

Stochastic Quasi-Newton Methods for Training Neural Networks

Indrapriyadarsini Sendilkkumar

Nishimura Laboratory
Student ID: 5594 5032

Doctoral Dissertation Review
4th August 2022

OUTLINE

Introduction

- Optimization
- Motivation and Objective

Background

- Gradient Based Neural Network Training
 - First Order Methods
 - Second Order Methods

Stochastic accelerated quasi-Newton Methods

- Proposed methods
- Results
- Convergence Analysis

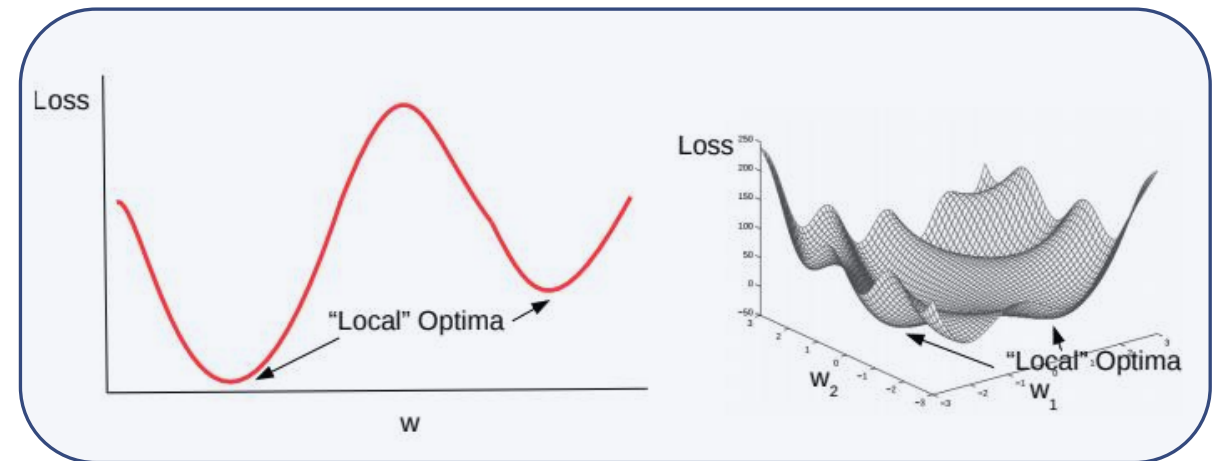
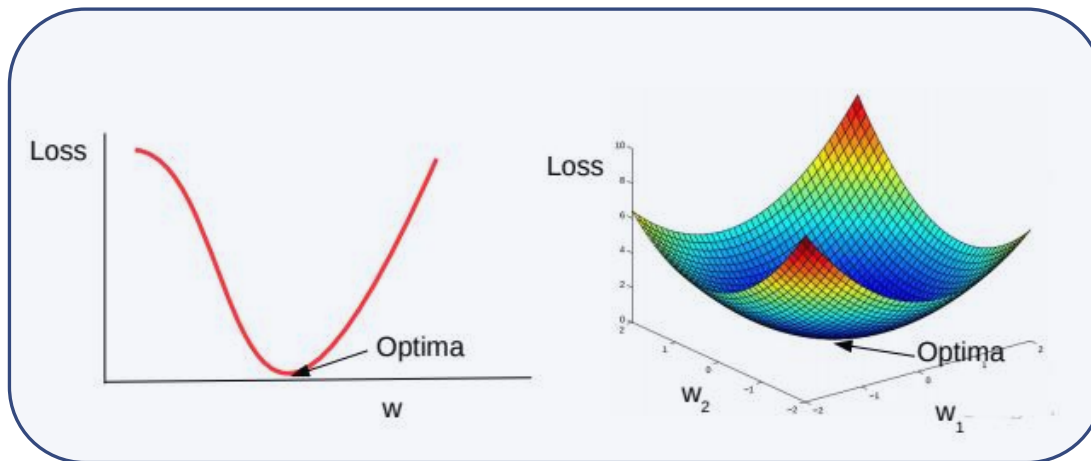
INTRODUCTION

- What is **optimization**?

Given $f(\mathbf{x})$, find \mathbf{x} that minimizes $f(\mathbf{x})$.

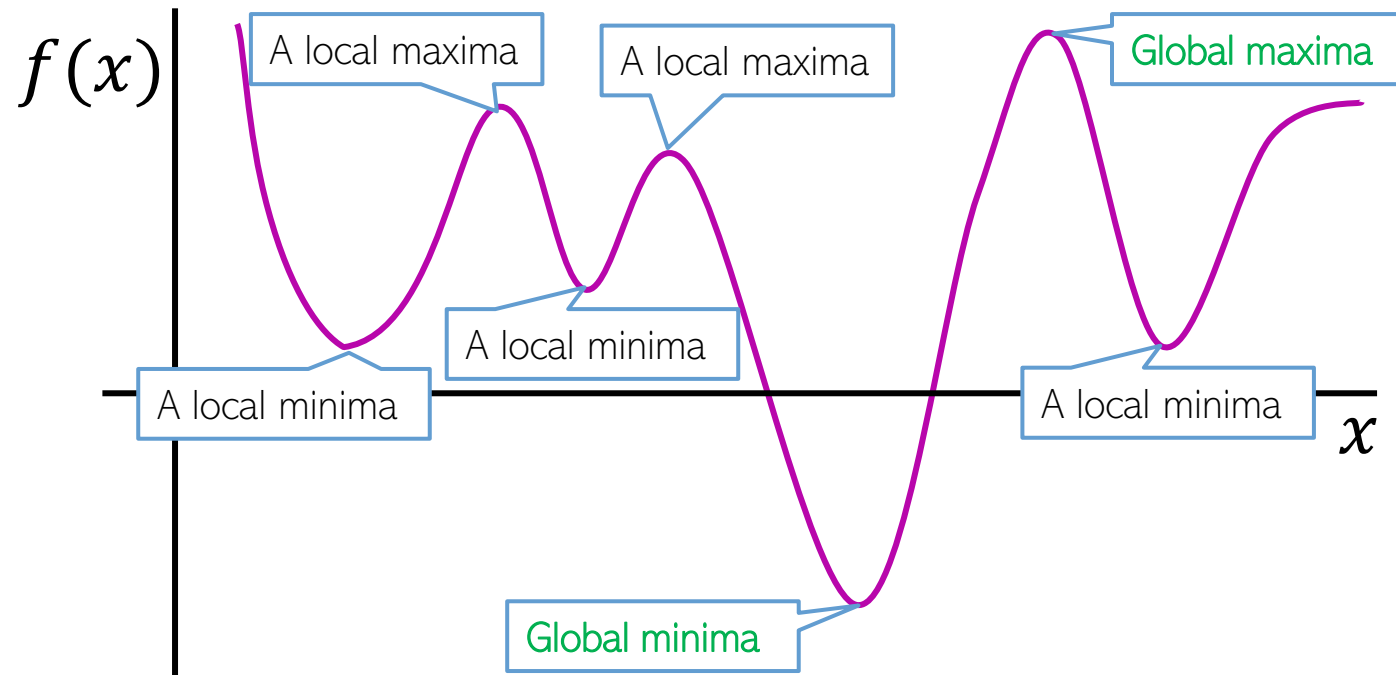
This is a key fundamental problem with broad applications across different areas

- A function being optimized can be either **convex** or **non-convex**



INTRODUCTION

- Many Machine Learning problems require to optimize a function f of some variable(s) x
- For simplicity, assume f is a scalar-valued function of a scalar x ($f: \mathbb{R} \rightarrow \mathbb{R}$)



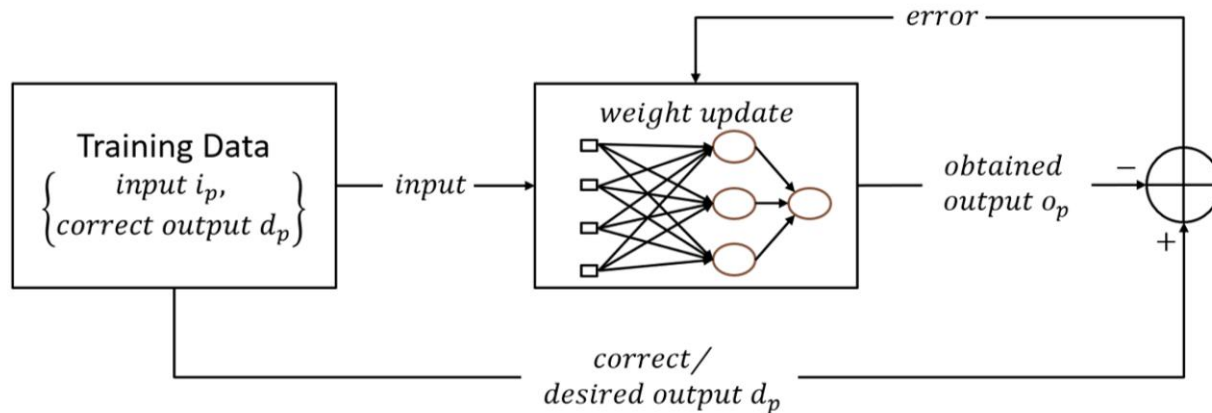
Usually interested in global optima but often want to find local optima, too

For deep learning models, often the local optima are what we can find (and they usually suffice)

- Functions can have one or more optima (maxima, minima), and maybe saddle points
- Finding the optima or saddles requires derivatives/gradients of the function

OPTIMIZATION IN SUPERVISED LEARNING

- Given a dataset $(i_p, d_p)_{p \in T_r}$
- Neural network : Parameterized model to map function $E_p(\mathbf{w}; i_p, d_p)$ $\mathbf{w} \in \mathbb{R}^d$



- Objective function

$$\min_{\mathbf{w} \in \mathbb{R}^d} E(\mathbf{w}) = \frac{1}{|T_r|} \sum_{p \in T_r} E_p(\mathbf{w})$$

Loss function

Mean Squared Error
(regression problems)

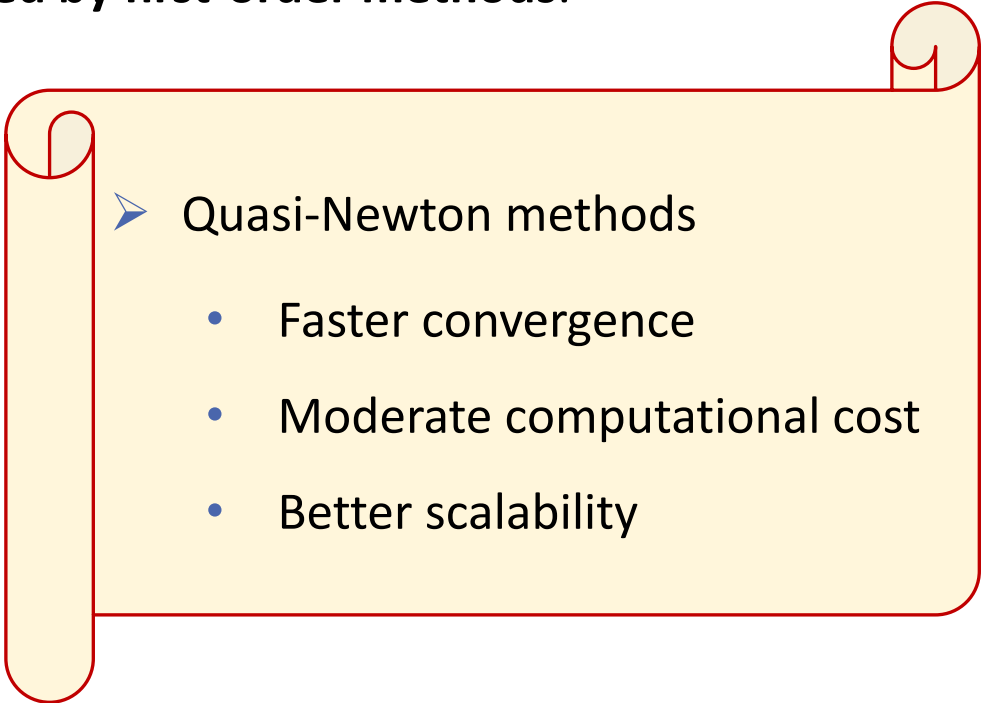
$$E_p(\mathbf{w}) = \frac{1}{2} \|d_p - o_p\|^2$$

Cross Entropy Error
(classification problems)

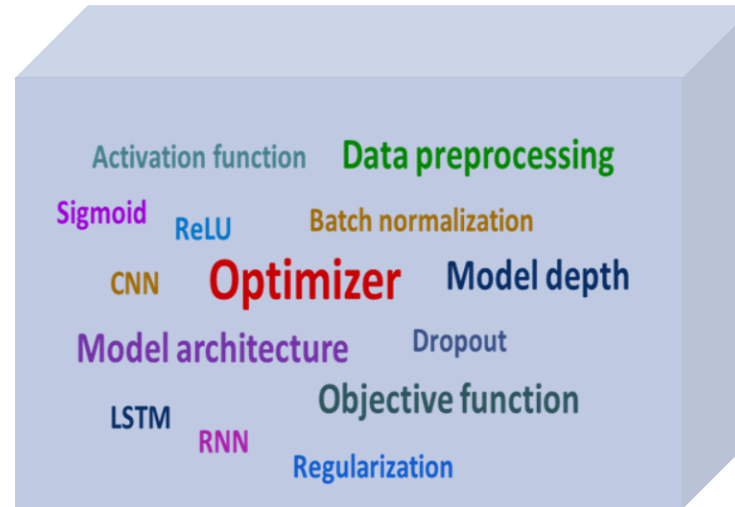
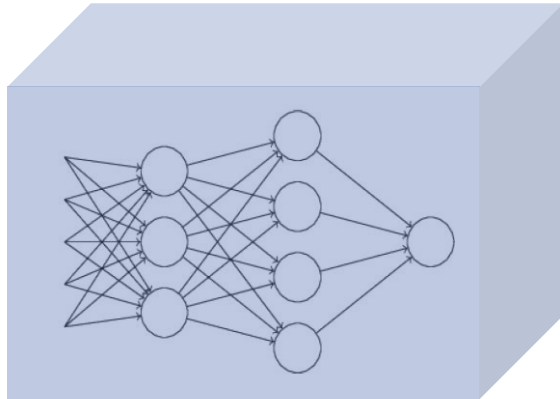
$$E_p(\mathbf{w}) = - \sum_{c=1}^M d_p \log o_p$$

MOTIVATION

- Classical machine learning models are either convex or can be reduced to a convex optimization problems solved with conventional mathematical optimization methods.
- Optimization in machine learning is presently **dominated by first-order methods**.
- First order methods exhibit
 - slow convergence
 - high sensitivity to hyperparameter settings
 - stagnation at high training errors, and
 - difficulty escaping flat regions and saddle points.

- 
- Quasi-Newton methods
 - Faster convergence
 - Moderate computational cost
 - Better scalability

OBJECTIVE AND SCOPE



- Study the behavior of first and second order methods in training neural networks
- Investigate the feasibility of Nesterov's acceleration on algorithms in the quasi-Newton family.
- Devise robust and efficient accelerated second-order optimizers suitable for stochastic training.
- Analyze computational cost and convergence guarantees.

OUTLINE

Introduction

Background

- Gradient Based Neural Network Training
 - First Order Methods
 - Second Order Methods
 - Quasi-Newton Method
 - Accelerated Quasi-Newton
 - Simulation Examples (Full Batch)

Stochastic accelerated quasi-Newton Methods

GRADIENT BASED ALGORITHMS

FIRST ORDER METHODS

- Slow convergence in highly non-linear problems
- Simple and low complexity

$$\mathbf{w} := \mathbf{w} - \alpha \frac{\partial E}{\partial \mathbf{w}} \quad \searrow \text{Gradient } \nabla E(\mathbf{w})$$

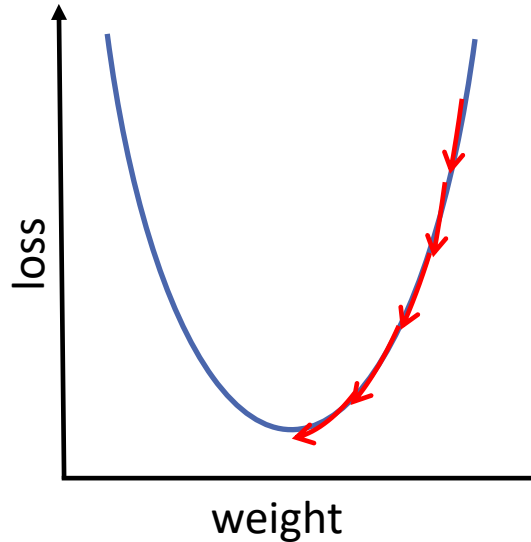
Classical Momentum
Nesterov's Accelerated Gradient (NAG)
AdaGrad, RMSProp, Adam

SECOND/APPROXIMATED SECOND ORDER METHODS

- Faster convergence
- Suitable for highly non-linear problems
- High computational cost

$$\mathbf{w} := \mathbf{w} - \alpha \mathbf{H} \nabla E(\mathbf{w}) \quad \rightarrow \text{Hessian}$$

Newton Method
Quasi-Newton Method (QN)
Nesterov's Accelerated quasi-Newton (NAQ)



FIRST ORDER ALGORITHMS

The weight vector is updated by the update vector \mathbf{v}_{k+1} as

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1} \quad \dots (\text{Eq. 1})$$

Steepest gradient descent(SGD) with a step size α_k

$$\mathbf{v}_{k+1} = -\alpha_k \nabla E(\mathbf{w}_k) \quad \dots (\text{Eq. 2})$$

Normal Gradient

Classical momentum (CM) method

$$\mathbf{v}_{k+1} = \mu \mathbf{v}_k - \alpha_k \nabla E(\mathbf{w}_k) \quad \dots (\text{Eq. 3})$$

Momentum term

Nesterov's Accelerated Gradient (NAG) method

$$\mathbf{v}_{k+1} = \mu \mathbf{v}_k - \alpha_k \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k) \quad \dots (\text{Eq. 4})$$

Momentum term

+

Nesterov's Accelerated Gradient (NAG)

QUASI-NEWTON METHOD

The weight is updated with update vector \mathbf{v}_{k+1} as:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1} \quad \dots (Eq. 5)$$

The weight update of quasi-Newton (QN) method is given as

$$\mathbf{v}_{k+1} = -\alpha_k \mathbf{H}_k \nabla E(\mathbf{w}_k) \quad \dots (Eq. 6)$$

Normal Gradient

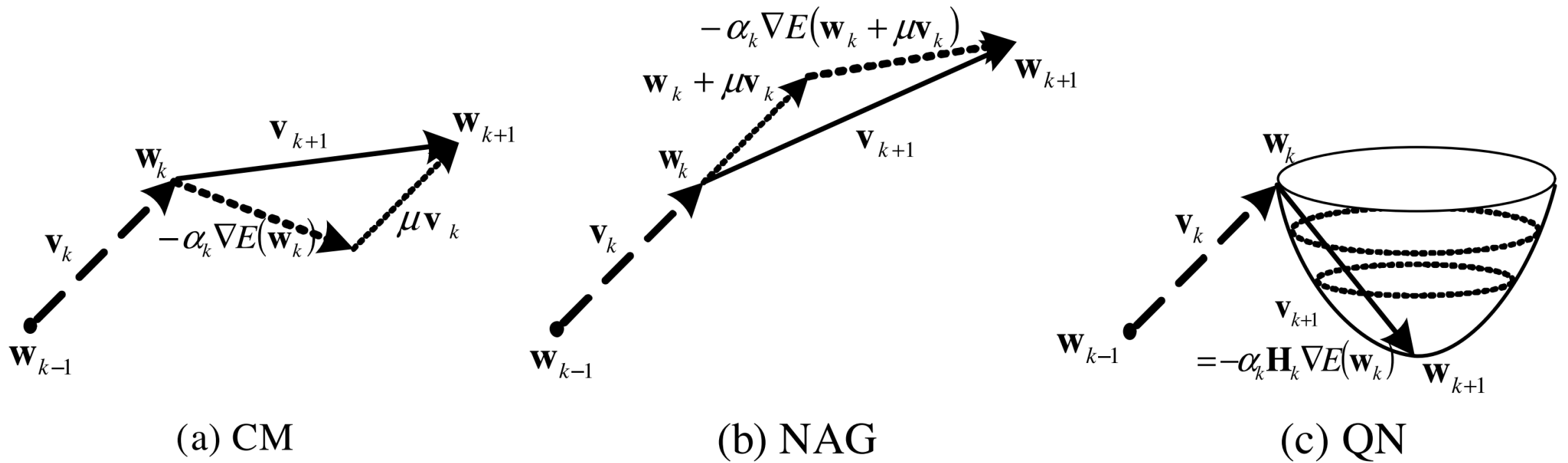
The matrix \mathbf{H}_k is iteratively approximated by BFGS formula

$$\mathbf{H}_{k+1} = (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T) \mathbf{H}_k (\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T) + \rho_k \mathbf{s}_k \mathbf{s}_k^T \quad \dots (Eq. 7)$$

$$\rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}, \quad \mathbf{s}_k = \mathbf{w}_{k+1} - \mathbf{w}_k \quad \text{and} \quad \mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k) \quad \dots (Eq. 8)$$

Normal Gradients

GEOMETRIC VIEWS



Source: H. Ninomiya, "A novel quasi-Newton-Optimization for neural network training incorporating Nesterov's accelerated gradient", IEICE NOLTA Journal, Oct. 2017.

