2020년 9월 25일

Stochastic Gradient Descent

Data Mining & Quality Analytics Lab

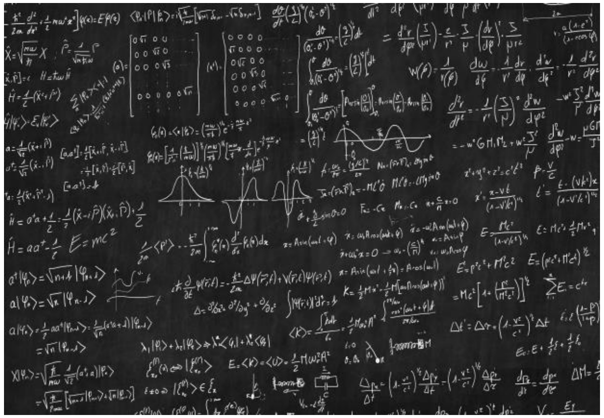
발표자 : 배진수

Presenter

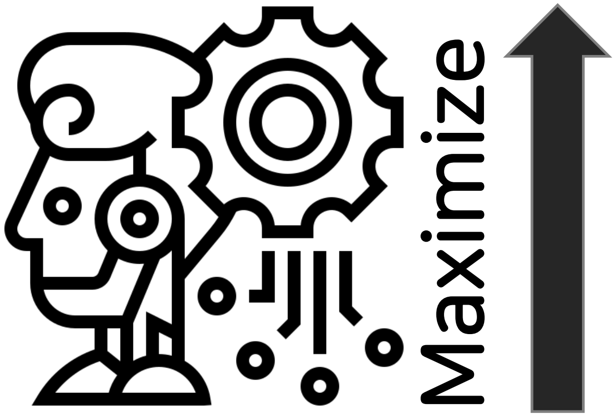
❖ 배진수(Jinsoo Bae)

- Korea University Data Mining & Quality Analytics Lab
- Ph. D Student (2020.03 ~ Present)
- Interface Between Machine learning and Optimization

