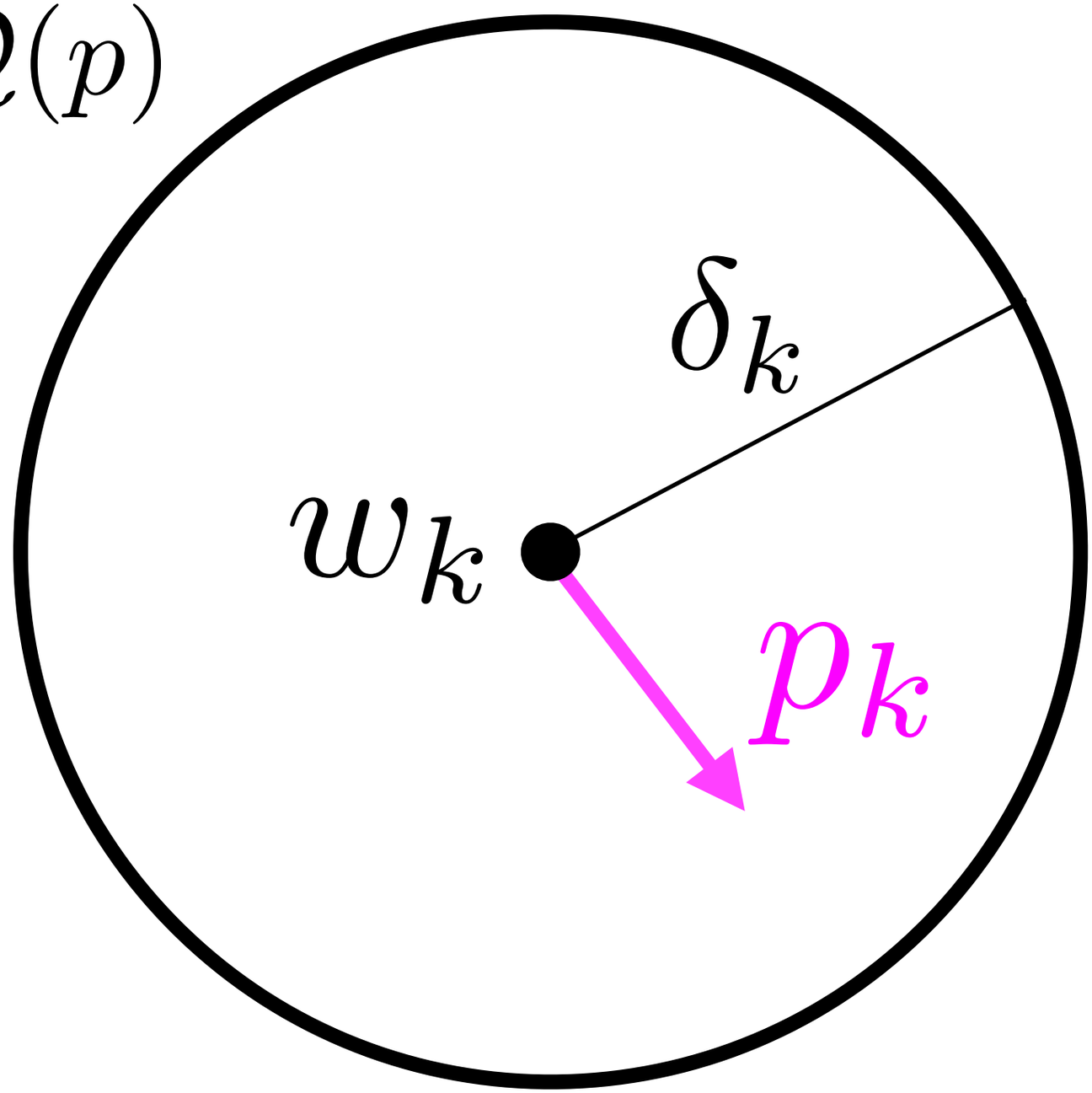
Line-search strategy

- Compute *stochastic* gradient
- Compute the quasi-Newton matrix, B_k
- Compute the search direction, $p_k = B_k^{-1} g_k$
- Compute step length using Wolfe Conditions
- Update parameters, $w_{k+1} \leftarrow w_k + \alpha_k p_k$