NESTEROV'S ACCELERATED QUASI-NEWTON METHOD (NAQ)

The update vector of NAQ

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu \mathbf{v}_k - \alpha_k \mathbf{H}_k \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k) \quad \dots (\text{Eq. 9})$$

Momentum term

Nesterov's Accelerated Gradient(NAG)

The matrix \mathbf{H}_k is iteratively approximated by

$$\mathbf{H}_{k+1} = (\mathbf{I} - \rho_k \mathbf{p}_k \mathbf{q}_k^T) \mathbf{H}_k (\mathbf{I} - \rho_k \mathbf{q}_k \mathbf{p}_k^T) + \rho_k \mathbf{p}_k \mathbf{p}_k^T \quad \dots (\text{Eq. 10})$$

$$\rho_k = \frac{1}{\mathbf{q}_k^T \mathbf{p}_k}, \quad \mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k) \quad \text{and} \quad \mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$$

Two gradient computations per iteration

Normal Gradient

Nesterov's Accelerated Gradient(NAG)

COMPARISON OF BFGS AND NAQ ALGORITHM

BFGS ALGORITHM

Require k_{max}

Initialize $\mathbf{w}_k \in \mathbb{R}^d$, $\mathbf{H}_k = \epsilon \mathbf{I}$, $\mathbf{v}_k = 0$ and $k \leftarrow 1$

Calculate $\nabla E(\mathbf{w}_k)$

while ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**

1. $\mathbf{g}_k \leftarrow -\mathbf{H}_k \nabla E(\mathbf{w}_k)$
2. Determine α_k using Armijo linesearch
3. $\mathbf{v}_{k+1} = \alpha_k \mathbf{g}_k$
4. $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$
5. Calculate $\nabla E(\mathbf{w}_{k+1})$
6. $\mathbf{p}_k \leftarrow \mathbf{w}_{k+1} - \mathbf{w}_k$
7. $\mathbf{q}_k \leftarrow \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k)$
8. Update \mathbf{H}_{k+1}
9. $k \leftarrow k + 1$

end while

NAQ ALGORITHM

Require $0 < \mu < 1$, k_{max}

Initialize $\mathbf{w}_k \in \mathbb{R}^d$, $\mathbf{H}_k = \epsilon \mathbf{I}$, $\mathbf{v}_k = 0$ and $k \leftarrow 1$

while ($\|\nabla E(\mathbf{w}_k)\| > \epsilon$ and $k < k_{max}$) **do**

1. Calculate $\nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$
2. $\mathbf{g}_k \leftarrow -\mathbf{H}_k \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$
3. Determine α_k using Armijo linesearch
4. $\mathbf{v}_{k+1} = \mu \mathbf{v}_k + \alpha_k \mathbf{g}_k$
5. $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$
6. Calculate $\nabla E_2 \leftarrow \nabla E(\mathbf{w}_{k+1})$
7. $\mathbf{p}_k \leftarrow \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k)$
8. $\mathbf{q}_k \leftarrow \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$
9. Update \mathbf{H}_{k+1}
10. $k \leftarrow k + 1$

end while

MOMENTUM QUASI-NEWTON METHOD (MoQ)

The update vector of NAQ

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu \mathbf{v}_k - \alpha_k \mathbf{H}_k \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k) \quad \dots (Eq. 11)$$

Momentum term

Nesterov's Accelerated Gradient(NAG)

Nesterov's accelerated gradient approximation

$$\nabla E(\mathbf{w}_k + \mu \mathbf{v}_k) \approx (1 + \mu_k) \nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1}) \quad \dots (Eq. 12)$$

and the Hessian matrix \mathbf{H}_k is updated as

$$\mathbf{H}_{k+1} = (\mathbf{I} - \rho_k \mathbf{p}_k \mathbf{q}_k^T) \mathbf{H}_k (\mathbf{I} - \rho_k \mathbf{q}_k \mathbf{p}_k^T) + \rho_k \mathbf{p}_k \mathbf{p}_k^T \quad \dots (Eq. 13)$$

$$\rho_k = \frac{1}{\mathbf{q}_k^T \mathbf{p}_k}, \quad \mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k) \quad \text{and} \quad \mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \{(1 + \mu_k) \nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1})\}$$

Shahzad Mahboubi, S. Indrapriyadarsini, Hiroshi Ninomiya, Hideki Asai, "Momentum acceleration of quasi-Newton Training for Neural Networks", 16th Pacific Rim International Conference on Artificial Intelligence, PRICAI 2019, (pp. 268-281). Springer, Cham.

BEALE FUNCTION

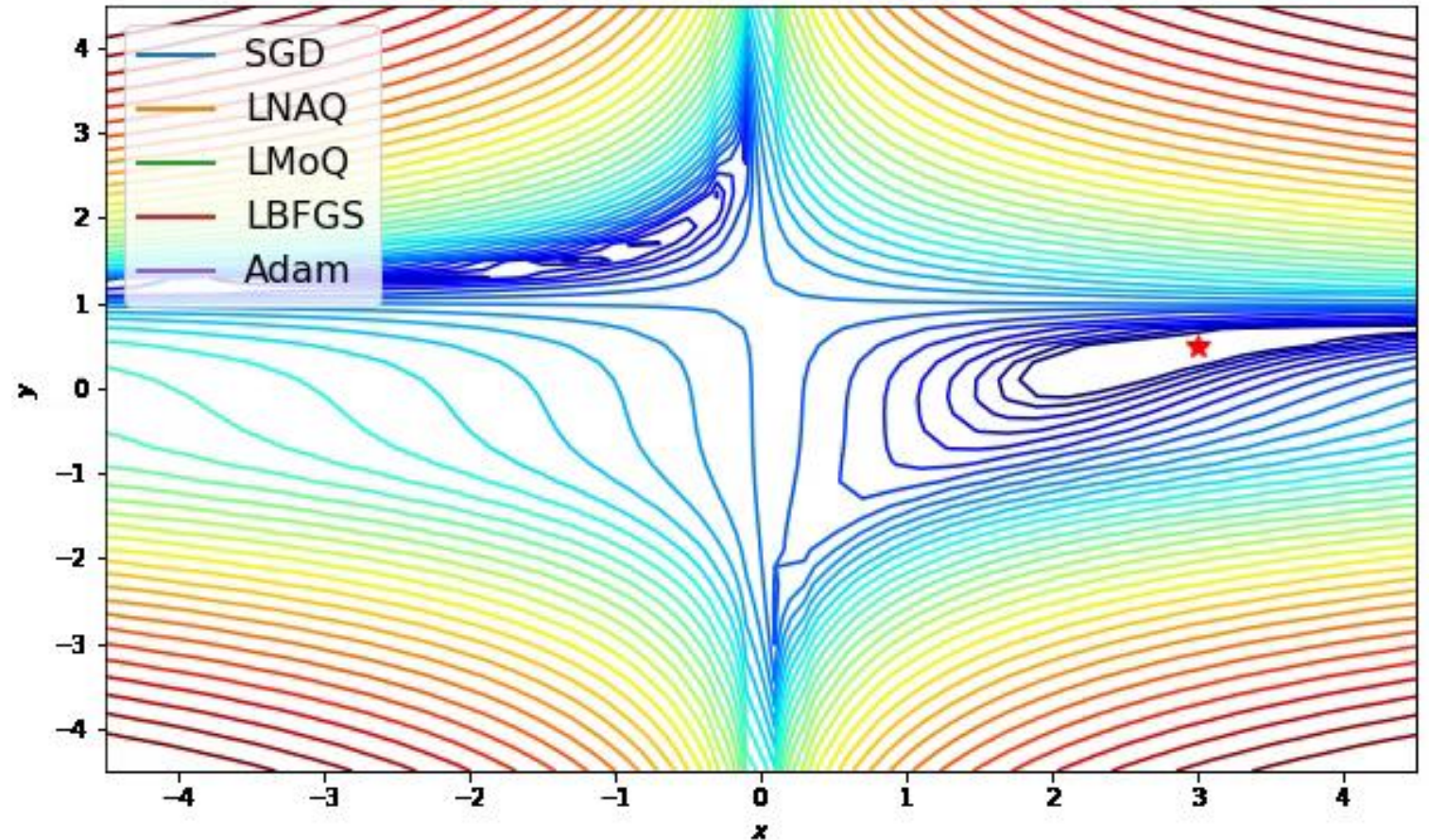
The Beale function is multimodal,
with sharp peaks at the corners of
the input domain

Unconstrained test function

$$f(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$$

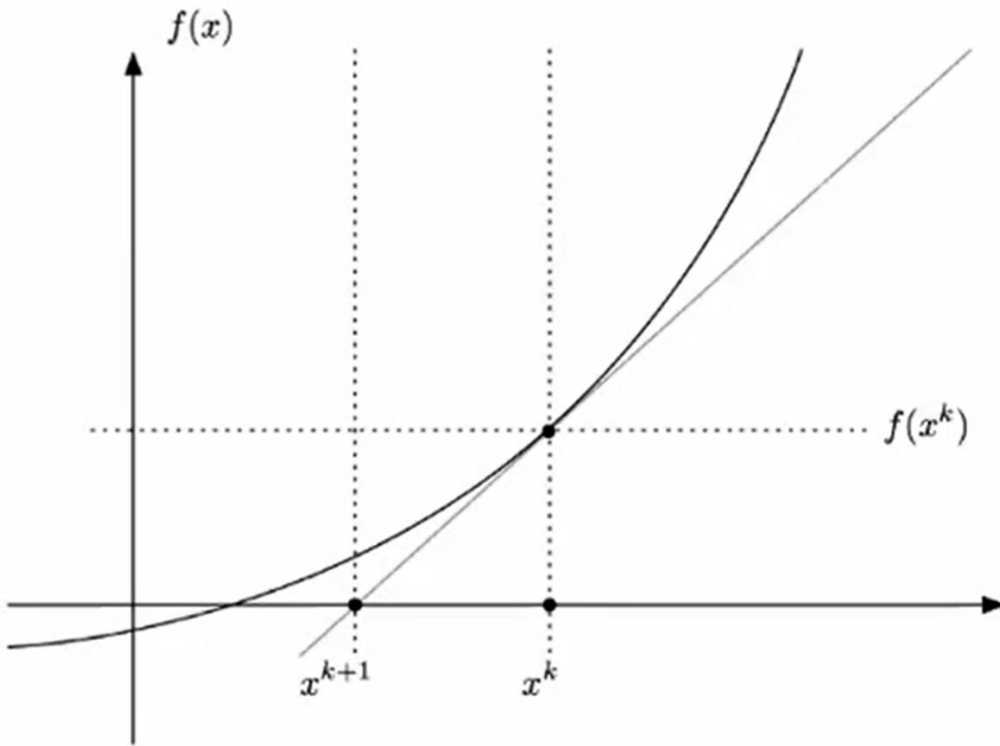
Global minimum

$$f(x^*) = 0 \text{ at } x^* = (3, 0.5)$$

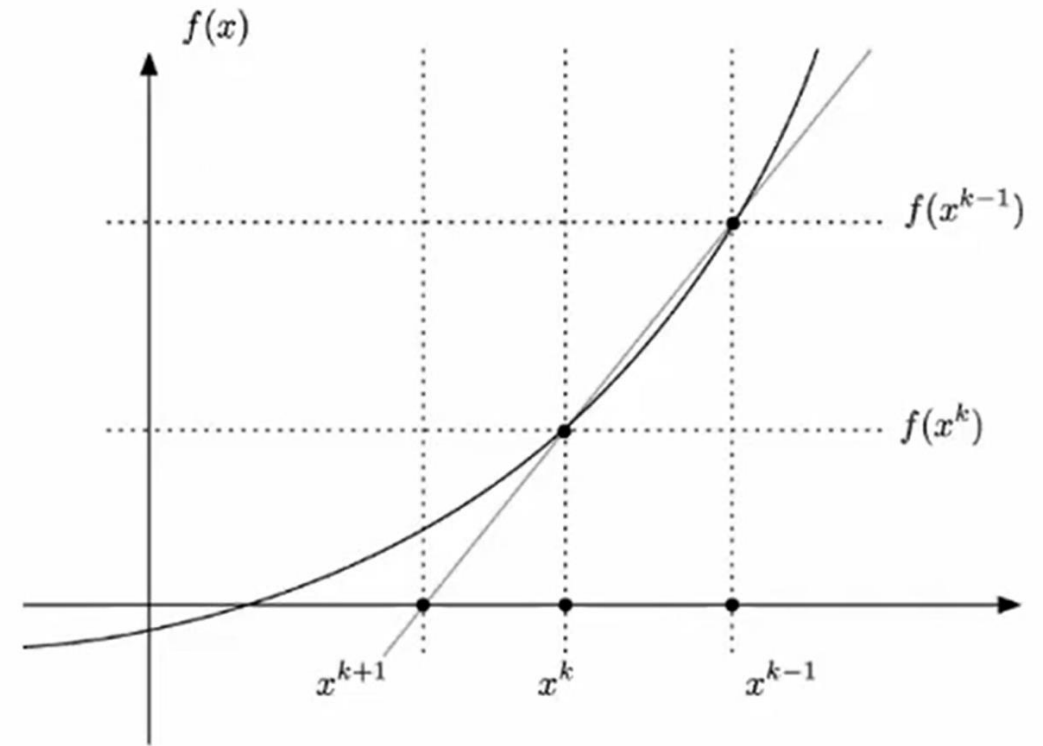


Global Optimization Test Problems. Retrieved June 2013, from http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm.

QUASI-NEWTON METHOD : SECANT CONDITION



Newton: $B^k = DF(x^k)$



Direct: $B^k(x^k - x^{k-1}) = F(x^k) - F(x^{k-1})$

Dual: $x^k - x^{k-1} = H^k(F(x^k) - F(x^{k-1}))$

QUASI-NEWTON METHOD

- $E(\mathbf{w}_k + \mathbf{d}) \approx m_k(\mathbf{d}) \approx E(\mathbf{w}_k) + \nabla E(\mathbf{w}_k)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \nabla^2 E(\mathbf{w}_k) \mathbf{d}. \quad \dots (Eq. 14)$

- The minimizer \mathbf{d}_k is given as

$$\mathbf{d}_k = -\nabla^2 E(\mathbf{w}_k)^{-1} \nabla E(\mathbf{w}_k) = -\mathbf{B}_k^{-1} \nabla E(\mathbf{w}_k). \quad \dots (Eq. 15)$$

- The new iterate \mathbf{w}_{k+1} is given as,

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha_k \mathbf{B}_k^{-1} \nabla E(\mathbf{w}_k), \quad \dots (Eq. 16)$$

and the quadratic model at the new iterate is given as

$$E(\mathbf{w}_{k+1} + \mathbf{d}) \approx m_{k+1}(\mathbf{d}) \approx E(\mathbf{w}_{k+1}) + \nabla E(\mathbf{w}_{k+1})^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \mathbf{B}_{k+1} \mathbf{d} \dots (Eq. 17)$$

QUASI-NEWTON METHOD + NESTEROV'S ACCELERATION

- The Nesterov's acceleration approximates the quadratic model at $\mathbf{w}_k + \mu \mathbf{v}_k$ instead of the iterate at \mathbf{w}_k

The new iterate \mathbf{w}_{k+1} is given as,

$$\begin{aligned}\mathbf{w}_{k+1} &= \mathbf{w}_k + \mu_k \mathbf{v}_k - \alpha_k \mathbf{B}_k^{-1} \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \\ &= \mathbf{w}_k + \mu_k \mathbf{v}_k + \alpha_k \mathbf{d}_k.\end{aligned}$$

QUASI-NEWTON METHOD + NESTEROV'S ACCELERATION

We have

$$E(\mathbf{w}_k + d) \approx m_k(d)$$
$$E(\mathbf{w}_{k+1} + d) \approx m_{k+1}(d)$$

Require:

m_{k+1} matches the gradient at the previous **two** iterations, i.e.,

1. $\nabla m_{k+1}|_{\mathbf{d}=0} = \nabla E(\mathbf{w}_{k+1} + \mathbf{d})|_{\mathbf{d}=0} = \nabla E(\mathbf{w}_{k+1})$.
2. $\nabla m_{k+1}|_{\mathbf{d}=-\alpha_k \mathbf{d}_k} = \nabla E(\mathbf{w}_{k+1} + \mathbf{d})|_{\mathbf{d}=-\alpha_k \mathbf{d}_k} = \nabla E(\mathbf{w}_{k+1} - \alpha_k \mathbf{d}_k) = \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$

QUASI-NEWTON METHOD + NESTEROV'S ACCELERATION

Proof: $E(\mathbf{w}_{k+1} + \mathbf{d}) \approx m_{k+1}(\mathbf{d}) = E(\mathbf{w}_{k+1}) + \nabla E(\mathbf{w}_{k+1})^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \nabla^2 E(\mathbf{w}_{k+1}) \mathbf{d}$

Condition 1: $\nabla m_{k+1}|_{\mathbf{d}=0} = \nabla E(\mathbf{w}_{k+1} + \mathbf{d})|_{\mathbf{d}=0} = \nabla E(\mathbf{w}_{k+1})$.

$$\nabla m_{k+1}(\mathbf{d}) = \nabla E(\mathbf{w}_{k+1}) + \nabla^2 E(\mathbf{w}_{k+1}) \mathbf{d}$$

$$\nabla m_{k+1}(0) = \nabla E(\mathbf{w}_{k+1}) + \nabla^2 E(\mathbf{w}_{k+1}) \mathbf{d} \big|_{\mathbf{d}=0} \Rightarrow \textit{satisfied}$$

Condition 2: $\nabla m_{k+1}|_{\mathbf{d}=-\alpha_k \mathbf{d}_k} = \nabla E(\mathbf{w}_{k+1} + \mathbf{d})|_{\mathbf{d}=-\alpha_k \mathbf{d}_k} = \nabla E(\mathbf{w}_{k+1} - \alpha_k \mathbf{d}_k) = \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k)$

$$\nabla m_{k+1}(-\alpha \mathbf{d}_k) = \nabla E(\mathbf{w}_{k+1}) - \alpha \nabla^2 E(\mathbf{w}_{k+1}) \mathbf{d}_k$$

$$\nabla m_{k+1}(-\alpha \mathbf{d}_k) = \nabla E(\mathbf{w}_{k+1}) - \alpha \nabla^2 E(\mathbf{w}_{k+1}) \mathbf{d}_k = \nabla E(\mathbf{w}_{k+1} - \alpha \mathbf{d}_k) = \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$$

$$\nabla E(\mathbf{w}_{k+1}) - \alpha \nabla^2 E(\mathbf{w}_{k+1}) \mathbf{d}_k = \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)$$

$$\nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k) = \mathbf{B}_{k+1} (\mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k))$$

$$\mathbf{q}_k = \mathbf{B}_{k+1} \mathbf{p}_k \Rightarrow \textit{Secant Condition}$$

$$(\mathbf{p}_k, \mathbf{q}_k) \Rightarrow \textit{Curvature Information Pair}$$

S. Indrapriyadarsini, Shahrzad Mahboubi, Hiroshi Ninomiya, Takeshi Kamio, Hideki Asai, "Accelerating Symmetric Rank 1 Quasi-Newton Method with Nesterov's Gradient", Algorithms 2022, 15(1), 6;