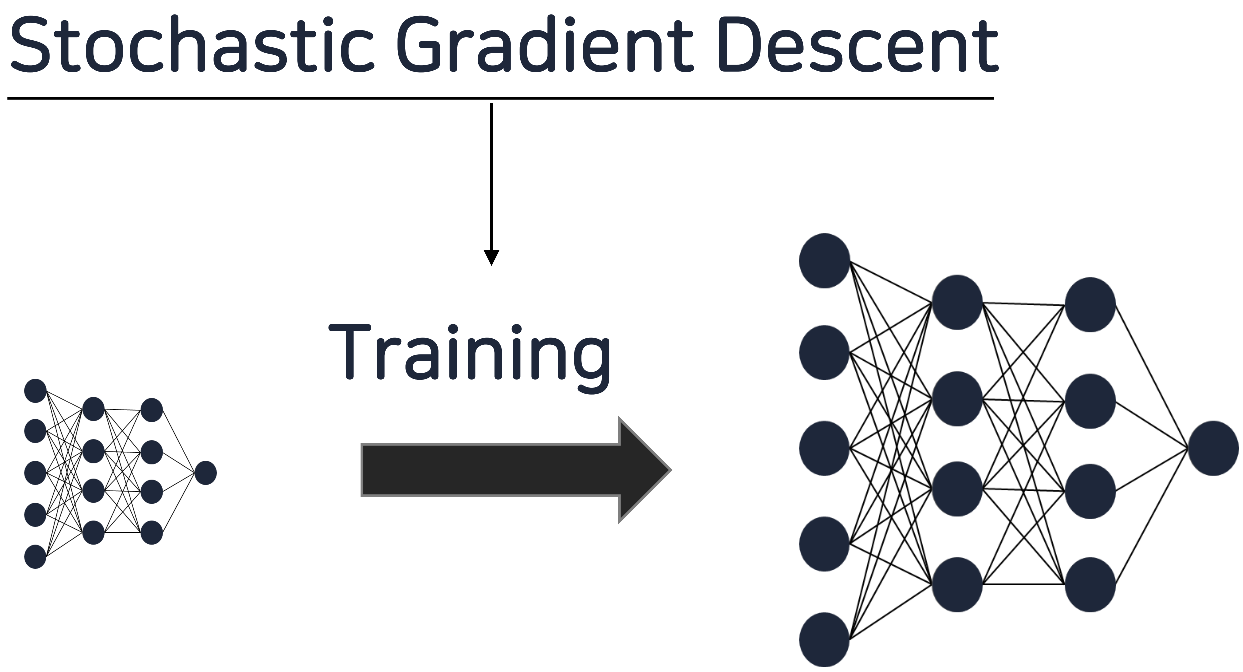
Math, statistical science



Machine learning and Optimization



Today



Today

Stochastic Gradient Descent

Adam


ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Diederik P. Kingma* — University of Amsterdam, OpenAI
Jimmy Lei Ba* — University of Toronto

ABSTRACT

1 INTRODUCTION

Lookahead


Lookahead Optimizer: k steps forward, 1 step back

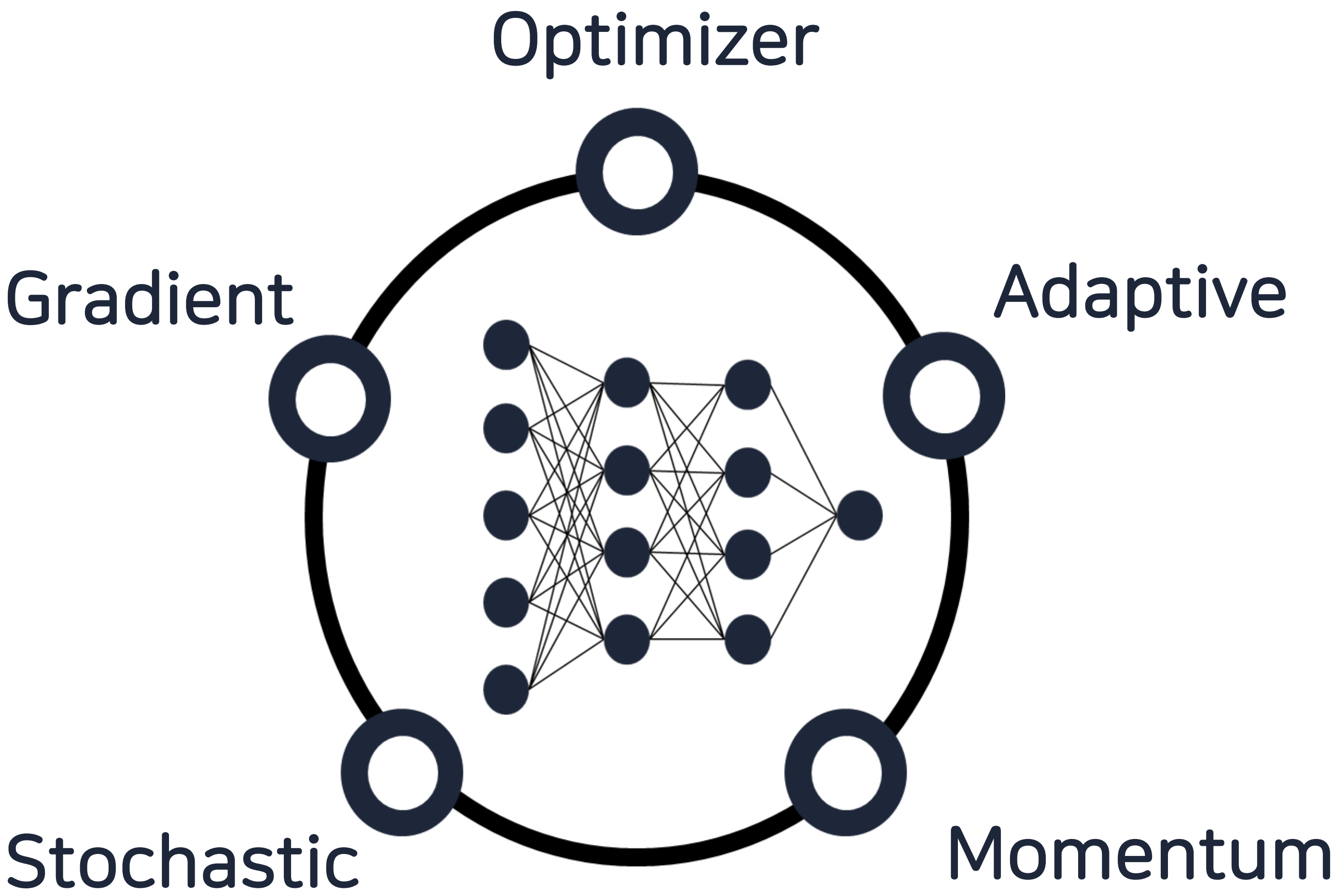
Michael R. Zhang, James Lucas, Geoffrey Hinton, Jimmy Ba
Department of Computer Science, University of Toronto, Vector Institute