Trust-Region Srategy

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

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



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

Trust-Region strategy

Compute *stochastic* gradient

Compute the quasi-Newton matrix, B_k

Compute the search direction by solving TR subproblem

$$p_k = \arg\min_p \mathcal{Q}_k(p) \text{ s.t. } \|p\|_2 \leq \delta_k$$

Compute the ratio of actual reduction to prediction reduction

$$\rho_k = \frac{\mathcal{L}(w_k) - \mathcal{L}(w_k + p_k)}{-\mathcal{Q}_k(p_k)}$$

Update TR radius, δ_k

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

Optimality Conditions

Theorem.

A vector, p^*, is a global solution for trust-region subproblem,

$$\arg\min_p \mathcal{Q}(p) \text{ s.t. } \|p\|_2 \leq \delta,$$

if and only if

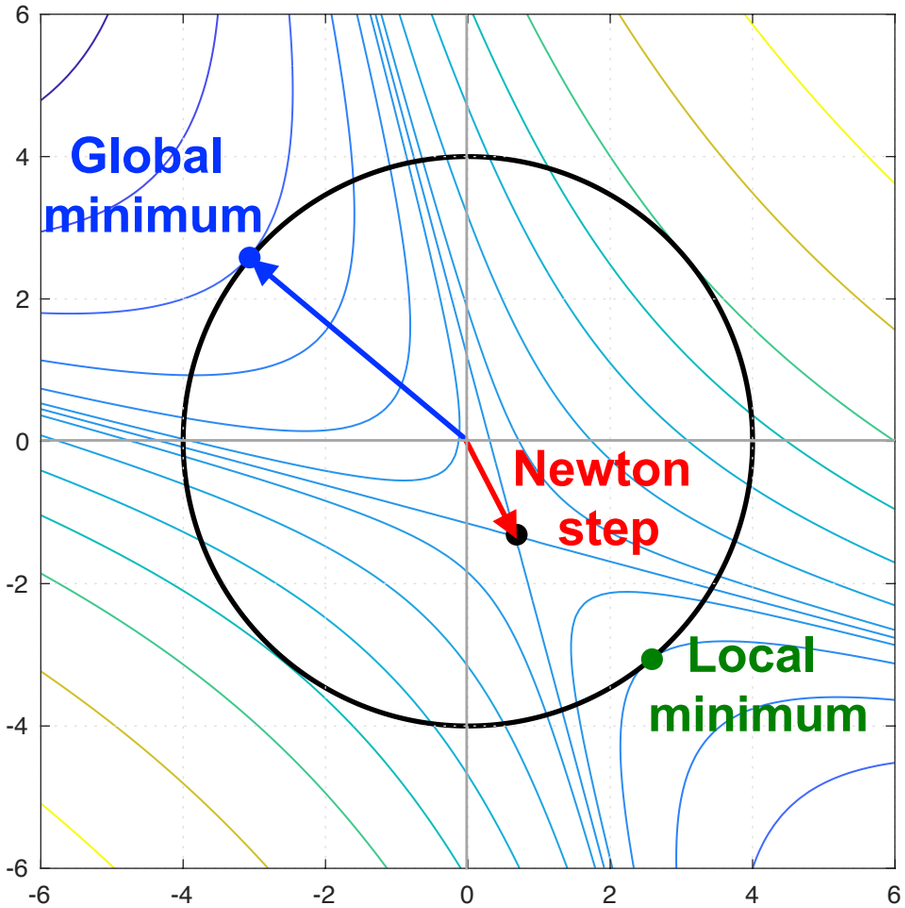
(i) $\|p^*\|_2 \leq \delta$

(ii) There exists a unique $\sigma^* \geq 0$ such that

(iii) $(B + \sigma^* I)p^* = -g$, and

(iv) $\sigma^*(\delta - \|p^*\|_2) = 0$

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.