MODIFIED NESTEROV'S ACCELERATED BFGS QUASI-NEWTON – mNAQ

1) Incorporating an additional $\hat{\xi}_k \mathbf{p}_k$ term for better convergence

$$\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k)$$

$$\mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k) + \hat{\xi}_k \mathbf{p}_k = \boldsymbol{\varepsilon}_k + \hat{\xi}_k \mathbf{p}_k \Rightarrow \textit{Modified Secant Condition} \quad \dots (\text{Eq. 18})$$

$$\hat{\mathbf{H}}_{k+1} = (\mathbf{I} - \rho_k \mathbf{p}_k \mathbf{q}_k^T) \hat{\mathbf{H}}_k (\mathbf{I} - \rho_k \mathbf{q}_k \mathbf{p}_k^T) + \rho_k \mathbf{p}_k \mathbf{p}_k^T$$

convergence term

2) Eliminating linesearch

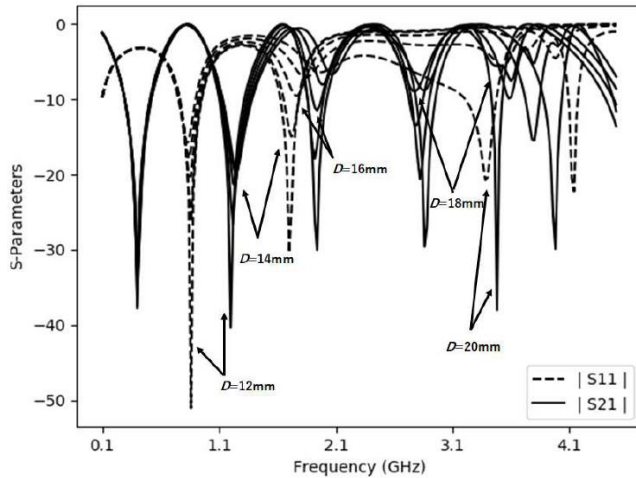
Determine step size α_k using the explicit formula

$$\alpha_k = - \frac{\delta \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)^T \hat{\mathbf{g}}_k}{\|\hat{\mathbf{g}}_k\|_{\mathbf{Q}_k}^2} \quad \dots (\text{Eq. 19})$$

Linesearch -> more number of function evaluations -> increased computation time

Indrapriyadarsini S., Shahrzad Mahboubi, Hiroshi Ninomiya, and Hideki Asai. "Implementation of a modified Nesterov's Accelerated quasi-Newton method on Tensorflow" In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE (2018) 1147–1154

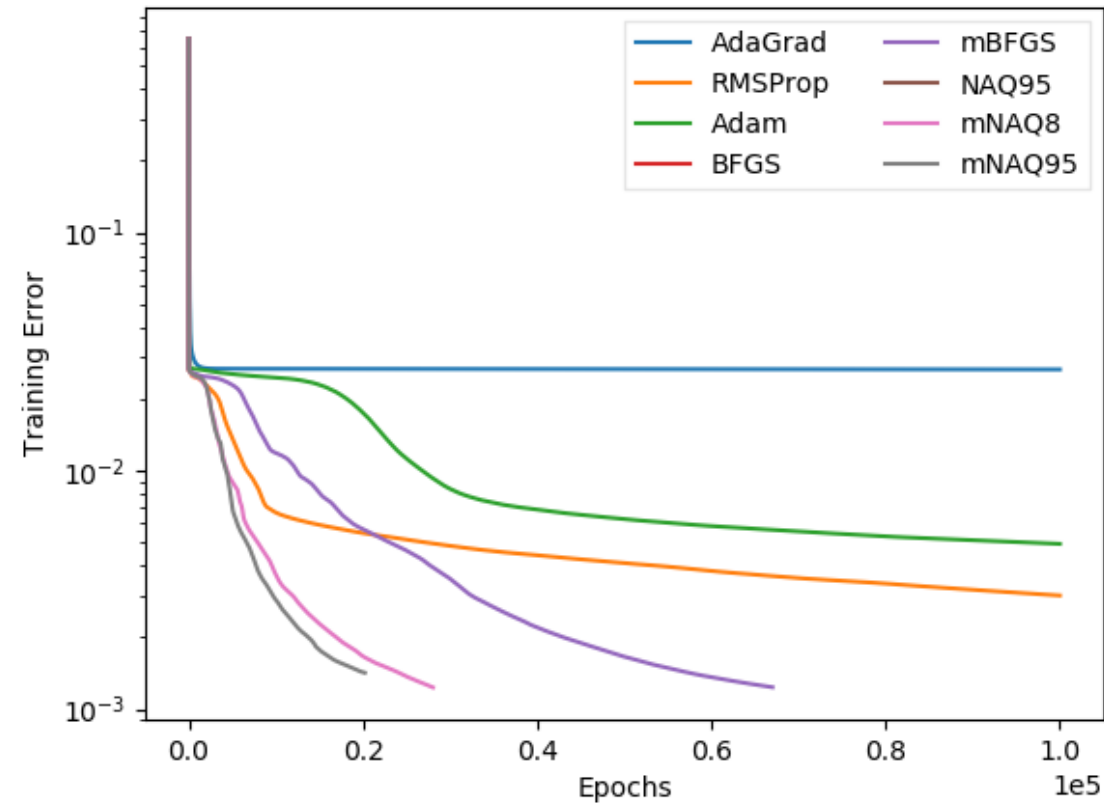
MICROSTRIP LOW PASS FILTER MODELING PROBLEM



Inputs : $D=12,14,16,18,20\text{mm}$
 Input frequency $f = 0.1 - 4.5\text{GHz}$
 Outputs: S parameters $|s_{11}|$ and $|s_{21}|$

- Input nodes = 2
- Hidden neurons = 45
- Output nodes = 2
- Parameters = 227
- Training data : 1105
- Test data: 884

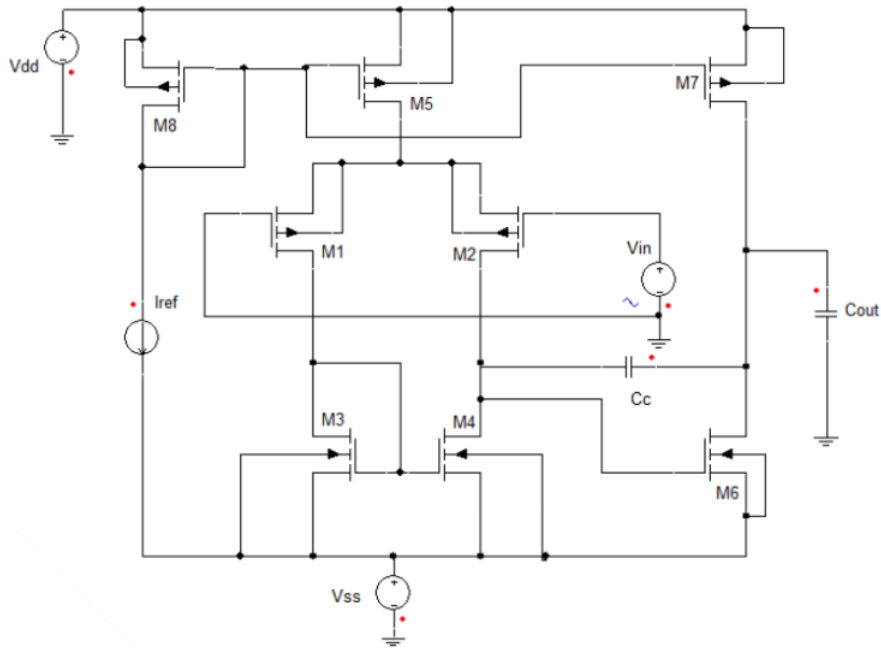
Average training error vs epoch over 15 trials



Algorithm	μ	$E(\mathbf{w})(\times 10^{-3})$ Ave/Best/Worst	Time (s)	Iteration count	$E_{test}(\mathbf{w})(\times 10^{-3})$ Ave/Best/Worst
AdaGrad	-	26.6 / 26.4 / 26.7	112	100,000	22.4 / 22.3 / 22.5
RMSprop	-	2.99 / 2.44 / 4.07	113	100,000	7.00 / 1.88 / 36.0
Adam	-	4.63 / 3.67 / 5.60	137	100,000	37.0 / 3.41 / 212.5
mBFGS	-	1.04 / 0.834 / 1.46	493	81,457	1.01 / 0.529 / 3.52
mNAQ	0.8	0.93 / 0.827 / 1.37	303	38,470	0.744 / 0.534 / 1.07
	0.85	1.02 / 0.756 / 1.62	314	39,678	7.32 / 5.75 / 87.8
	0.9	1.00 / 0.716 / 1.46	242	30,619	0.842 / 0.558 / 1.87
	0.95	1.24 / 0.834 / 1.85	209	26,547	2.08 / 0.600 / 13.7

Indrapriyadarsini S., Shahrzad Mahboubi, Hiroshi Ninomiya, and Hideki Asai. "Implementation of a modified Nesterov's Accelerated quasi-Newton method on Tensorflow" In: 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE (2018) 1147–1154

CIRCUIT DESIGN OPTIMIZATION

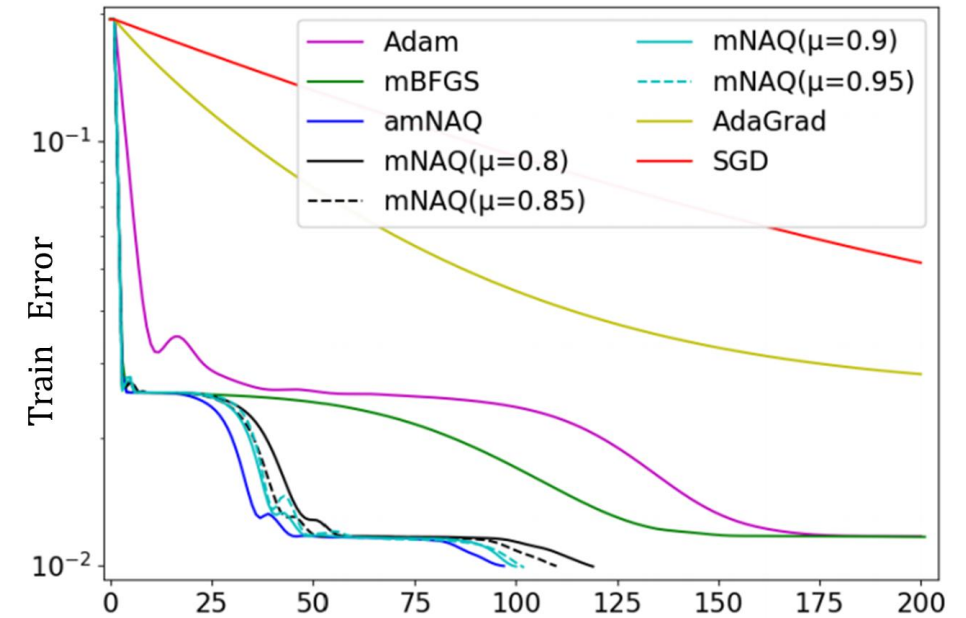


DESIGN SPECIFICATION	
Parameter	Value
Supply Voltage	$\pm 2.5V$
$\mu_n C_{ox}$	$160\mu A/V^2$
$\mu_p C_{ox}$	$40\mu A/V^2$
Unity GBW	> 1 MHz
Open Loop Gain A_o (dB)	> 50 dB
Phase Margin	> 60 deg

Adaptive Momentum

$$\mu_k = \theta_k(1 - \theta_k)/(\theta_k^2 + \theta_{k+1}),$$

$$\theta_{k+1}^2 = (1 - \theta_{k+1})\theta_k^2 + \eta\theta_{k+1}.$$

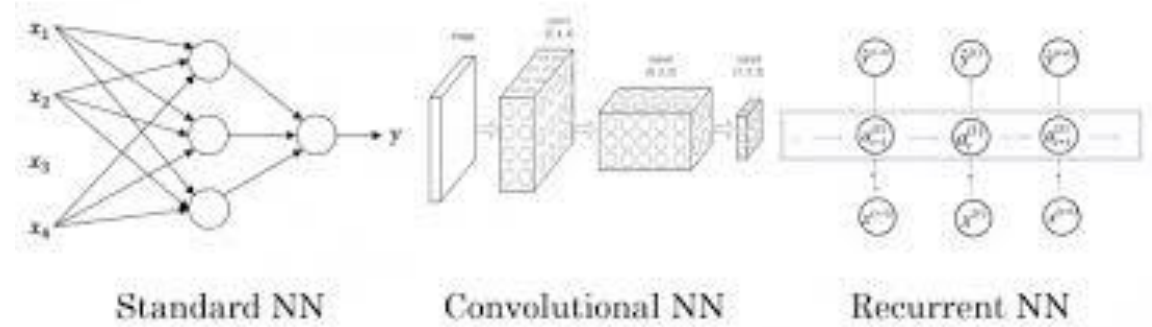


SUMMARY OF THE RESULTS OVER 30 TRIALS

Algorithm	μ_k	$E_{train}(\mathbf{w})(\times 10^{-3})$	CR (%)	Average epochs	$E_{test}(\mathbf{w})(\times 10^{-3})$
		Ave/Best/Worst			Ave/Best/Worst
SGD	-	66.402 / 43.153 / 113.334	-	200	68.428 / 45.852 / 118.620
AdaGrad	-	35.102 / 26.927 / 53.736	-	200	36.784 / 29.450 / 57.535
Adam	-	11.777 / 11.288 / 16.394	-	200	13.576 / 13.103 / 17.860
BFGS	-	11.354 / 11.287 / 11.464	-	200	13.193 / 13.142 / 13.261
mNAQ	0.8	10.010 / 9.892 / 11.194	90	161	11.862 / 11.610 / 13.008
	0.85	10.005 / 9.889 / 11.097	93.3	156	11.859 / 11.616 / 12.907
	0.9	9.966 / 9.880 / 10.478	93.3	156	11.813 / 11.603 / 12.416
	0.95	10.305 / 9.874 / 11.328	63.3	178	12.098 / 11.477 / 13.154
amNAQ	-	9.997 / 9.849 / 11.285	96.7	146	11.799 / 11.546 / 13.105

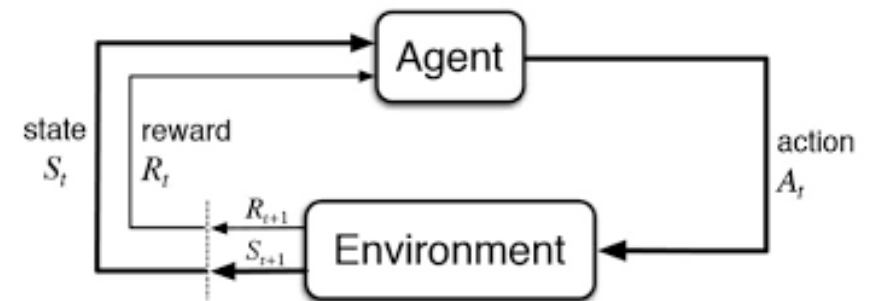
Indrapriyadarsini S., Shahrzad Mahboubi, Hiroshi Ninomiya, Takeshi Kamio and Hideki Asai. "A Neural Network Approach to Analog Circuit Design Optimization using Nesterov's Accelerated Quasi-Newton Method." 2020 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2020

OPTIMIZATION IN LARGE SCALE PROBLEMS

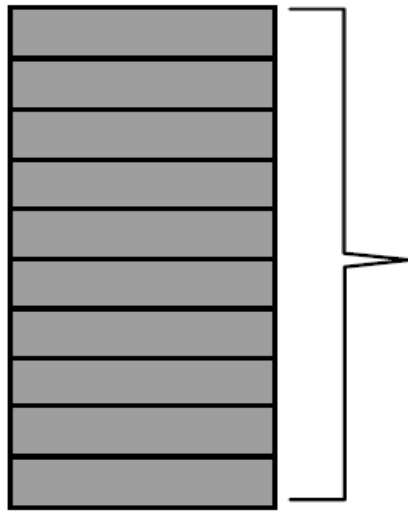


Require

- Fast training
- Good accuracy
- Reduce computation cost



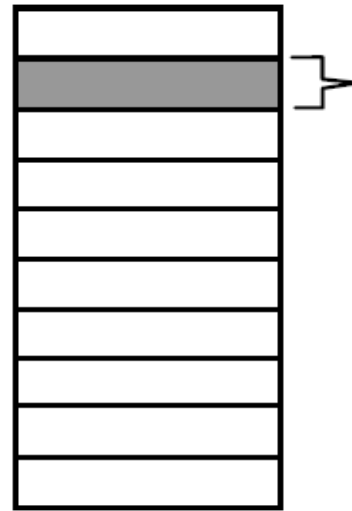
FULL BATCH VS STOCHASTIC/MINI-BATCH



Training Data

Full Batch

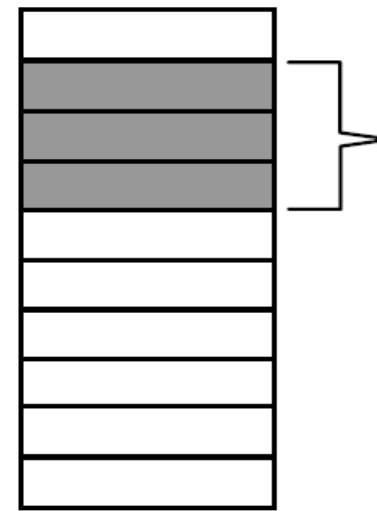
High stability but for large scale problems – high computation and time consuming



Training Data

Stochastic

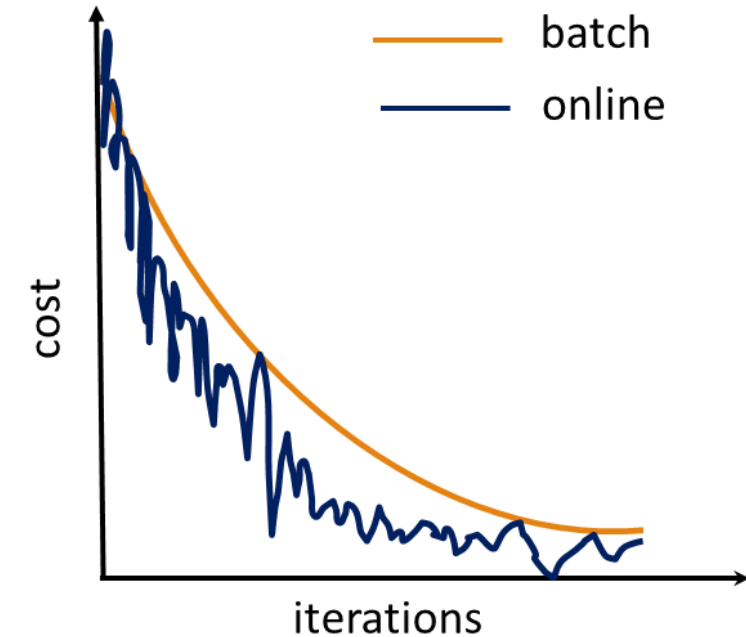
Noisy but fast and lesser computation cost – suitable for large scale optimization



Training Data

Mini-batch

Combines the advantages of full batch and stochastic methods



One epoch is when the entire dataset is processed by the neural network.
In full batch, 1 iteration = 1 epoch
In mini-batch, if $|Tr| = 100$, $b = 10$
10 iterations = 1 epoch

OUTLINE

Introduction

Background

Stochastic accelerated quasi-Newton Methods

- Proposed Methods, Results and Convergence Analysis
 - Accelerated Stochastic Quasi-Newton [oLNAQ / oLMoQ]
 - Adaptive Stochastic Quasi-Newton [aSNAQ]
 - Accelerated Symmetric Rank-1 Quasi-Newton [LSR1-N]

STOCHASTIC BFGS QUASI-NEWTON METHOD (oBFGS)

- The update vector of quasi-Newton (QN) method

$$\mathbf{v}_{k+1} = -\alpha_k \mathbf{H}_k \nabla E(\mathbf{w}_k, \mathbf{X}_k) \quad \dots (\text{Eq. 20})$$

- The matrix \mathbf{H}_k is iteratively approximated by **BFGS** formula

$$\mathbf{H}_{k+1} = (\mathbf{I} - \rho_k \mathbf{s}_k \mathbf{y}_k^T) \mathbf{H}_k (\mathbf{I} - \rho_k \mathbf{y}_k \mathbf{s}_k^T) + \rho_k \mathbf{s}_k \mathbf{s}_k^T \quad \dots (\text{Eq. 21})$$

$$\rho_k = \frac{1}{\mathbf{y}_k^T \mathbf{s}_k}, \quad \mathbf{s}_k = \mathbf{w}_{k+1} - \mathbf{w}_k \quad \text{and} \quad \mathbf{y}_k = \underbrace{\nabla E(\mathbf{w}_{k+1}, \mathbf{X}_k)}_{\text{Two Gradients per iteration}} - \underbrace{\nabla E(\mathbf{w}_k, \mathbf{X}_k)}_{\text{Two Gradients per iteration}} \quad \dots (\text{Eq. 22})$$