Abstract

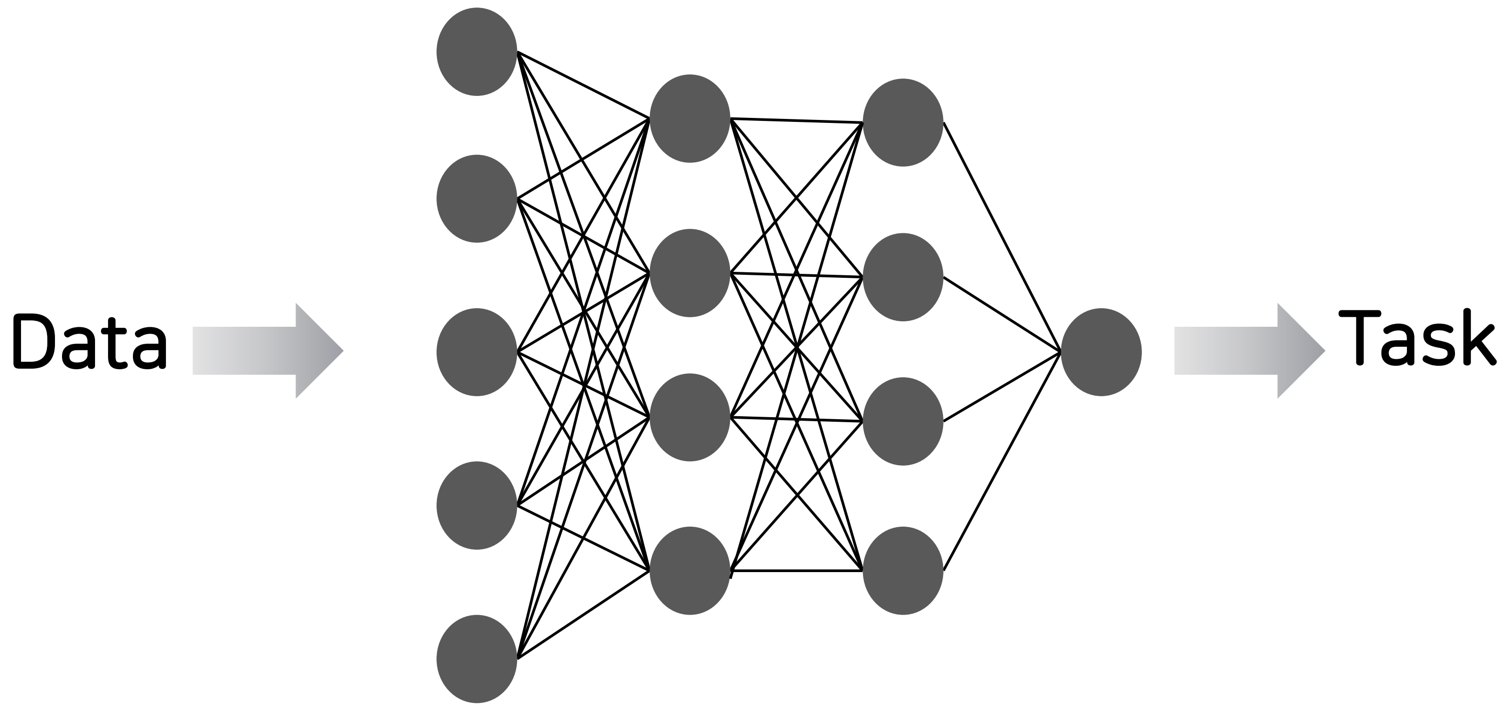
1 Introduction