Solving Trust-region subproblem

Eigen-decomposition of $B_k = B_0 + \Psi_k M_k \Psi_k^T$

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

Sherman-Morrison-Woodbury Formula gives a closed form solution to optimal search direction

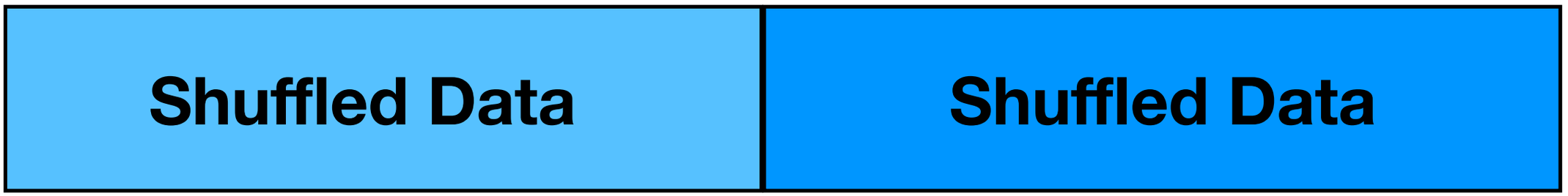
$$p_k^* = -\frac{1}{\tau^*} \left[I - \Psi_k (\tau^* M_k^{-1} + \Psi_k^T \Psi_k)^{-1} \Psi_k^T \right] g_k,$$

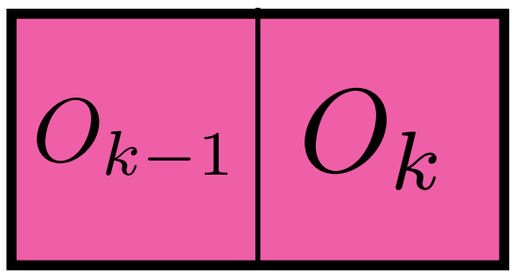
is Lagrange multiplier, and there are fast and accurate methods to find it. (See Brust et al (2017).)

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

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.

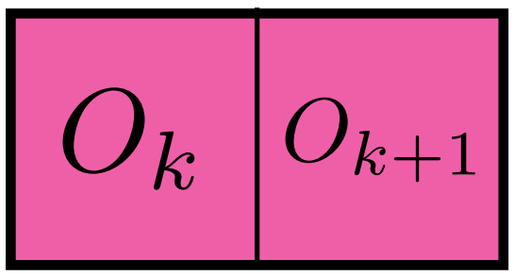
Multi-Batch L-BFGS



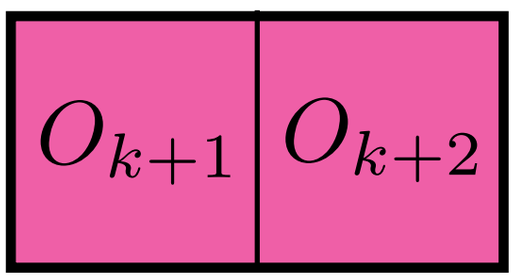


$\mathcal{S}_k$

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



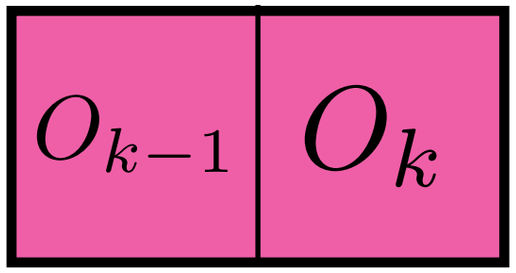
$\mathcal{S}_{k+1}$



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

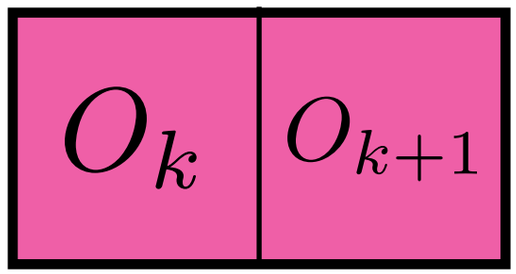
Berahas et al. (2016). “A multi-batch L-BFGS method for machine learning,” in Advances in Neural Information Processing Systems 29, pp. 1055–1063.