- Step size calculated as $\alpha_k = \tau / (\tau + k) \alpha_0$

Schraudolph, N.N., Yu, J., Günter, S.: A stochastic quasi-newton method for online convex optimization. In: Artificial Intelligence and Statistics. (2007) 436–443

STOCHASTIC NESTEROV'S ACCELERATED QUASI-NEWTON

- The update vector of stochastic quasi-Newton (QN) method

$$\mathbf{v}_{k+1} = -\alpha_k \mathbf{H}_k \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k, \mathbf{X}_k)$$

NAQ computes two gradients per iteration (on same mini-batch)

$$\mathbf{p}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k)$$

$$\mathbf{q}_k = \nabla E(\mathbf{w}_{k+1}, \mathbf{X}_k) - \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k, \mathbf{X}_k) + \lambda \mathbf{p}_k$$

$$\hat{\mathbf{H}}_{k+1} = (\mathbf{I} - \rho_k \mathbf{p}_k \mathbf{q}_k^T) \hat{\mathbf{H}}_k (\mathbf{I} - \rho_k \mathbf{q}_k \mathbf{p}_k^T) + \rho_k \mathbf{p}_k \mathbf{p}_k^T$$

Reduced sampling noise

Same computational cost as o(L)BFGS + faster convergence

Indrapriyadarsini S., Shahrzad Mahboubi, Hiroshi Ninomiya, and Hideki Asai. "A Stochastic Quasi-Newton Method with Nesterov's Accelerated Gradient", Joint European Conference on Machine Learning and Principles of Knowledge Discovery in Databases, ECML-PKDD, Springer, 2019

STOCHASTIC NESTEROV'S ACCELERATED QUASI-NEWTON

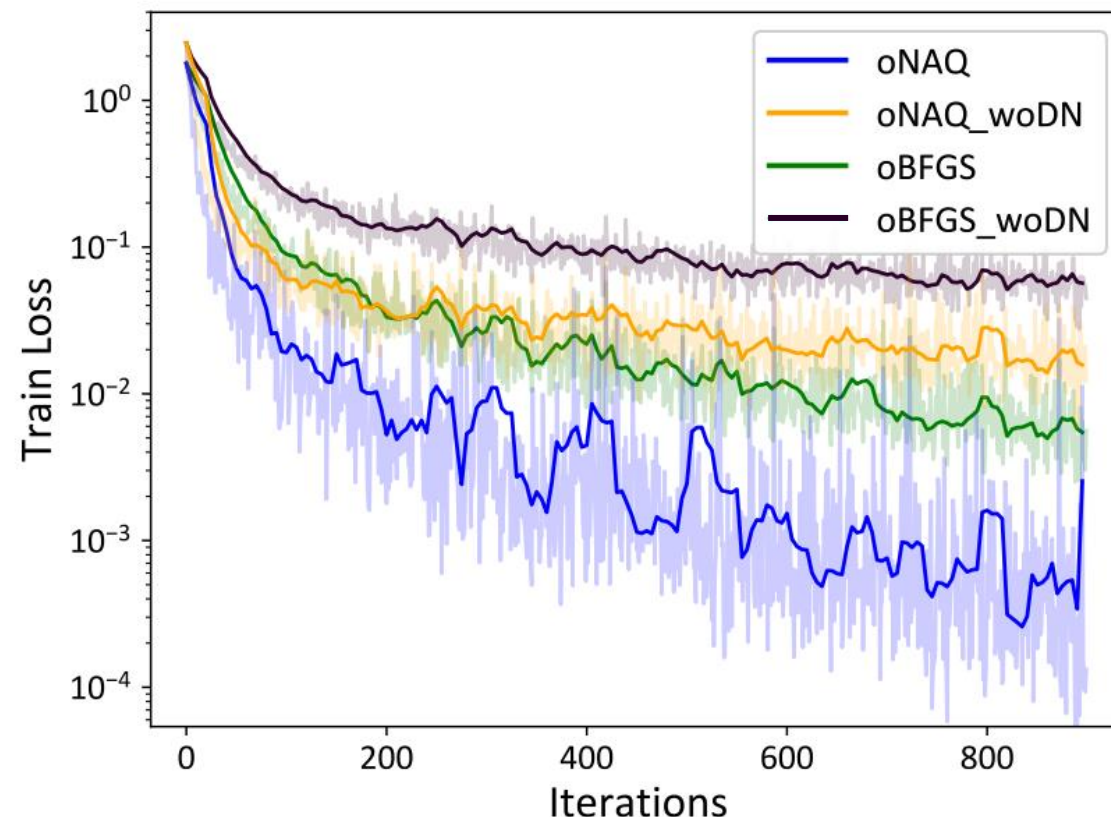
Direction Normalization

Further to improve the stability, direction normalization is introduced.

$$\hat{\mathbf{g}}_k \leftarrow -\hat{\mathbf{H}}_k \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k, \mathbf{X}_k)$$

$$\hat{\mathbf{g}}_k = \frac{\hat{\mathbf{g}}_k}{\|\hat{\mathbf{g}}_k\|_2}$$

Normalizing the search direction at each iteration ensure that the algorithm does not move too far away from the current objective



Effect of direction normalization

Indrapriyadarsini S., Shahrzad Mahboubi, Hiroshi Ninomiya, and Hideki Asai. "A Stochastic Quasi-Newton Method with Nesterov's Accelerated Gradient", Joint European Conference on Machine Learning and Principles of Knowledge Discovery in Databases, ECML-PKDD, Springer, 2019

COMPARISON OF oBFGS AND oNAQ ALGORITHM

oBFGS ALGORITHM

Require minibatch $X_k, 0 < \mu < 1, k_{max}$

Initialize $\mathbf{w}_k \in \mathbb{R}^d, \mathbf{H}_k = \epsilon \mathbf{I}$ and $k \leftarrow 1$

while $k < k_{max}$ do

1. $\nabla E_1 \leftarrow \nabla E(\mathbf{w}_k, X_k)$
2. $\mathbf{g}_k \leftarrow -\mathbf{H}_k \nabla E(\mathbf{w}_k, X_k)$
3. $\mathbf{g}_k = \mathbf{g}_k / \|\mathbf{g}_k\|_2 \leftarrow \text{Direction Normalization}$
4. $\alpha_k = \tau / (\tau + k) \alpha_0$
5. $\mathbf{v}_{k+1} = \alpha_k \mathbf{g}_k$
6. $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$
7. $\nabla E_2 \leftarrow \nabla E(\mathbf{w}_{k+1}, X_k)$
8. $\mathbf{s}_k \leftarrow \mathbf{w}_{k+1} - \mathbf{w}_k$
9. $\mathbf{y}_k \leftarrow \nabla E_2 - \nabla E_1 + \lambda \mathbf{s}_k$
10. Update \mathbf{H}_k
11. $k \leftarrow k + 1$

end while

oNAQ ALGORITHM

Require minibatch $X_k, 0 < \mu < 1, k_{max}$

Initialize $\mathbf{w}_k \in \mathbb{R}^d, \hat{\mathbf{H}}_k = \epsilon \mathbf{I}, \mathbf{v}_k = 0$ and $k \leftarrow 1$

while $k < k_{max}$ do

1. $\nabla E_1 \leftarrow \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k, X_k)$
2. $\hat{\mathbf{g}}_k \leftarrow -\hat{\mathbf{H}}_k \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k, X_k)$
3. $\hat{\mathbf{g}}_k = \hat{\mathbf{g}}_k / \|\hat{\mathbf{g}}_k\|_2 \leftarrow \text{Direction Normalization}$
4. $\alpha_k = \alpha_0 / \sqrt{k}$
5. $\mathbf{v}_{k+1} = \mu \mathbf{v}_k + \alpha_k \hat{\mathbf{g}}_k$
6. $\mathbf{w}_{k+1} = \mathbf{w}_k + \mathbf{v}_{k+1}$
7. $\nabla E_2 \leftarrow \nabla E(\mathbf{w}_{k+1}, X_k)$
8. $\mathbf{p}_k \leftarrow \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k)$
9. $\mathbf{q}_k \leftarrow \nabla E_2 - \nabla E_1 + \lambda \mathbf{p}_k$
10. Update $\hat{\mathbf{H}}_k$
11. $k \leftarrow k + 1$

end while

LIMITED MEMORY

$$\mathbf{H} = \nabla^2 E(\mathbf{w}) = \begin{bmatrix} \frac{\partial^2 E}{\partial w_1^2} & \frac{\partial^2 E}{\partial w_1 w_2} & \cdots & \frac{\partial^2 E}{\partial w_1 w_D} \\ \frac{\partial^2 E}{\partial w_2 w_1} & \frac{\partial^2 E}{\partial w_2^2} & \cdots & \frac{\partial^2 E}{\partial w_2 w_D} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 E}{\partial w_d w_1} & \frac{\partial^2 E}{\partial w_d w_2} & \cdots & \frac{\partial^2 E}{\partial w_d^2} \end{bmatrix}$$

- Require

$$\mathbf{g}_k \leftarrow -\mathbf{H}_k \nabla E(\mathbf{w}_k, \mathbf{X}_k)$$

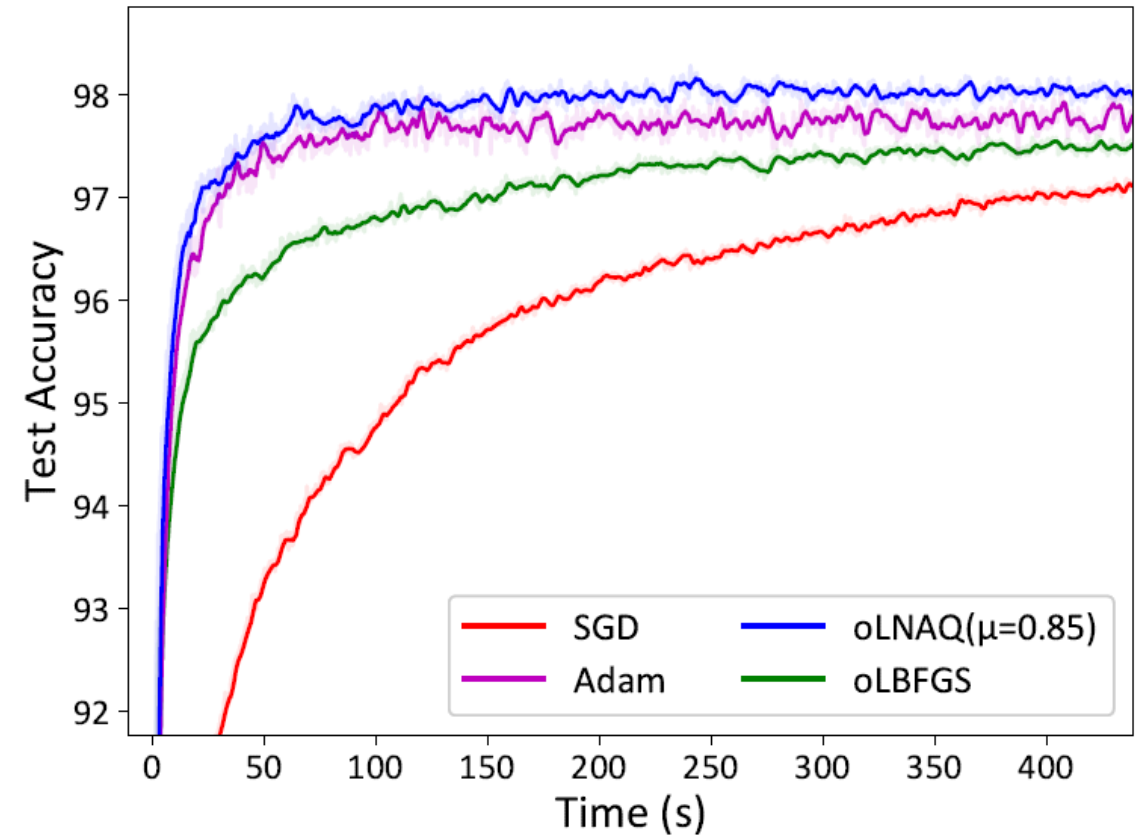
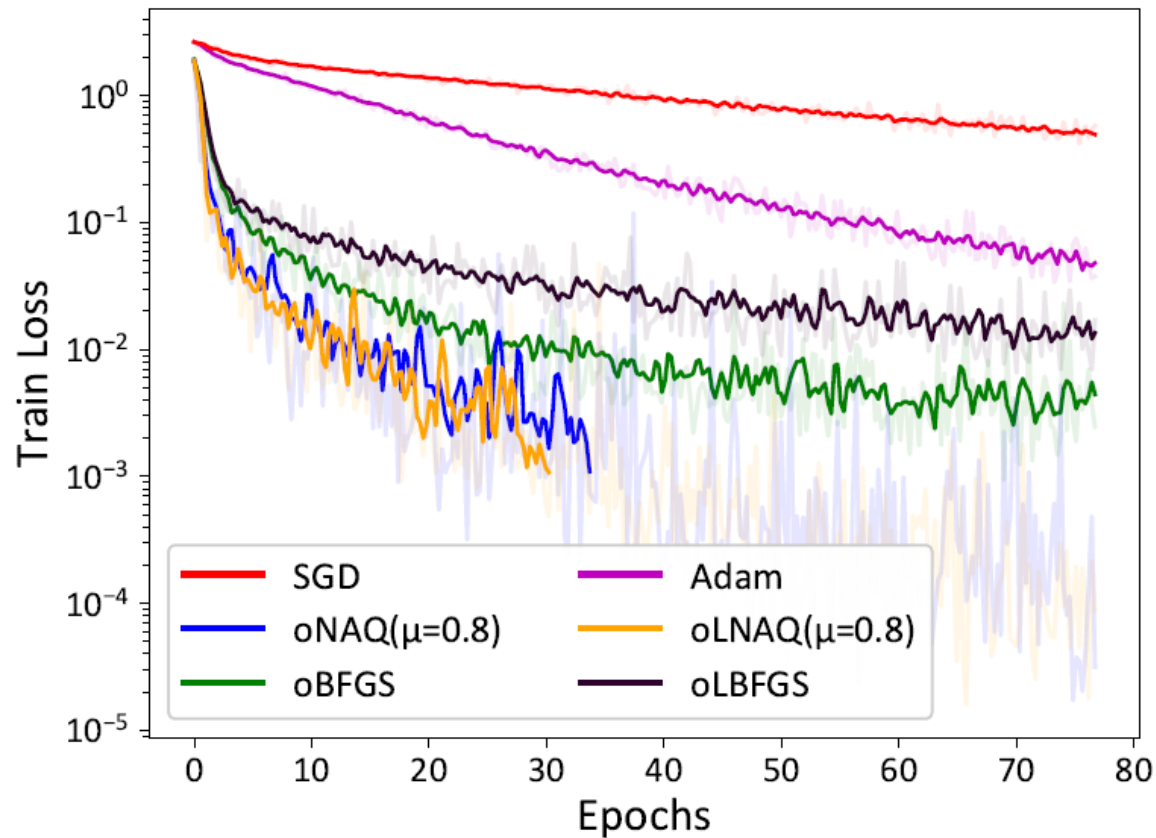
- Store recent m curvature information pairs

Two-loop Recursion

Require The current gradient $\nabla E(\boldsymbol{\theta}_k, \mathbf{X}_k)$, memory m , curvature pair information

1. $\boldsymbol{\eta}_k \leftarrow -\nabla E(\boldsymbol{\theta}_k, \mathbf{X}_k)$
2. **for** $i = 1, 2, \dots, \min(m, k - 1)$ **do**
3. $\beta_i = (\boldsymbol{\sigma}_{k-i}^T \boldsymbol{\eta}_k) / (\boldsymbol{\sigma}_{k-i}^T \boldsymbol{\gamma}_{k-i})$
4. $\boldsymbol{\eta}_{k \cdot} = \boldsymbol{\eta}_k - \beta_i \boldsymbol{\gamma}_{k-i}$
5. **end for**
6. **if** $k > 1$
7. $\boldsymbol{\eta}_{k \cdot} = \boldsymbol{\eta}_k (\boldsymbol{\sigma}_k^T \boldsymbol{\gamma}_k) / (\boldsymbol{\gamma}_k^T \boldsymbol{\gamma}_k)$
8. **end if**
9. **for** $i = \min(m, k - 1), \dots, 2, 1$ **do**
10. $\tau = (\boldsymbol{\gamma}_i^T \boldsymbol{\eta}_k) / (\boldsymbol{\gamma}_i^T \boldsymbol{\sigma}_i)$
11. $\boldsymbol{\eta}_{k \cdot} = \boldsymbol{\eta}_k - (\beta_i - \tau) \boldsymbol{\sigma}_i$
12. **end for**
13. **return** $\boldsymbol{\eta}_k$

STOCHASTIC NESTEROV'S ACCELERATED QUASI-NEWTON – oNAQ

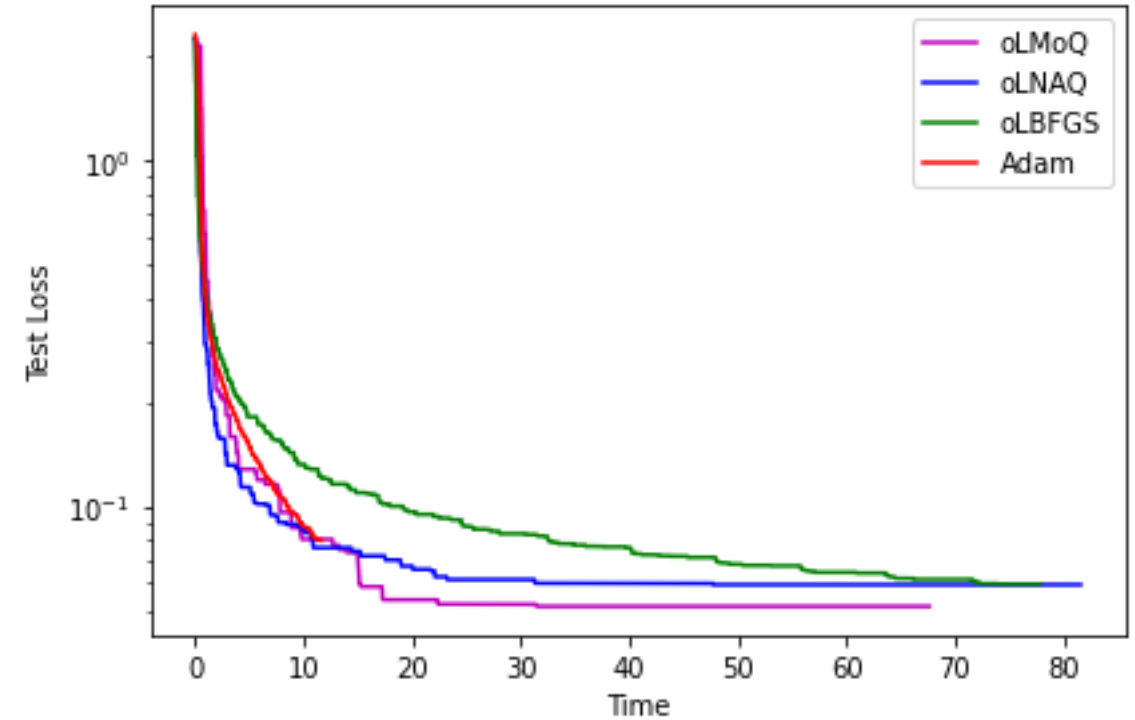
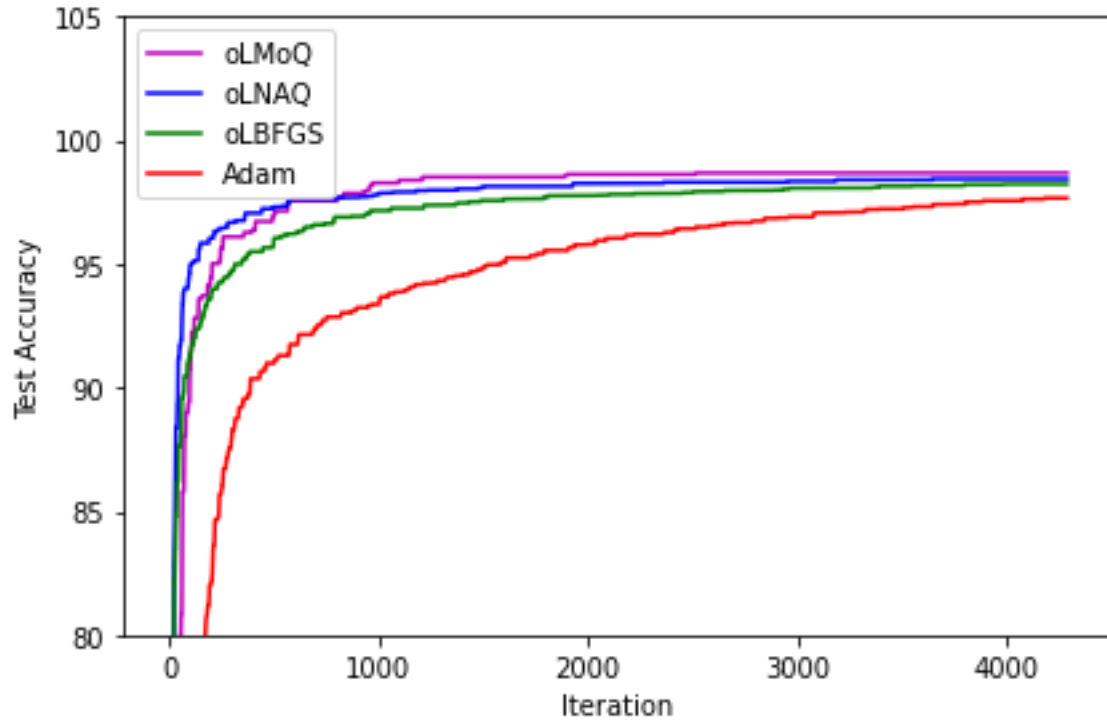