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.

Initialization Method I

Spectral estimate of Hessian

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

By solving this simple problem:

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

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

Initialization Method II

For a quadratic function,

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

We have

$$\nabla^2 \mathcal{L}(w) = H$$

Therefore Hessian satisfies the Secant Equations

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

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

Initialization Method II

Since the quasi-Newton matrices

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

should satisfy the Secant Equation

$$B_k S_k^T = Y_k$$

We should satisfy a curvature condition in order to avoid introducing fasle curvature conditions.

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

Jacob Rafati, and Roummel F. Marcia (2018). Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning. (ICMLA 2018), Orlando, Florida.

Initialization Method II

We can define an upper bound by solving a general Eigen-value problem

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

An upper bound for initial value to avoid false curvature information

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

Jacob Rafati, and Roummel F. Marcia (2018). Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning. (ICMLA 2018), Orlando, Florida.

Initialization Method III

We should satisfy a curvature condition in order to avoid introducing fasle curvature conditions.

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

Since the L-BFGS compact matrices contains γ_k

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

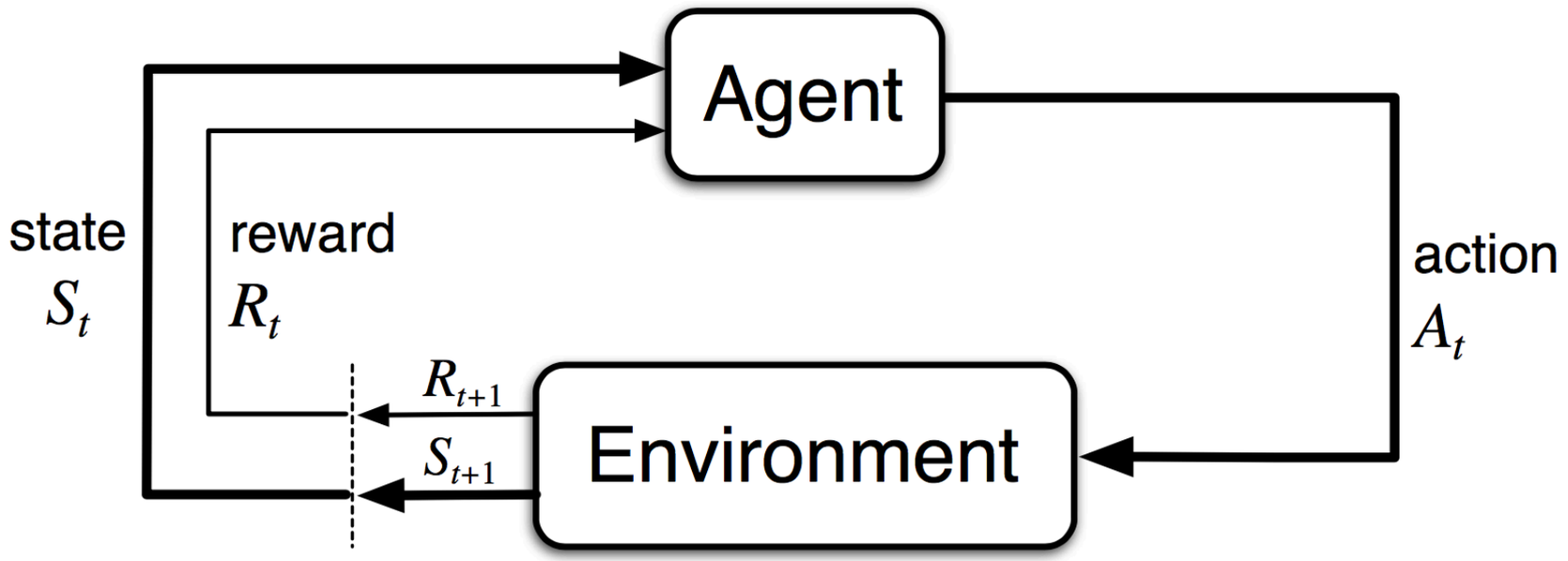
We can improve the upper-bound by solving

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

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

Jacob Rafati, and Roummel F. Marcia (2018). Improving L-BFGS Initialization For Trust-Region Methods In Deep Learning. (ICMLA 2018), Orlando, Florida.

Reinforcement Learning



Reinforcement learning (RL) is learning how to map situations (**states**) to decisions (**actions**) to **maximize future rewards.**

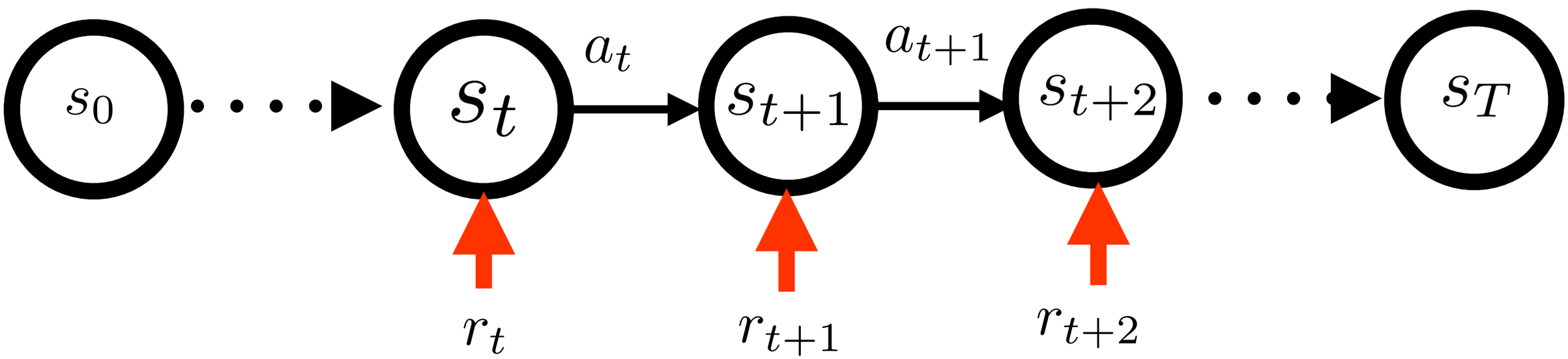
Sutton and Barto (2017). Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, USA, 2nd edition.

Return

Return is the cumulative sum of a future rewards:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$$