Results on MNIST Classification

Indrapriyadarsini S., Shahrzad Mahboubi, Hiroshi Ninomiya, and Hideki Asai. "A Stochastic Quasi-Newton Method with Nesterov's Accelerated Gradient", Joint European Conference on Machine Learning and Principles of Knowledge Discovery in Databases, ECML-PKDD, Springer, 2019

STOCHASTIC MOMENTUM ACCELERATED QUASI-NEWTON – oMoQ

$$\nabla E(\mathbf{w}_k + \mu \mathbf{v}_k) \approx (1 + \mu_k) \nabla E(\mathbf{w}_k) - \mu_k \nabla E(\mathbf{w}_{k-1})$$



Results on MNIST Classification on LeNet-5

S. Indrapriyadarsini, Shahrzad Mahboubi, Hiroshi Ninomiya, Takeshi Kamio, Hideki Asai, "A Stochastic Momentum Accelerated Quasi-Newton Method for Neural Networks (Student Abstract)", Proceedings of the 36th AAAI Conference on Artificial Intelligence, Feb 2022

STOCHASTIC MOMENTUM / NESTEROV'S ACCELERATED QUASI-NEWTON

Summary of Computational Cost and Storage

	Algorithm	Computational Cost	Storage
full batch	BFGS	$nd + d^2 + \zeta nd$	d^2
	NAQ	$2nd + d^2 + \zeta nd$	d^2
	MoQ	$nd + d^2 + \zeta nd$	$d^2 + d$
	LBFGS	$nd + 4md + 2d + \zeta nd$	$2md$
	LNAQ	$2nd + 4md + 2d + \zeta nd$	$2md$
	LMoQ	$nd + 4md + 2d + \zeta nd$	$(2m + 1)d$
online	oBFGS	$2bd + d^2$	d^2
	oNAQ	$2bd + d^2$	d^2
	oMoQ	$bd + d^2$	$d^2 + d$
	oLBFGS	$2bd + 6md$	$2md$
	oLNAQ	$2bd + 6md$	$2md$
	oLMoQ	$bd + 6md$	$(2m + 1)d$

CONVERGENCE ANALYSIS : AN ALTERNATE EXPRESSION

We have,

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \mu_k \mathbf{v}_k - \alpha_k \mathbf{H}_k \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k), \quad \mathbf{v}_{k+1} = \mu_k \mathbf{v}_k - \alpha_k \mathbf{H}_k \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k).$$

Let $\mathbf{a}_k = \frac{\mathbf{v}_k}{\alpha_k}$

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \alpha_k \mathbf{a}_{k+1}, \quad \mathbf{a}_{k+1} = \mu_k \mathbf{a}_k - \mathbf{H}_k \nabla E(\mathbf{w}_k + \mu_k \alpha_k \mathbf{a}_k).$$

Let $\hat{\mathbf{w}}_k = \mathbf{w}_k + \mu_k \alpha_k \mathbf{a}_k$.

$$\hat{\mathbf{w}}_{k+1} = \hat{\mathbf{w}}_k - \alpha_k \sum_{i=0}^k (\mu_k^{k-i} \mathbf{H}_i \nabla E(\hat{\mathbf{w}}_i)) + \mu_k \alpha_k \left[(1 - \mu_k) \sum_{i=0}^{k-1} (\mu_k^{k-i} \mathbf{H}_i \nabla E(\hat{\mathbf{w}}_i)) - \mathbf{H}_k \nabla E(\hat{\mathbf{w}}_k) \right]$$

$$\hat{\mathbf{w}}_{k+1} \approx \hat{\mathbf{w}}_k - \alpha_k \mathbf{H}_k \nabla E(\hat{\mathbf{w}}_k) + \mu_k \alpha_k \left[(1 - \mu_k) \mathbf{H}_{k-1} \nabla E(\hat{\mathbf{w}}_{k-1}) - \mathbf{H}_k \nabla E(\hat{\mathbf{w}}_k) \right]$$

$$\hat{\mathbf{w}}_{k+1} \approx \hat{\mathbf{w}}_k - \alpha_k \mathbf{H}_k \nabla E(\hat{\mathbf{w}}_k) - \mu_k \alpha_k \mathbf{H}_k \nabla E(\hat{\mathbf{w}}_k)$$

$$\hat{\mathbf{w}}_{k+1} \approx \hat{\mathbf{w}}_k - (1 + \mu_k) \alpha_k \mathbf{H}_k \nabla E(\hat{\mathbf{w}}_k)$$

CONVERGENCE ANALYSIS

Assumption 1: The sequence of iterates \mathbf{w}_k and $\hat{\mathbf{w}}_k$ remains in the closed and bounded set Ω on which the objective function is twice continuously differentiable and has Lipschitz continuous gradient, i.e. there exists a constant $L > 0$ such that

$$\|\nabla E_b(\hat{\mathbf{w}}_{k+1}) - \nabla E_b(\hat{\mathbf{w}}_k)\| \leq L \|\hat{\mathbf{w}}_{k+1} - \hat{\mathbf{w}}_k\| \quad \forall \hat{\mathbf{w}}_k \in \mathbb{R}^d,$$

If *Assumption 1* holds true, then it implies that the objective function satisfies,

$$E_b(\mathbf{w}_{k+1}) \leq E_b(\mathbf{w}_k + \mu \mathbf{v}_k) + \nabla E_b(\mathbf{w}_k + \mu \mathbf{v}_k)^T \mathbf{d} + \frac{L}{2} \|\mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k)\|_2^2$$

And, the stochastic (minibatch) gradient is an unbiased estimator of the full gradient i.e.,

$$\mathbb{E}[\nabla E_b(\mathbf{w}_k + \mu \mathbf{v}_k, X_k)] \approx \nabla E_b(\mathbf{w}_k + \mu \mathbf{v}_k)$$

CONVERGENCE ANALYSIS

Assumption 2: The Hessian matrix $\mathbf{B}_k = \nabla^2 \mathbf{E}_b(\hat{\mathbf{w}}_k)$ constructed with mini-batch samples X_k is bounded and well-defined, .i.e, there exists constants ρ and L , such that

$$\rho \preceq \|\mathbf{B}_k\| \preceq L \quad \forall k = 1, 2, \dots, k_{max} \in \mathbb{N}$$

for all mini-batch samples $X_k \subset T_r$ of batch size b .

Assumption 2 ensures the subsampled Hessian matrix is symmetric positive semidefinite and bounded

Assumption 3: There exists a constant γ^2 such that $\forall \hat{\mathbf{w}}_k \in \mathbb{R}^d$ and batches $X_k \subset T_r$ of size b ,

$$\mathbb{E}_{X_k} [\|\nabla E(\hat{\mathbf{w}}_k, X_k)\|^2] \leq \gamma^2$$

Assumption 3 sets a bound on the stochastic variance noise of the subsampled gradients

Assumption 4: The sequence of step size α_k selected is nonsummable but square summable i.e.,

$$\sum_{k=0}^{\infty} \alpha_k = \infty \quad \text{and} \quad \sum_{k=0}^{\infty} \alpha_k^2 \leq \infty$$

CONVERGENCE ANALYSIS

Lemma : If Assumptions 1, 2, and 3 holds true, then the iterates $\{\hat{\mathbf{w}}_k\}$ and the objective function $E(\hat{\mathbf{w}}_k)$ satisfy,

$$\mathbb{E}_{X_k} [E(\hat{\mathbf{w}}_{k+1})] - E(\mathbf{w}^*) \leq E(\hat{\mathbf{w}}_k) - E(\mathbf{w}^*) - (1 + \mu_k^2)\alpha_k\delta_1\|\nabla E(\hat{\mathbf{w}}_k)\|^2 + \frac{L}{2}(1 + \mu_k^2)^2\alpha_k^2\delta_2^2\gamma^2$$

Theorem : Let $\{\hat{\mathbf{w}}_k\}$ be the sequence of iterates generated by the algorithm. If Assumptions 1 to 4 holds true, and

$$\delta_1\mathbf{I} \leq \|\mathbf{H}_k\| \leq \delta_2\mathbf{I} \quad \forall k = 1, 2, \dots, k_{max} \in \mathbb{N} \text{ and } 0 < \delta_1 \leq \delta_2$$

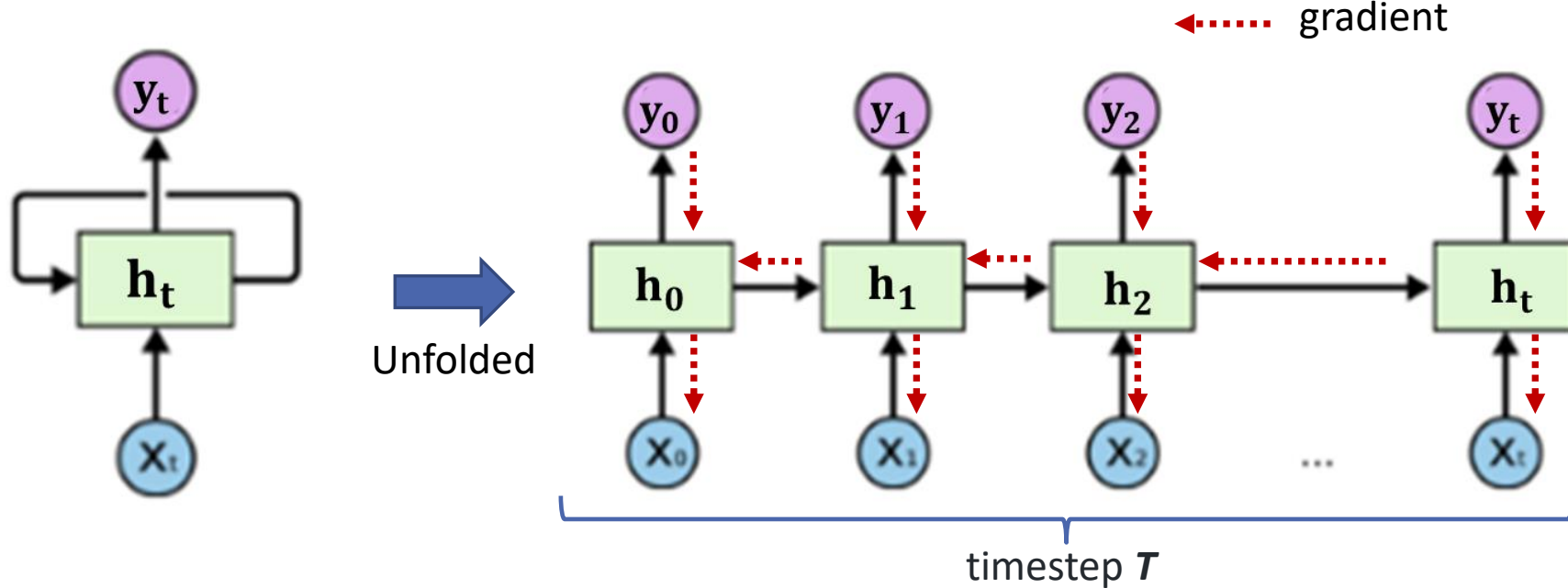
then

$$\lim_{k \rightarrow \infty} \|\nabla E(\hat{\mathbf{w}}_k)\| = 0.$$

And the convergence rate is given as

$$\frac{E(\hat{\mathbf{w}}_{k+1}) - E(\mathbf{w}^*)}{E(\hat{\mathbf{w}}_0) - E(\mathbf{w}^*)} \leq \left(1 - L(1 + \mu)\alpha_k \left[2\delta_1 + L(1 + \mu)\alpha_k\delta_2^2\right]\right)^k$$

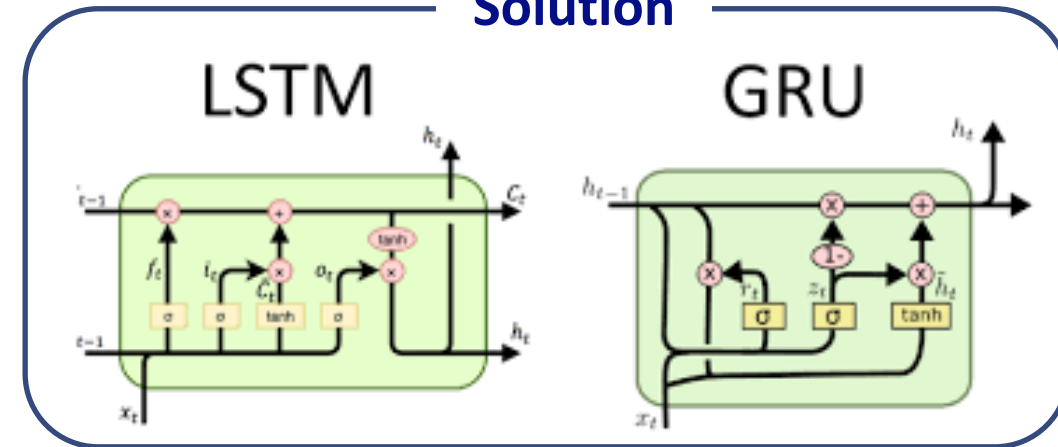
RECURRENT NEURAL NETWORKS



Recurrent Neural Networks

- Popular for sequence modeling problems
- Backpropagation through time
- Difficult training long sequences
- Vanishing/exploding gradient

Solution



Adaptive Stochastic Nesterov's Accelerated quasi-Newton – aSNAQ

- Builds on the algorithmic framework of SQN and **adaQN**



Indrapriyadarsini S., Shahrzad Mahboubi, Hiroshi Ninomiya, and Hideki Asai. "An Adaptive Stochastic Nesterov's Accelerated Quasi-Newton Method for Training RNNs", *Nonlinear Theory and its Applications, NOLTA, IEICE, 2019 (Best Student Paper Award)*

Adaptive Stochastic Nesterov's Accelerated quasi-Newton – aSNAQ

- Initial Scaling of the LBFGS matrix i.e. $\mathbf{H}_k^{(0)}$

$$\mathbf{H}_k^{(0)} \text{ is usually initialized as } \mathbf{H}_k^{(0)} = \frac{\mathbf{y}_k^T \mathbf{s}_k}{\mathbf{y}_k^T \mathbf{y}_k} \mathbf{I}$$

Scalar initialization of the L-BFGS matrix does not address the vanishing/exploding gradient issue.

\mathbf{s} and \mathbf{y} are noisy estimates of the differences of the weights and gradients, and thus could further deteriorate the performance.

Therefore $\mathbf{H}_k^{(0)}$ is initialized in the two-loop recursion algorithm based on the accumulated gradient information

$$[\mathbf{H}_k^{(0)}]_{ii} = \frac{1}{\sqrt{\sum_{j=0}^k \nabla E(\mathbf{w}_j)_i^2 + \varepsilon}}$$

Adaptive Stochastic Nesterov's Accelerated quasi-Newton – aSNAQ

➤ Direction Normalization

Common issue in training RNNs

- Exploding gradients – norm of the gradient increases exponentially
- Vanishing gradients – norm of the gradient tends to zero exponentially

Thus making it difficult for the model to learn the correlation.

Direction normalization scales the search direction in each iteration by its l_2 norm

$$\mathbf{g}_k \leftarrow \mathbf{H}_k \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k, \mathbf{X}_k)$$

$$\mathbf{g}_k = \frac{\mathbf{g}_k}{\|\mathbf{g}_k\|_2}$$

Nesterov's Accelerated Gradient(NAG)

Adaptive Stochastic Nesterov's Accelerated quasi-Newton – aSNAQ

➤ Curvature information matrix

QN methods generate high-quality steps even with crude curvature information.

Fisher Information matrix (FIM) yields a better estimate of the curvature.

A FIFO memory buffer \mathbf{F} of size $m_{\mathbf{F}}$ accumulates at each iteration the FIM as

$$F_i = \nabla E(\mathbf{w}_k) \nabla E(\mathbf{w}_k)^T$$

Normal Gradients

This accumulated FIM is used in the computation of the \mathbf{y} vector

$$\mathbf{y} \leftarrow \frac{1}{|\mathbf{F}|} \left(\sum_{i=1}^{|\mathbf{F}|} F_i \cdot \mathbf{s} \right) \quad \text{where } \mathbf{s} \leftarrow \mathbf{w}_n - \mathbf{w}_o$$

Adaptive Stochastic Nesterov's Accelerated quasi-Newton – aSNAQ

➤ Step acceptance and control

The weights are aggregated every iteration

$$\begin{aligned} \mathbf{v}_{k+1} &\leftarrow \mu \mathbf{v}_k + \alpha_k \mathbf{g}_k & \mathbf{w}_s &\leftarrow \mathbf{w}_s + \mathbf{w}_k \\ \mathbf{w}_{k+1} &\leftarrow \mathbf{w}_k + \mathbf{v}_{k+1} & \mathbf{v}_s &\leftarrow \mathbf{v}_s + \mathbf{v}_k \end{aligned}$$

Weights and Hessian matrix updated once every L iterations

$$\mathbf{w}_n \leftarrow \mathbf{w}_s / L \qquad \mathbf{v}_n \leftarrow \mathbf{v}_s / L$$

Store curvature pairs s and y only if they are sufficiently large enough.

if $s^T y > \epsilon \cdot y^T y$ **then**
 Store curvature pairs (s, y) in (S, Y)
else
 Skip storage

Step Control

if $E(\mathbf{w}_n) > \gamma E(\mathbf{w}_0)$ **then**

 Clear (S, Y) and F buffer

 Reset $\mathbf{w}_k = \mathbf{w}_0$ and $\mathbf{v}_k = \mathbf{v}_0$

 Update $\mu = \max(\mu / \phi, \mu_{min})$

else

$s \leftarrow \mathbf{w}_n - \mathbf{w}_0$

Adaptive momentum

$y \leftarrow \frac{1}{|F|} (\sum_{i=1}^{|F|} F_i \cdot s)$

 Update $\mu = \max(\mu \cdot \phi, \mu_{min})$

Adaptive Stochastic Nesterov's Accelerated quasi-Newton – aSNAQ

Summary of Computational and Storage Cost.