$\gamma \in (0, 1]$ is a discount factor

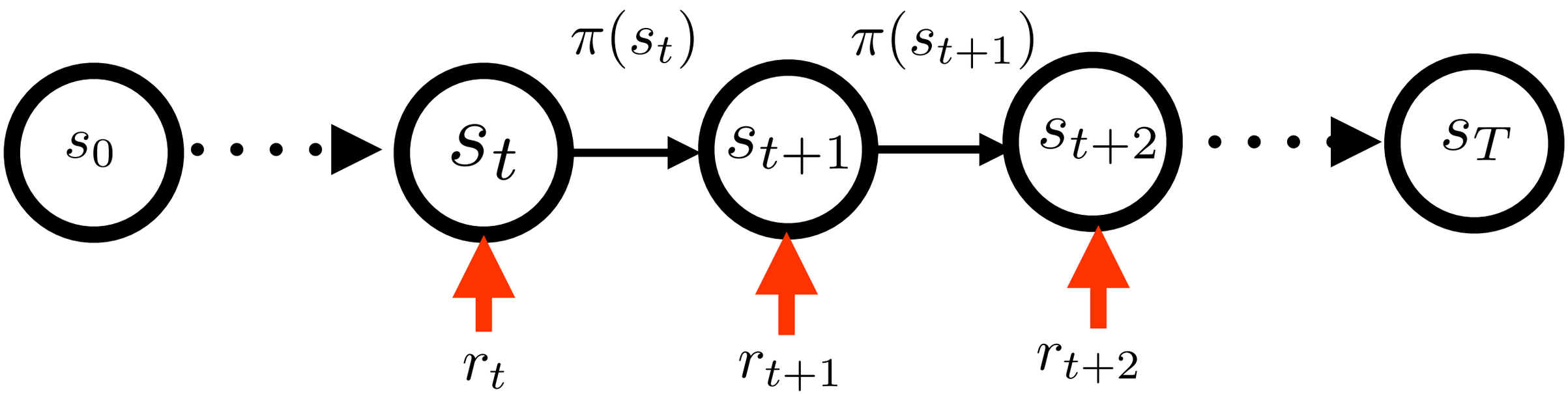


Policy Function

- Policy Function: At each time step agent implements a mapping from states to possible actions

$$\pi : \mathcal{S} \rightarrow \mathcal{A}$$

$$a_t = \pi(s_t)$$



Q-Function

- State-Action Value Function (Q-Function) is the expected return when starting from (*s,a)* and following *a policy* thereafter

$$q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$$

$$q_\pi(s, a) = \mathbb{E}_\pi \left[G_t \mid S_t = s, A_t = a \right]$$

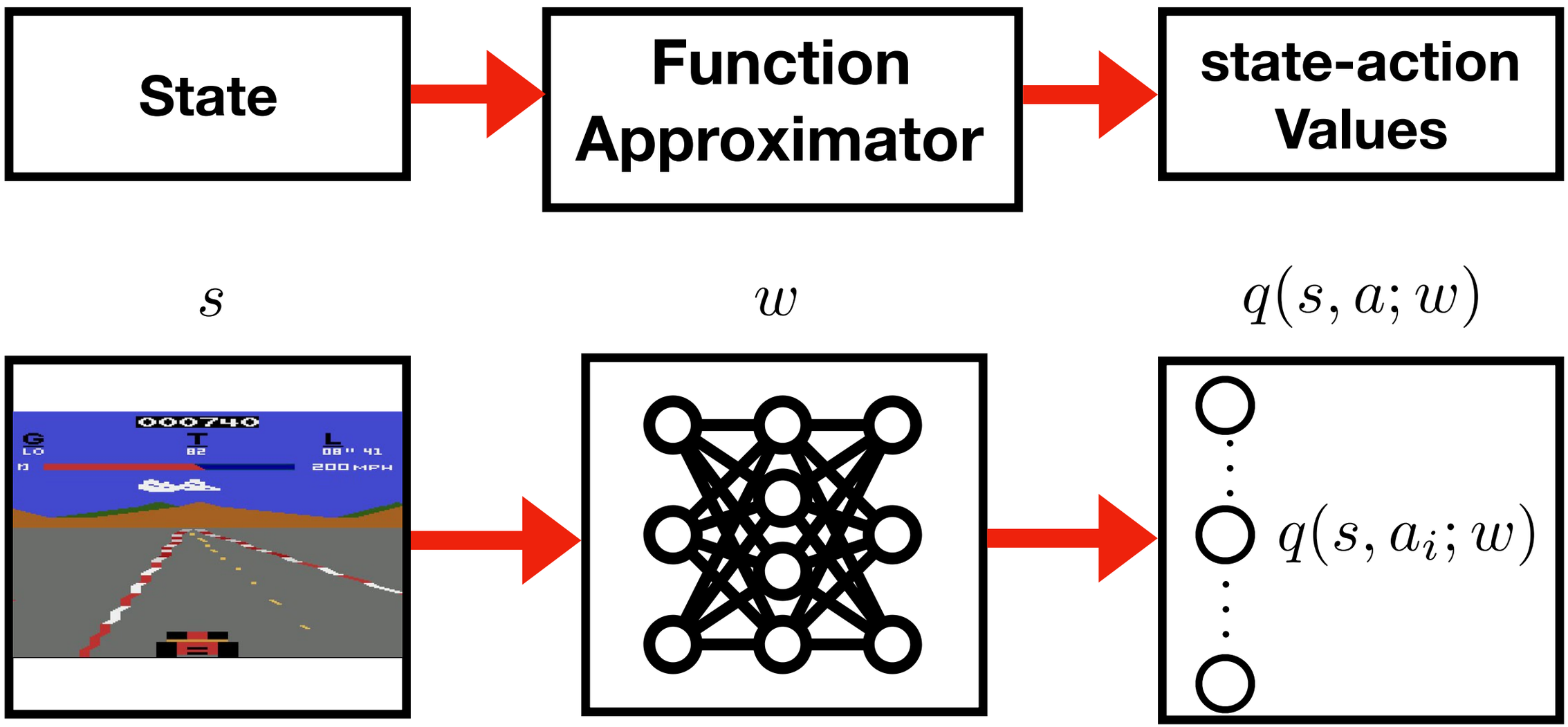
- The optimal *Q*-Function is the maximum expected return.