Algorithm	Computational Cost	Storage
BFGS	$nd + d^2 + \zeta nd$	d^2
NAQ	$2nd + d^2 + \zeta nd$	d^2
adaQN	$bd + (4m_L + m_F + 2)d + (b + 4)d/L$	$(2m_L + m_F)d$
aSNAQ	$2bd + (4m_L + m_F + 3)d + (b + 4)d/L$	$(2m_L + m_F)d$

CONVERGENCE ANALYSIS : aSNAQ

RECALL

Assumption 1 : Lipschitz continuity

Assumption 2 : Subsampled Hessian is symmetric positive semidefinite and bounded

Assumption 3 : Stochastic variance noise $< \gamma^2$

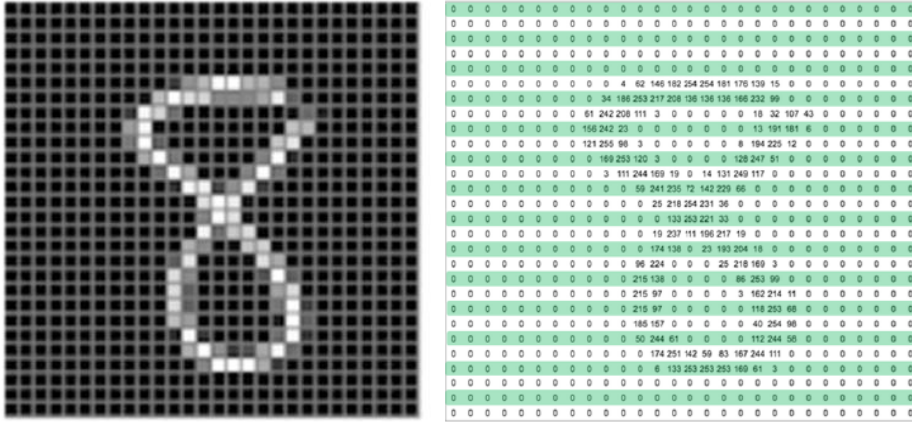
Assumption 4 : Step size α is non summable but square summable

Theorem: Suppose *Assumptions 1 to 4* holds true and let $\{\hat{\mathbf{w}}_k\}$ be the iterated generated by the algorithm for a constant step size α_k chosen such the $0 \leq \alpha_k = (1 + \mu_k)\alpha \leq \frac{\delta_1}{\delta_2^2 L}$ converges to a stationary point \mathbf{w}^* at a linear rate.

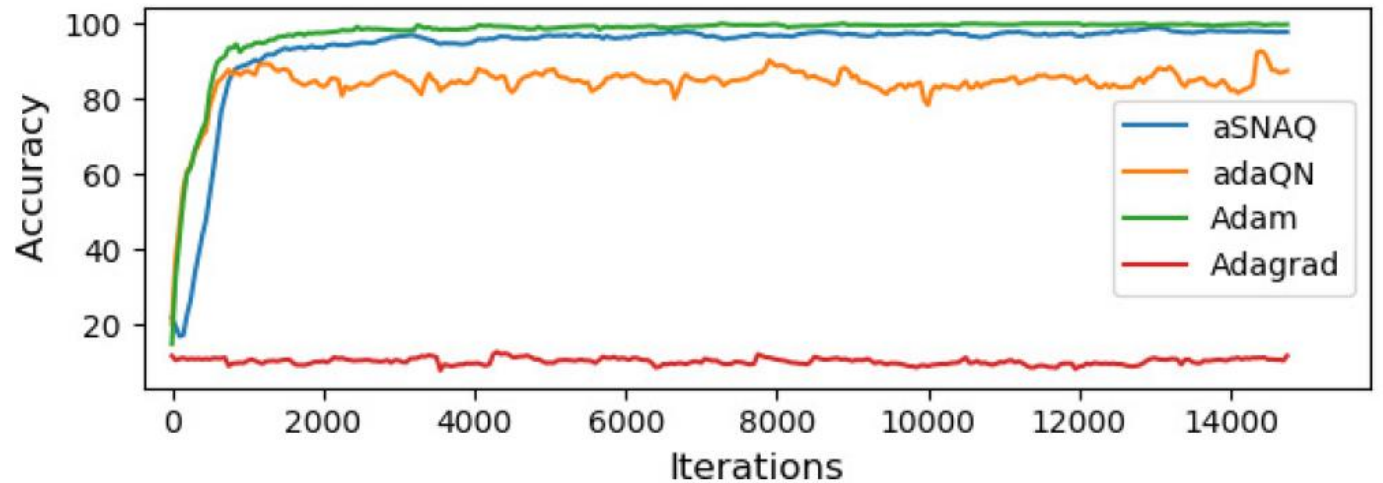
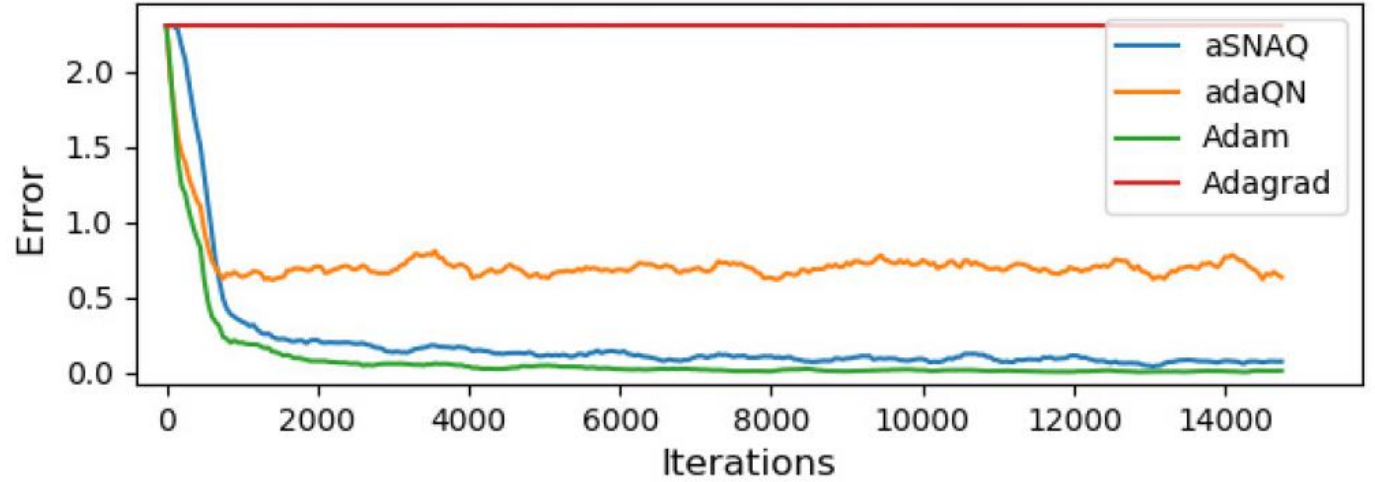
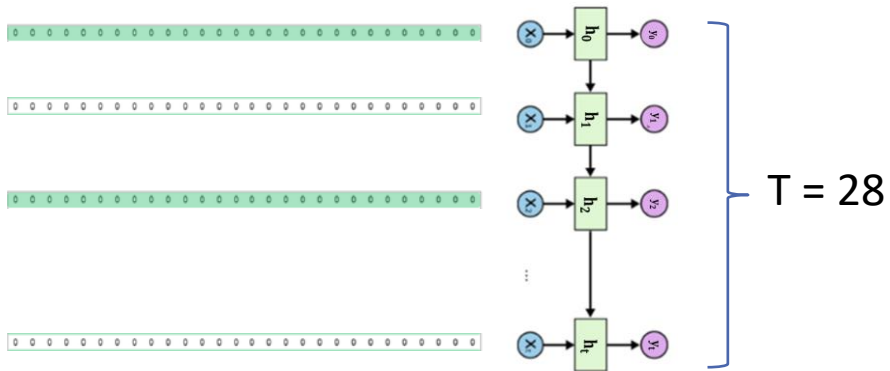
The convergence rate is given as

$$\frac{E(\hat{\mathbf{w}}_{k+1}) - E(\mathbf{w}^*)}{E(\hat{\mathbf{w}}_0) - E(\mathbf{w}^*)} \leq [1 - \alpha(1 + \mu_k)\delta_1\rho]^k$$

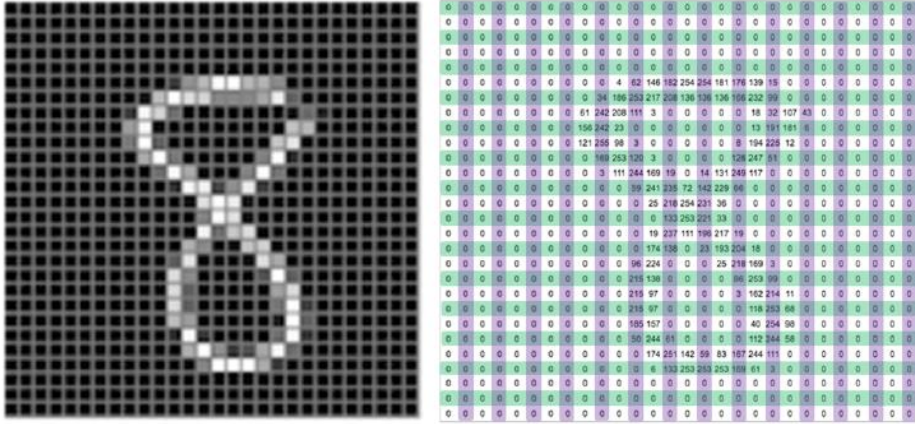
Adaptive Stochastic Nesterov's Accelerated quasi-Newton – aSNAQ



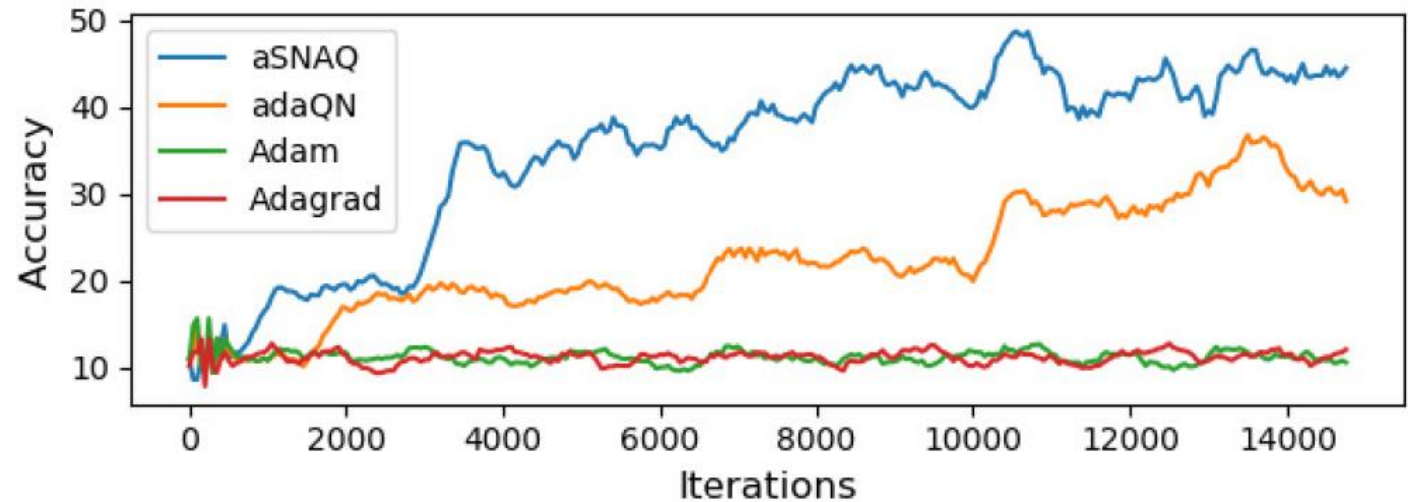
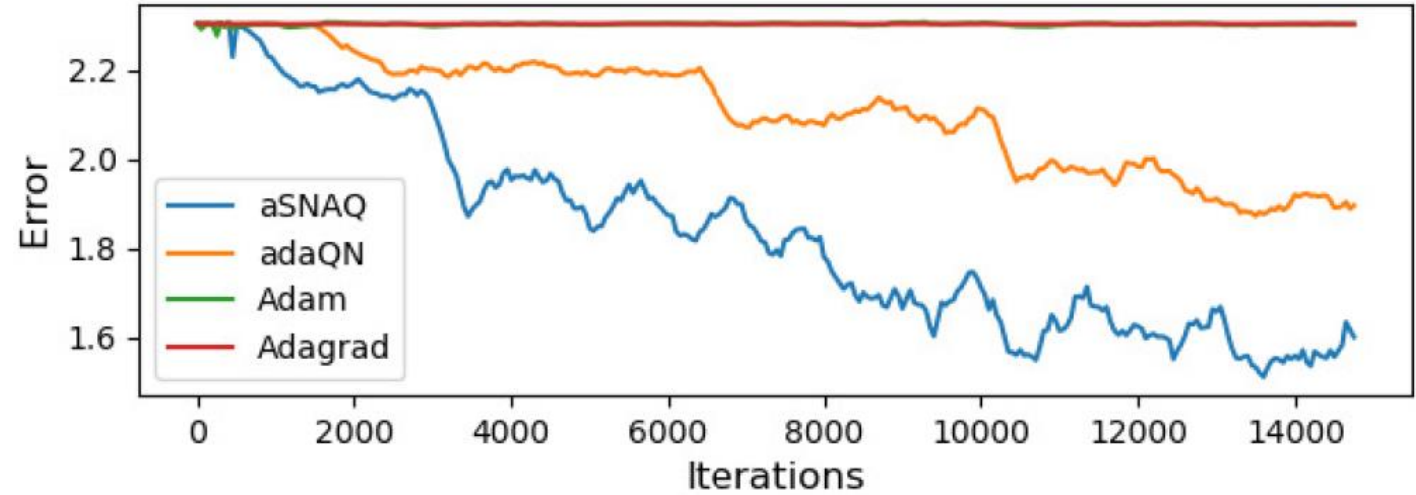
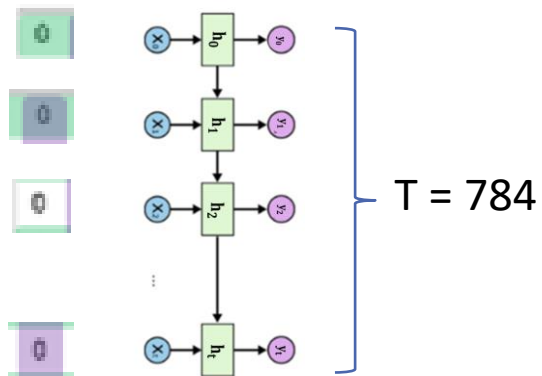
MNIST row-by-row sequencing



Adaptive Stochastic Nesterov's Accelerated quasi-Newton – aSNAQ

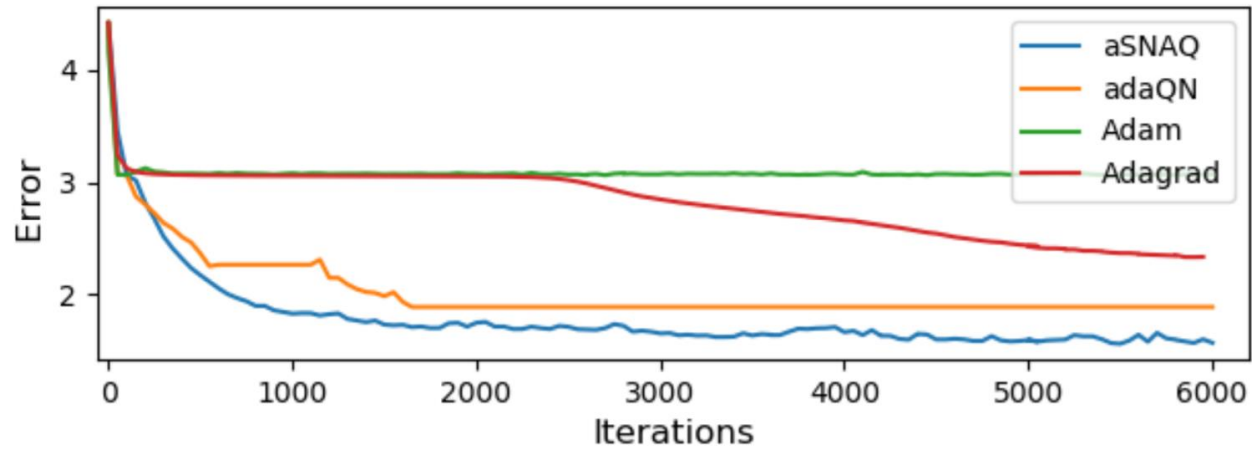


MNIST pixel-by-pixel sequence

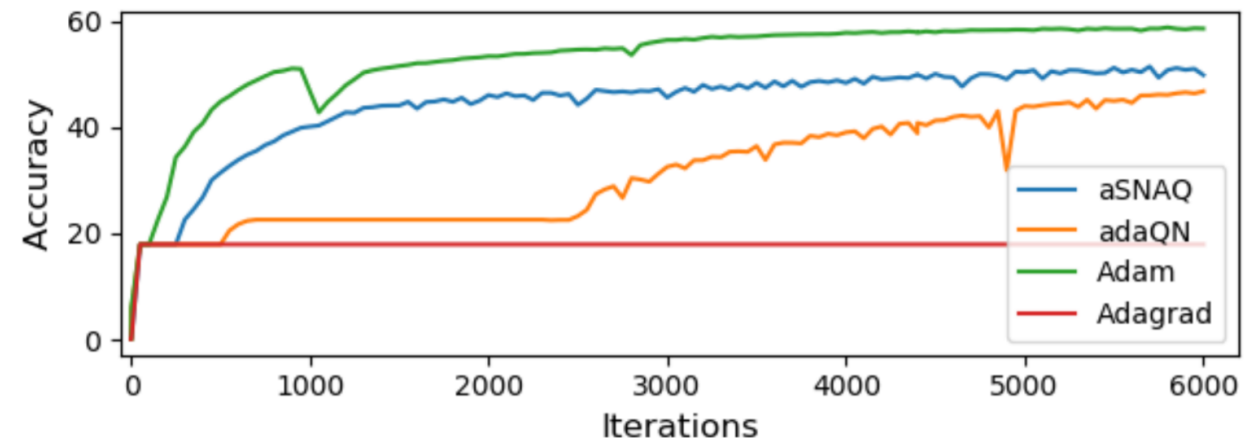
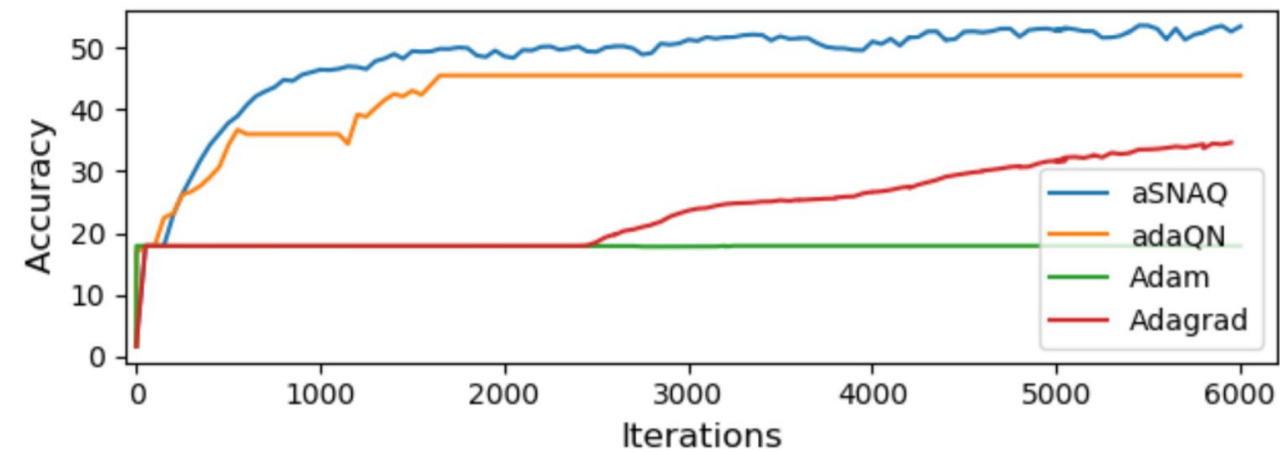
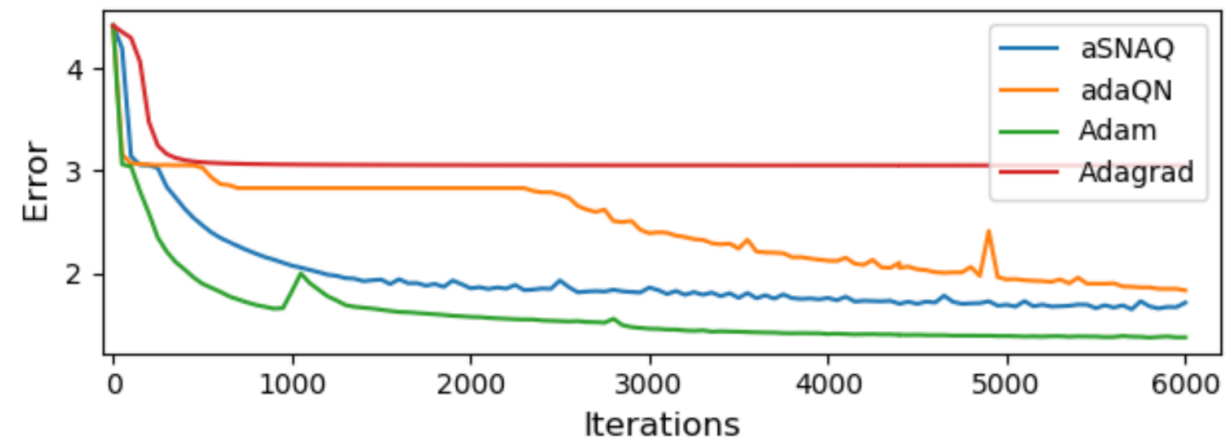