$$q^*(s, a) = \max_\pi q_\pi(s, a)$$

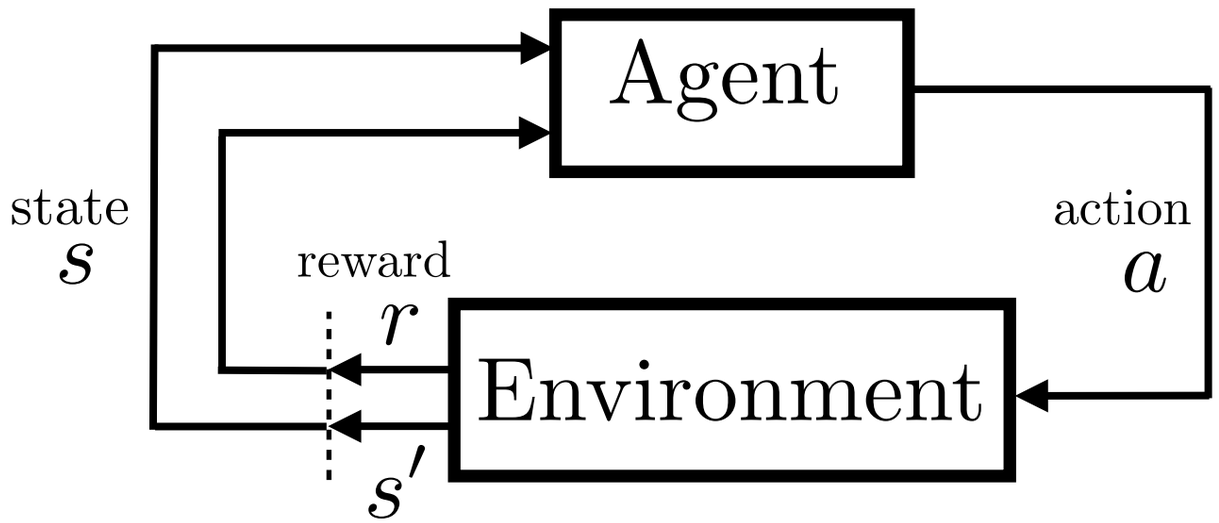
$$\pi^*(s) = \arg\max_a q^*(s, a)$$

Generalization

$$Q(s, a) \approx q(s, a; w)$$



Empirical Risk Minimization in Deep RL



$$\min_{w \in \mathbb{R}} \mathcal{L}(w) \triangleq \frac{1}{N} \sum_{e \in \mathcal{D}} \left(r + \gamma \max_{a'} q(s', a'; w) - q(s, a; w) \right)^2$$

$\mathcal{D} = \{(s, a, s', r)\}$ is Agent's Experiences Memory.

Sutton and Barto (2017). Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, USA, 2nd edition.