Adaptive Stochastic Nesterov's Accelerated quasi-Newton – aSNAQ

Character Level Language modeling (5-layer RNN)



Character Level Language modeling (2-layer LSTM)

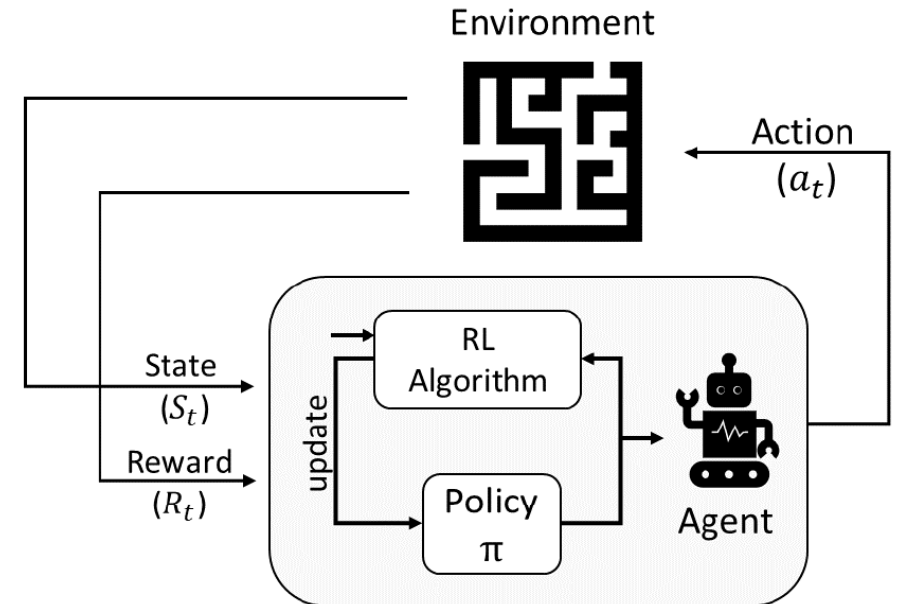


aSNAQ FOR REINFORCEMENT LEARNING

- The Reinforcement Learning problem is modelled as a **Markov's Decision Process** (MDP).
- To solve the MDP, the estimates of the value function of all possible actions is learnt using **Q-learning**, a form of temporal difference learning
- The optimal action-value function satisfies the **Bellman equation** to maximize the cumulative reward

$$Q^*(s, a) = E_{s' \sim \zeta} [R + \gamma \max_{a'} Q^*(s', a') | s, a]$$

... (Eq. 23)



aSNAQ FOR REINFORCEMENT LEARNING

In deep Q-learning, this function is optimized by a **neural network** parameterized by w .

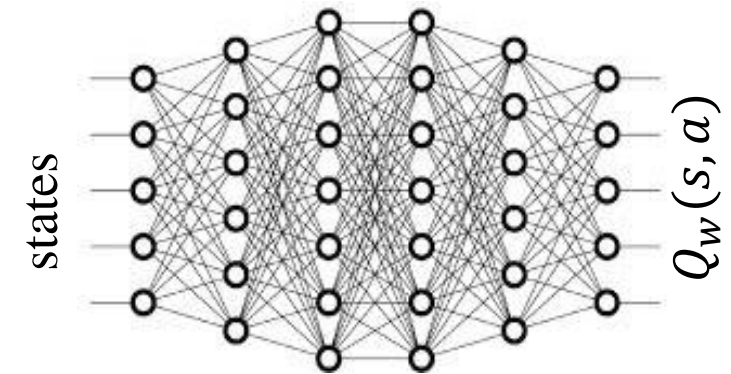
The loss function $L(w)$ is given as,

$$L(w) = E_{(s,a) \sim \zeta} [(Y - Q_w(s, a))^2] \quad \dots (Eq. 24)$$

where

$$Y = E_{s' \sim \zeta} [R + \gamma \max_{a'} Q_w(s', a')] \quad \dots (Eq. 25)$$

Bellman error $L(w)$ non-convex function



Deep Q-Network (DQN)

- The Reinforcement Learning problem is challenging to train
- Samples are a continuous stream of experiences unlike batches in supervised learning
- Makes it more prone to unlearning effective features over time

PCB ROUTING USING REINFORCEMENT LEARNING

Synthesis and physical design optimizations are the core tasks of the VLSI / ASIC design flow. *Global routing* has been a challenging problem in IC physical design.

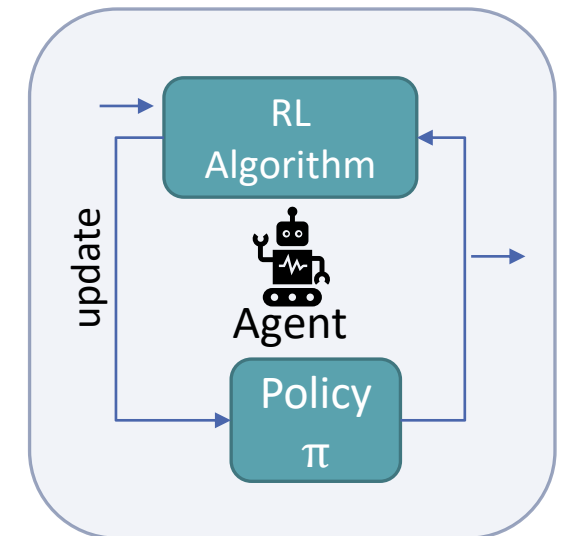
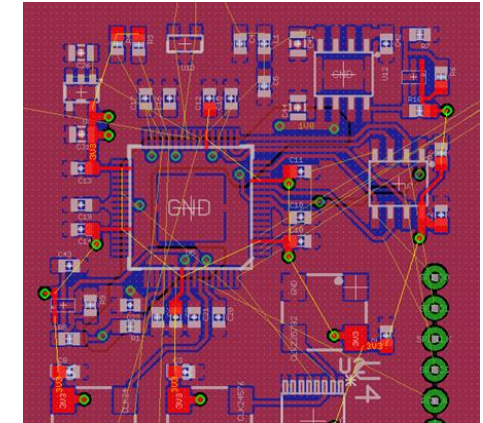
Objective

Given a netlist with the description of all the components, their connections and position, the goal of the global router is to determine the path of all the connections without violating the constraints and design rules.

- Route all pins and nets
- Minimize total wirelength (WL)
- Minimize total overflows

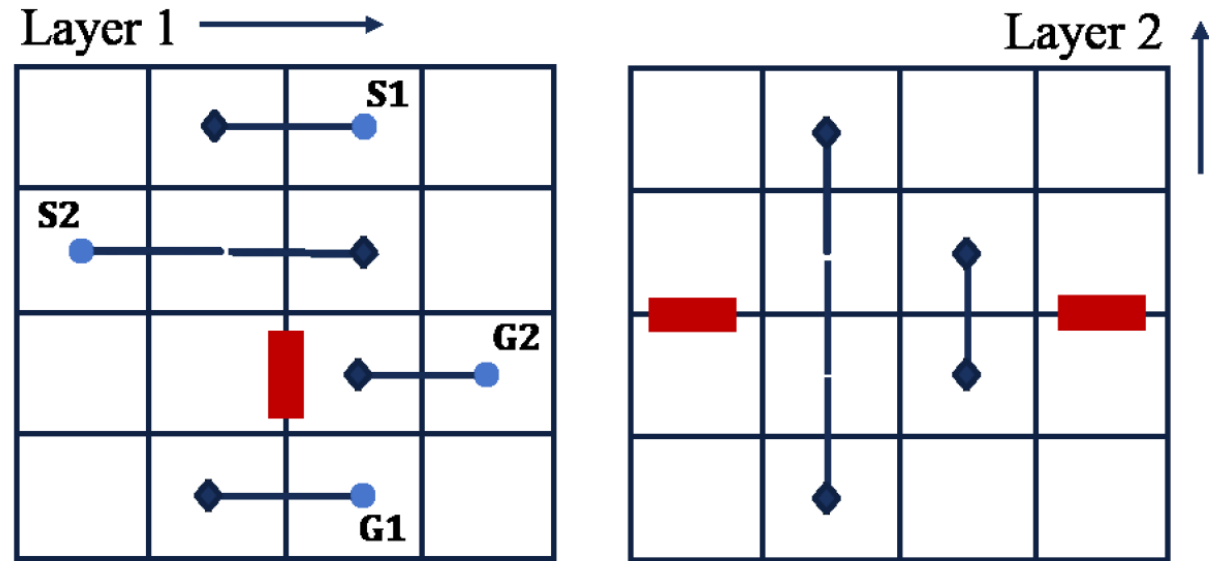
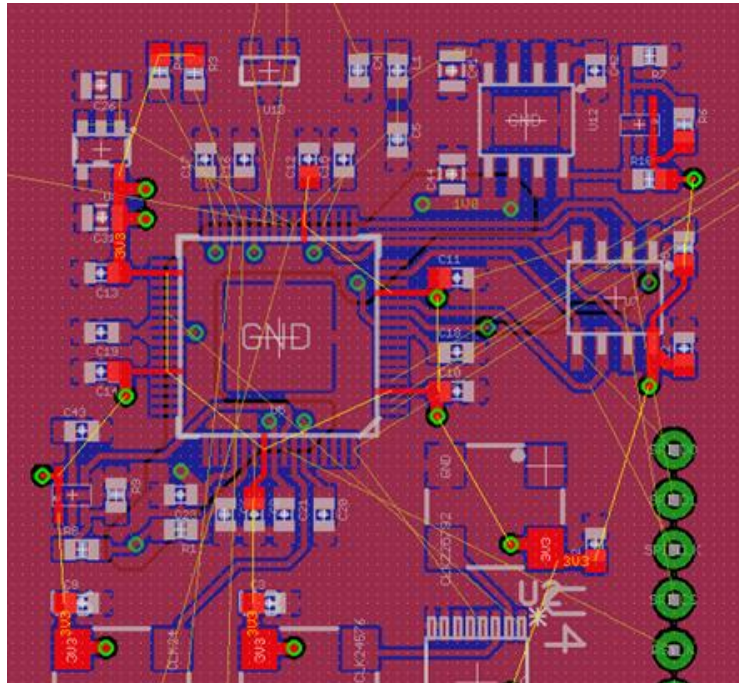
Conventional routing automation tools are usually based on analytical and path search algorithms which are **NP complete**.

S. Indrapriyadarsini, Shahrzad Mahboubi, Hiroshi Ninomiya, Takeshi Kamio, Hideki Asai, "A Nesterov's Accelerated quasi-Newton method for Global Routing using Deep Reinforcement Learning", International Symposium on Nonlinear Theory and its Applications, NOLTA, IEICE, 2020 (Student Paper Award) - (Extended paper – NOLTA journal IEICE, Jul 2021)



Objective

- Route all pins and nets
- Minimize total wirelength (WL)
- Minimize total overflows



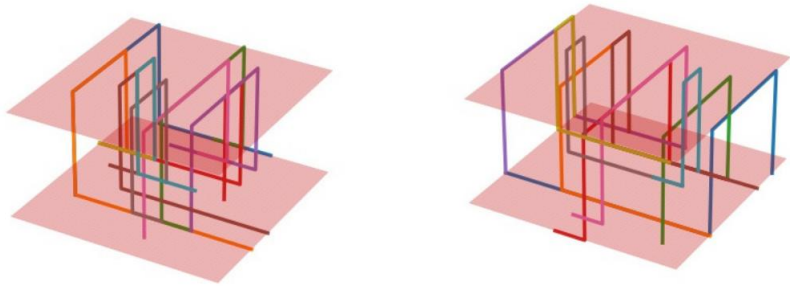
● Pin ◆ Via ■ Block — Wire

Objective function:

$$L(w) = E_{(s,a) \sim \zeta} [(Y - Q_w(s, a))^2]$$

where

$$Y = E_{s' \sim \zeta} [R + \gamma Q_{w^-}(s', \operatorname{argmax}_{a'} Q_w(s', a'))]$$



Netlist Num	A*	Adam				RMSprop				aSNAQ			
	WL	WL	diff	\mathcal{R}_{best}	\mathcal{E}_{first}	WL	diff	\mathcal{R}_{best}	\mathcal{E}_{first}	WL	diff	\mathcal{R}_{best}	\mathcal{E}_{first}
1	390	-	-	4386	-	-	-	4363	-	368	-22	4667	224
2	386	-	-	4505	-	-	-	4513	-	376	-10	4610	148
3	379	-	-	4234	-	-	-	4533	-	-	-	4382	-
4	369	348	-21	4690	228	350	-19	4685	151	345	-24	4699	75
5	366	362	-4	4679	100	361	-5	4681	135	369	+3	4656	458
6	352	348	-4	4691	337	344	-8	4697	222	335	-17	4701	106
7	430	-	-	4053	-	-	-	4322	-	-	-	4324	-
8	398	-	-	4522	-	-	-	4513	-	377	-21	4663	135
9	369	369	0	4669	225	347	-22	4687	153	348	-21	4693	64
10	366	359	-7	4674	106	375	+9	4660	223	357	-9	4683	67
11	379	380	+1	4660	174	380	+1	4658	262	-	-	4523	-
12	351	346	-5	4692	133	351	0	4689	95	348	-3	4692	58
13	395	411	+16	4616	456	397	+2	4645	180	394	-1	4640	87
14	340	343	+3	4700	57	338	-2	4706	77	341	+1	4699	49
15	375	374	-1	4659	267	384	+9	4660	211	371	-4	4668	195
16	386	355	-31	4682	279	-	-	4542	-	348	-38	4690	224
17	360	363	+3	4681	83	365	+5	4674	73	360	0	4680	56
18	343	334	-9	4705	81	333	-10	4712	84	331	-12	4714	46
19	389	385	-4	4655	164	400	+11	4615	412	377	-12	4657	239
20	337 (1)	337	0	4710	79	333	-4	4703	231	333	-4	4704	27
21	367	366	-1	4678	258	368	+1	4675	111	352	-15	4690	124
22	356 (2)	366	+10	4675	219	-	-	4528	-	-	-	4243	-
23	384	380	-4	4654	338	382	-2	4651	227	374	-10	4662	128
24	419 (2)	-	-	4303	-	-	-	4473	-	388	-31	4632	291
25	364 (2)	368	+4	4658	149	362	-2	4682	361	365	+1	4675	159
26	367 (3)	348	-19	4648	114	342	-25	4698	101	342	-25	4696	147
27	371 (2)	356	-15	4680	197	-	-	4530	-	356	-15	4682	220
28	399 (5)	-	-	4343	-	-	-	4482	-	391	-8	4640	282
29	366 (1)	360	-6	4682	104	363	-3	4677	140	362	-4	4678	96
30	387 (4)	371	-16	4669	139	380	-7	4638	390	369	-18	4666	106

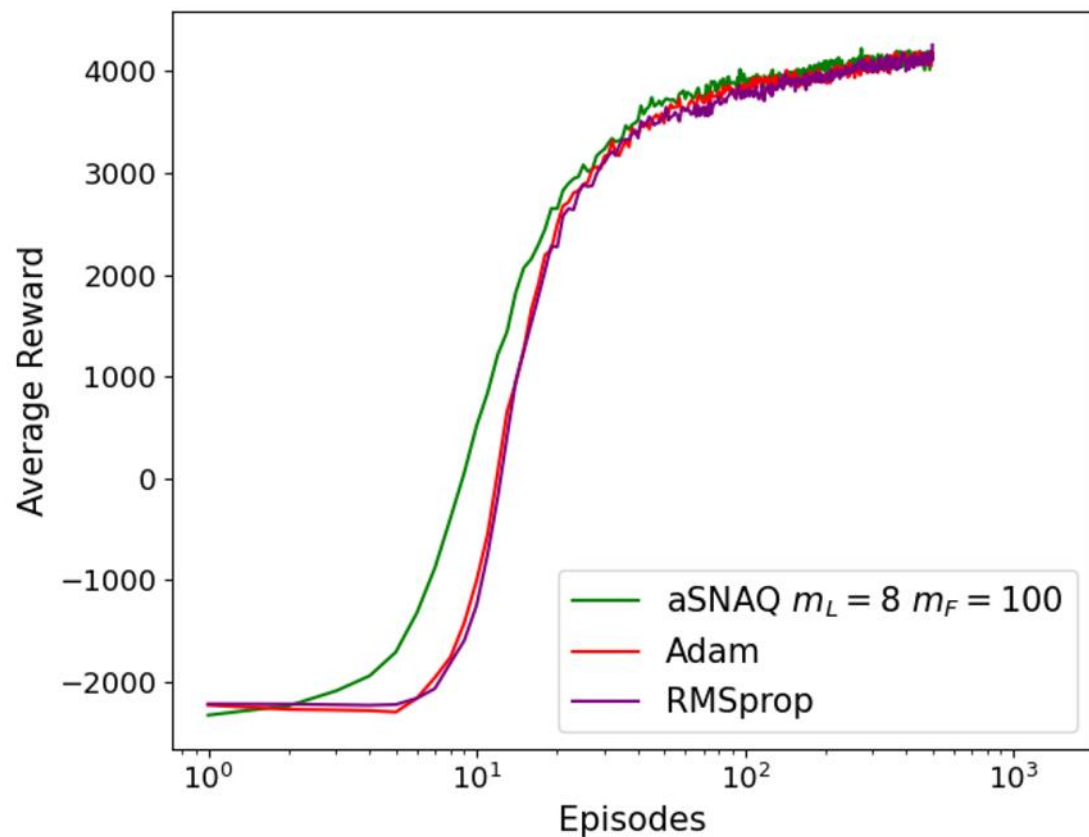
Problem set generator : H. Liao, et. al, "A deep reinforcement learning approach for global routing," Jour. Mech. Design, vol. 142(6), June 2020

- Each Netlist – 50 two-pin pairs
- Max episode $\varepsilon = 500$
- WL (\cdot) : wirelength (overflow) evaluated by ISPD'08 contest evaluator
- – indicates could not be routed within ε
- *diff* wirelength reduction compared to A*

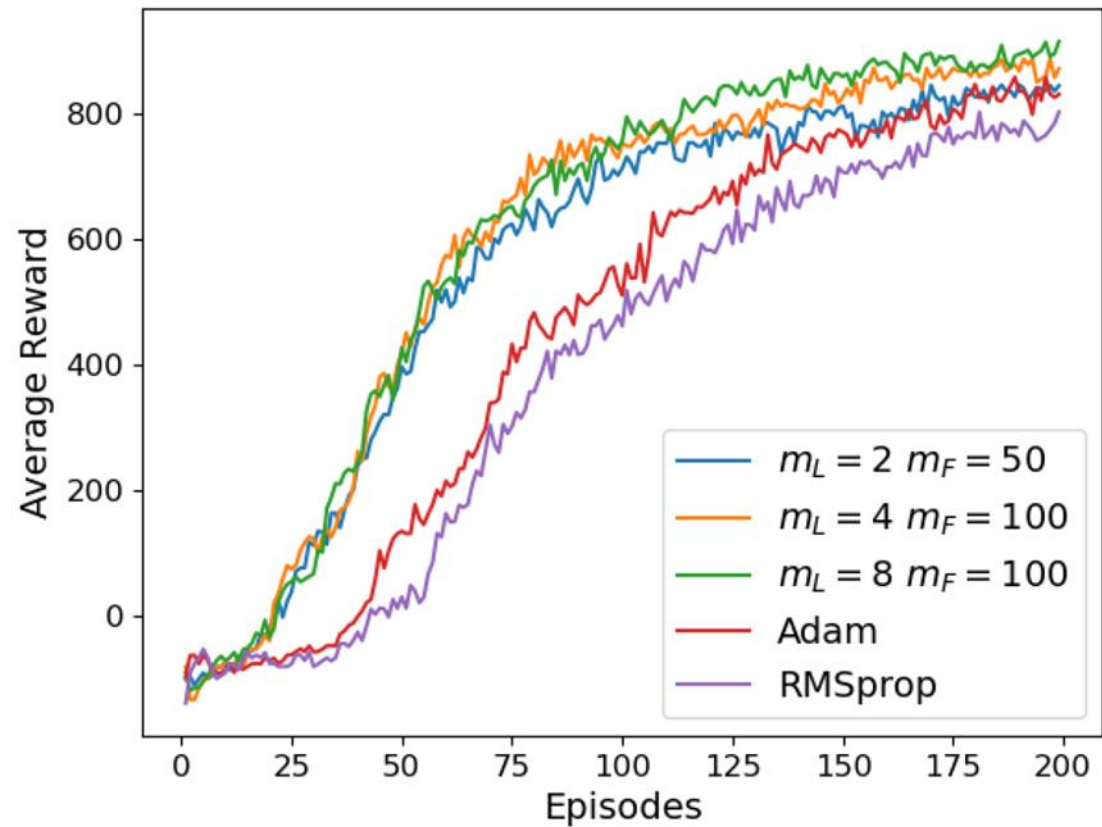
	Adam	RMSprop	aSNAQ
Success	23/30	20/30	26/30
<i>diff</i>	110	71	319

Key Takeaways

- In RL the training set is dynamically populated
- DQNs use mean-squared Bellman (non-convex function)
- Second order methods – aSNAQ show better convergence



Average reward over 30 benchmarks with 50 two-pin nets.



Average reward over 25 benchmarks with 10 two-pin nets.

OTHER QUASI-NEWTON METHOD + NESTEROV'S ACCELERATION ?

Method	$B_{k+1} =$	$H_{k+1} = B_{k+1}^{-1} =$
BFGS	$B_k + \frac{y_k y_k^T}{y_k^T \Delta x_k} - \frac{B_k \Delta x_k (B_k \Delta x_k)^T}{\Delta x_k^T B_k \Delta x_k}$	$\left(I - \frac{\Delta x_k y_k^T}{y_k^T \Delta x_k} \right) H_k \left(I - \frac{y_k \Delta x_k^T}{y_k^T \Delta x_k} \right) + \frac{\Delta x_k \Delta x_k^T}{y_k^T \Delta x_k}$
Broyden	$B_k + \frac{y_k - B_k \Delta x_k}{\Delta x_k^T \Delta x_k} \Delta x_k^T$	$H_k + \frac{(\Delta x_k - H_k y_k) \Delta x_k^T H_k}{\Delta x_k^T H_k y_k}$
Broyden family	$(1 - \varphi_k) B_{k+1}^{\text{BFGS}} + \varphi_k B_{k+1}^{\text{DFP}}, \quad \varphi \in [0, 1]$	
DFP	$\left(I - \frac{y_k \Delta x_k^T}{y_k^T \Delta x_k} \right) B_k \left(I - \frac{\Delta x_k y_k^T}{y_k^T \Delta x_k} \right) + \frac{y_k y_k^T}{y_k^T \Delta x_k}$	$H_k + \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}$
SR1	$B_k + \frac{(y_k - B_k \Delta x_k)(y_k - B_k \Delta x_k)^T}{(y_k - B_k \Delta x_k)^T \Delta x_k}$	$H_k + \frac{(\Delta x_k - H_k y_k)(\Delta x_k - H_k y_k)^T}{(\Delta x_k - H_k y_k)^T y_k}$

*Wikipedia

ACCELERATING SR1 WITH NESTEROV'S GRADIENT

- Quasi-Newton + Nesterov's acceleration **satisfies secant condition**
- The Hessian is updated using the Symmetric rank-1 update formula given as

$$\mathbf{H}_{k+1} = \mathbf{H}_k + \frac{(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)^T}{(\mathbf{s}_k - \mathbf{H}_k \mathbf{y}_k)^T \mathbf{y}_k},$$

where,

$$\mathbf{y}_k = \nabla E(\mathbf{w}_{k+1}) - \nabla E(\mathbf{w}_k + \mu_k \mathbf{v}_k) \text{ and } \mathbf{s}_k = \mathbf{w}_{k+1} - (\mathbf{w}_k + \mu_k \mathbf{v}_k)$$

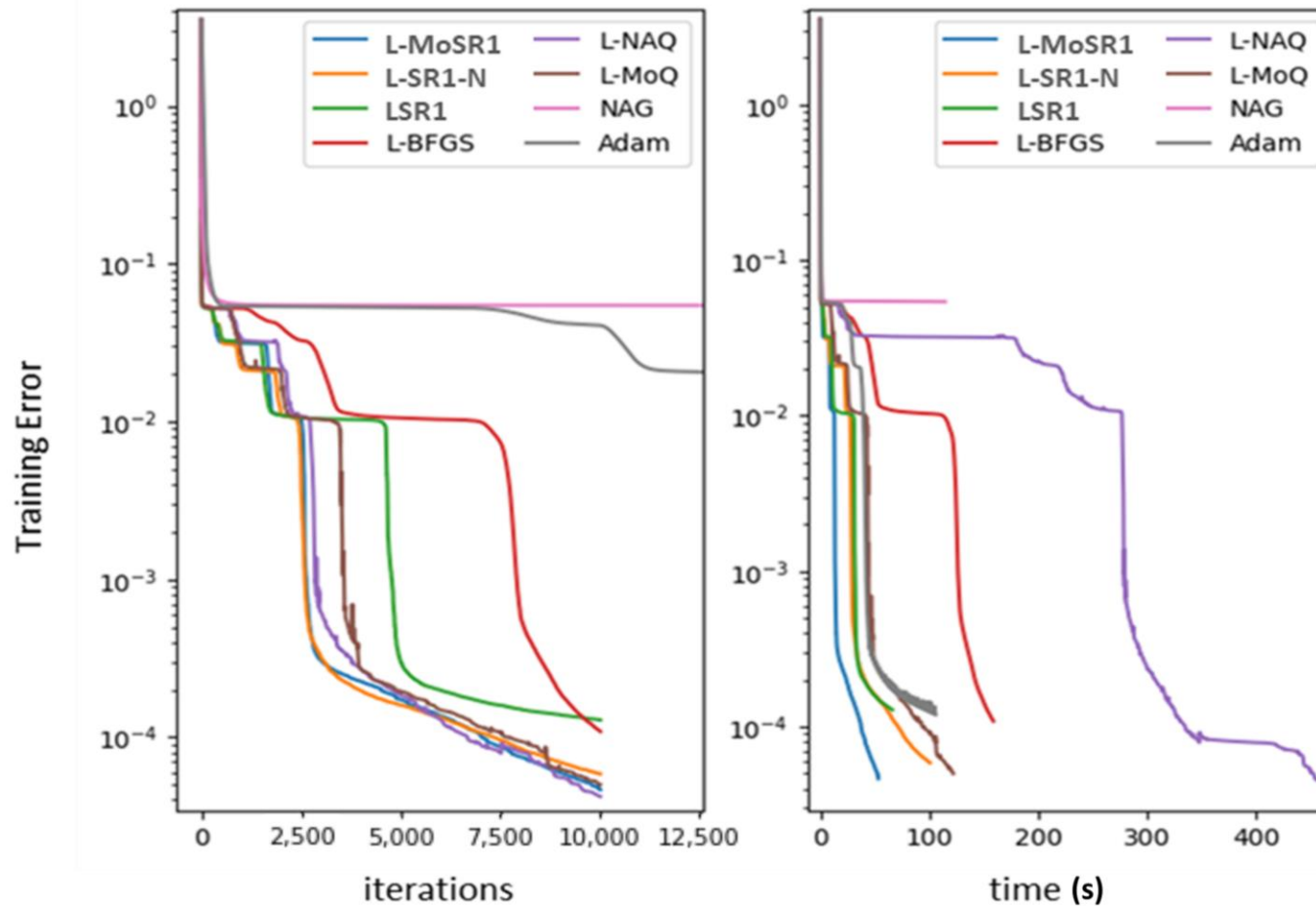
- Ensure positive semi-definiteness by performing the update only if

$$|\mathbf{s}_k^T (\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)| \geq \rho \|\mathbf{s}_k\| \|\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k\|$$

S. Indrapriyadarsini, Shahrzad Mahboubi, Hiroshi Ninomiya, Takeshi Kamio, Hideki Asai, "Accelerating Symmetric Rank 1 Quasi-Newton Method with Nesterov's Gradient", Algorithms 2022, 15(1), 6;

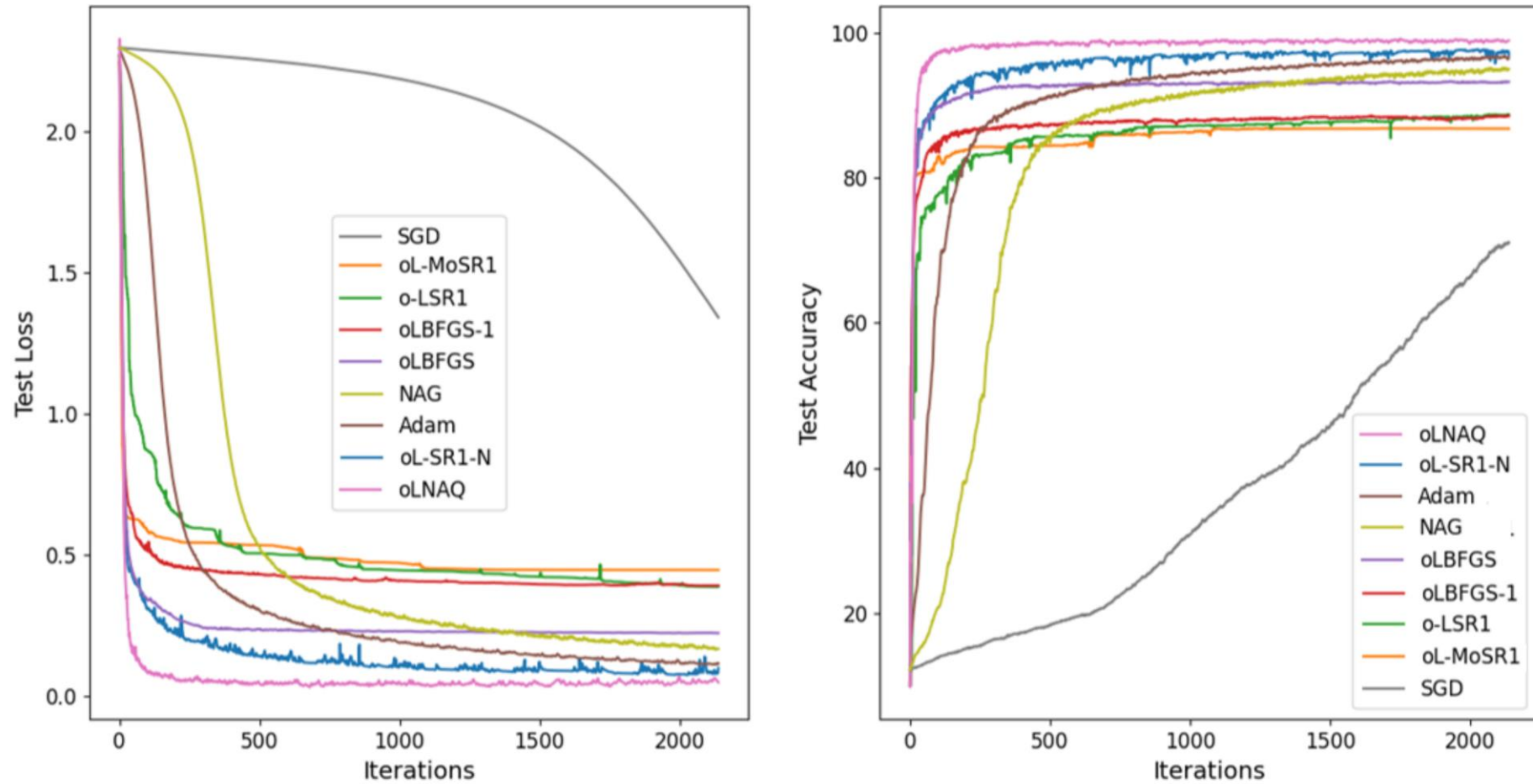
SR1 + NESTEROV'S ACCELERATION (FULL BATCH)

Average results on levy function approximation problem with $mL=10$ (full batch).



S. Indrapriyadarsini, Shahrzad Mahboubi, Hiroshi Ninomiya, Takeshi Kamio, Hideki Asai, "Accelerating Symmetric Rank 1 Quasi-Newton Method with Nesterov's Gradient", Algorithms 2022, 15(1), 6;

SR1 + NESTEROV'S ACCELERATION (STOCHASTIC)



Results of MNIST on LeNet-5 architecture with $b=256$ and $mL=8$.

S. Indrapriyadarsini, Shahrzad Mahboubi, Hiroshi Ninomiya, Takeshi Kamio, Hideki Asai, "Accelerating Symmetric Rank 1 Quasi-Newton Method with Nesterov's Gradient", *Algorithms* 2022, 15(1), 6;

CONVERGENCE ANALYSIS

Assumption 1: The sequence of iterates \mathbf{w}_k and $\hat{\mathbf{w}}_k$ remains in the closed and bounded set Ω on which the objective function is twice continuously differentiable and has Lipschitz continuous gradient, i.e. there exists a constant $L > 0$ such that

$$\|\nabla E(\mathbf{w}_{k+1}) - \nabla E(\hat{\mathbf{w}}_k)\| \leq L \|\mathbf{w}_{k+1} - \hat{\mathbf{w}}_k\| \quad \forall \mathbf{w}_{k+1}, \hat{\mathbf{w}}_k \in \mathbb{R}^d$$

If *Assumption 1* holds true, then it implies that the objective function satisfies,

$$E(\mathbf{w}_{k+1}) \leq E(\mathbf{w}_k + \mu \mathbf{v}_k) + \nabla E(\mathbf{w}_k + \mu \mathbf{v}_k)^T \mathbf{d} + \frac{L}{2} \|\mathbf{w}_{k+1} - (\mathbf{w}_k + \mu \mathbf{v}_k)\|_2^2$$

Assumption 2: The Hessian matrix is bounded and well-defined, i.e, there exists constants ρ and M , such that

$$\rho \leq \|\mathbf{B}_k\| \leq M \quad \forall k$$

and for each iteration

$$|\mathbf{s}_k^T (\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k)| \geq \rho \|\mathbf{s}_k\| \|\mathbf{y}_k - \mathbf{B}_k \mathbf{s}_k\|$$

Assumption 2 ensures Hessian matrix is symmetric positive semidefinite and bounded

S. Indrapriyadarsini, Shahrzad Mahboubi, Hiroshi Ninomiya, Takeshi Kamio, Hideki Asai, "Accelerating Symmetric Rank 1 Quasi-Newton Method with Nesterov's Gradient", Algorithms 2022, 15(1), 6;

CONVERGENCE ANALYSIS

Assumption 3: Let \mathbf{B}_k be any $n \times n$ symmetric matrix and \mathbf{s}_k be an optimal solution to the trust region subproblem,

$$\min_{\mathbf{d}} m_k(\mathbf{d}) = E(\hat{\mathbf{w}}_k) + \mathbf{d}^T \nabla E(\hat{\mathbf{w}}_k) + \frac{1}{2} \mathbf{d}^T \mathbf{B}_k \mathbf{d},$$

where $\hat{\mathbf{w}}_k + \mathbf{d}$ lies in the trust region. Then for all $k \geq 0$,

$$|\nabla E(\hat{\mathbf{w}}_k)^T \mathbf{s}_k + \frac{1}{2} \mathbf{s}_k^T \mathbf{B}_k \mathbf{s}_k| \geq \frac{1}{2} \|\nabla E(\hat{\mathbf{w}}_k)\| \min \left\{ \Delta_k, \frac{\|\nabla E(\hat{\mathbf{w}}_k)\|}{\|\mathbf{B}_k\|} \right\}$$

Assumption 3 ensures that the subproblem solved by the trust region method is sufficiently optimal at each iteration.

CONVERGENCE ANALYSIS

Lemma : If Assumptions 1 to 3 holds true, and \mathbf{s}_k be an optimal solution to the trust region subproblem, and if the initial Hessian $\mathbf{H}_{k+1} = \gamma_k$ is bounded (i.e., $\mathbf{0} \leq \gamma_k \leq \hat{\gamma}_k$) then for all $k \geq 0$, the Hessian update given by the SR1+N algorithm is bounded

$$\|\mathbf{B}_{k+1}\| \leq \left(1 + \frac{1}{\rho}\right)^{m_L} \gamma_k + \left[\left(1 + \frac{1}{\rho}\right)^{m_L} - 1 \right] M$$

Theorem : Given a level set $\Omega = \{\mathbf{w} \in \mathbb{R}^d : E(\mathbf{w}) < E(\mathbf{w}_0)\}$ that is bounded, let $\{\mathbf{w}_k\}$ be the sequence of iterates generated by the SR1+N algorithm. If Assumptions 1 to 3 holds true, then,

$$\lim_{k \rightarrow \infty} \|\nabla E(\mathbf{w}_k)\| = 0.$$

CONCLUSION

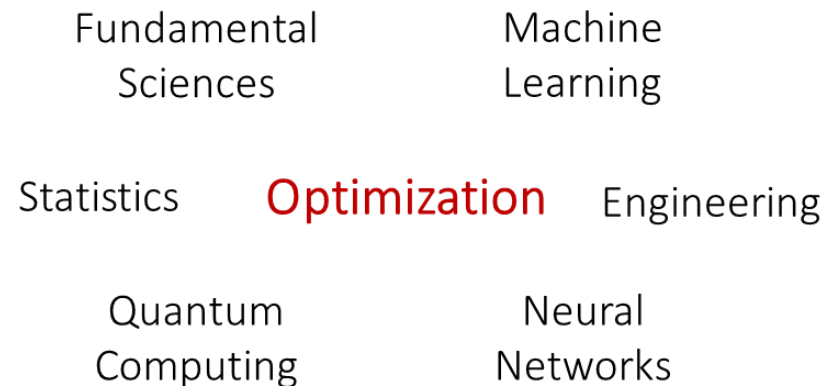
- Second order methods show better convergence compared to first order methods in training neural networks.
- Proposed a family of stochastic Nesterov and momentum accelerated second order methods for training neural networks
 - $o(L)$ NAQ / $o(L)$ MoQ → accelerate conventional $o(L)$ BFGS method, stochastic
 - aSNAQ → RNNs, deep Q-networks
 - L-SR1-N → confirmed feasibility of Nesterov's acceleration in rank-1 methods

CONCLUSION

- Empirically demonstrated the robustness and efficiency of Nesterov and momentum accelerated quasi-Newton methods in training neural networks
 - oLNAQ and oLMoQ $>$ o(L)BFGS, Adam, SGD
 - aSNAQ \rightarrow efficient in training long sequence RNNs
 - L-SR1-N \rightarrow improved the performance of L-SR1 to the level of rank-2 methods.
- Provided theoretical analysis on convergence of stochastic Nesterov and momentum accelerated NAQ and MoQ methods and analyzed computational cost
 - Limited memory forms showed computational cost in the order $O(d)$
 - Convergence guarantees with at least linear rate.

FUTURE WORKS

- Parallel and distributed implementations, with efficient techniques to reduce overheads
 - R. Anil, et. al, “Scalable second order optimization for deep learning,” *arXiv preprint arXiv:2002.09018*, 2020.
 - Y. Fei, et. al, “Parallel l-bfgs-b algorithm on gpu,” *Computers & graphics*, vol. 40, pp. 1–9, 2014.
 - W. Chen, et. al, “Large-scale l-bfgs using mapreduce,” *Advances in neural information processing systems*, vol. 27, 2014.
- Reduce sampling noise with overlapping and multi-batch strategies
 - K. Crammer, A. Kulesza, and M. Dredze, “Adaptive regularization of weight vectors,” *Advances in neural information processing systems*, vol. 22, 2009.



THANK YOU