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

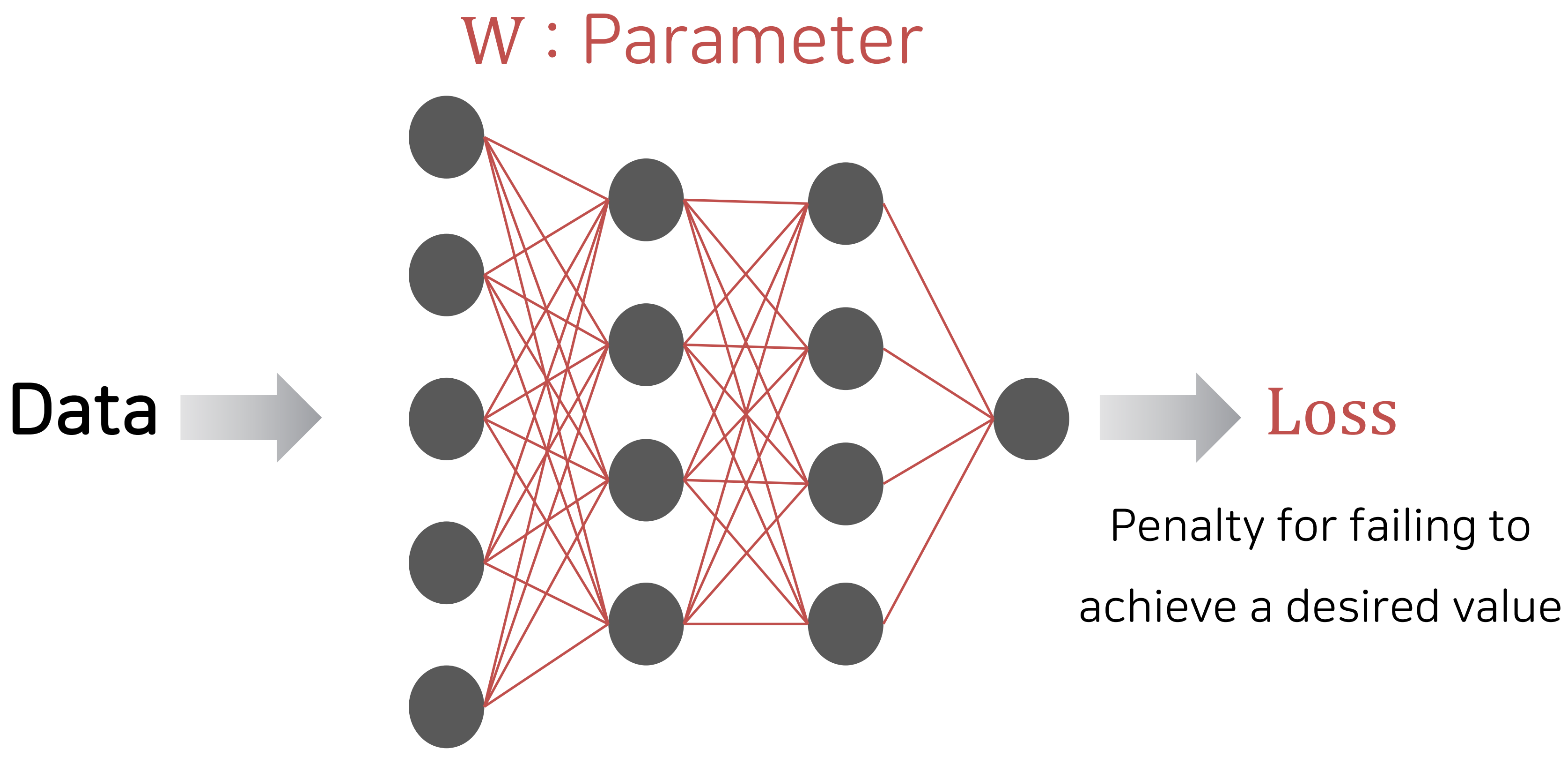
Keyword



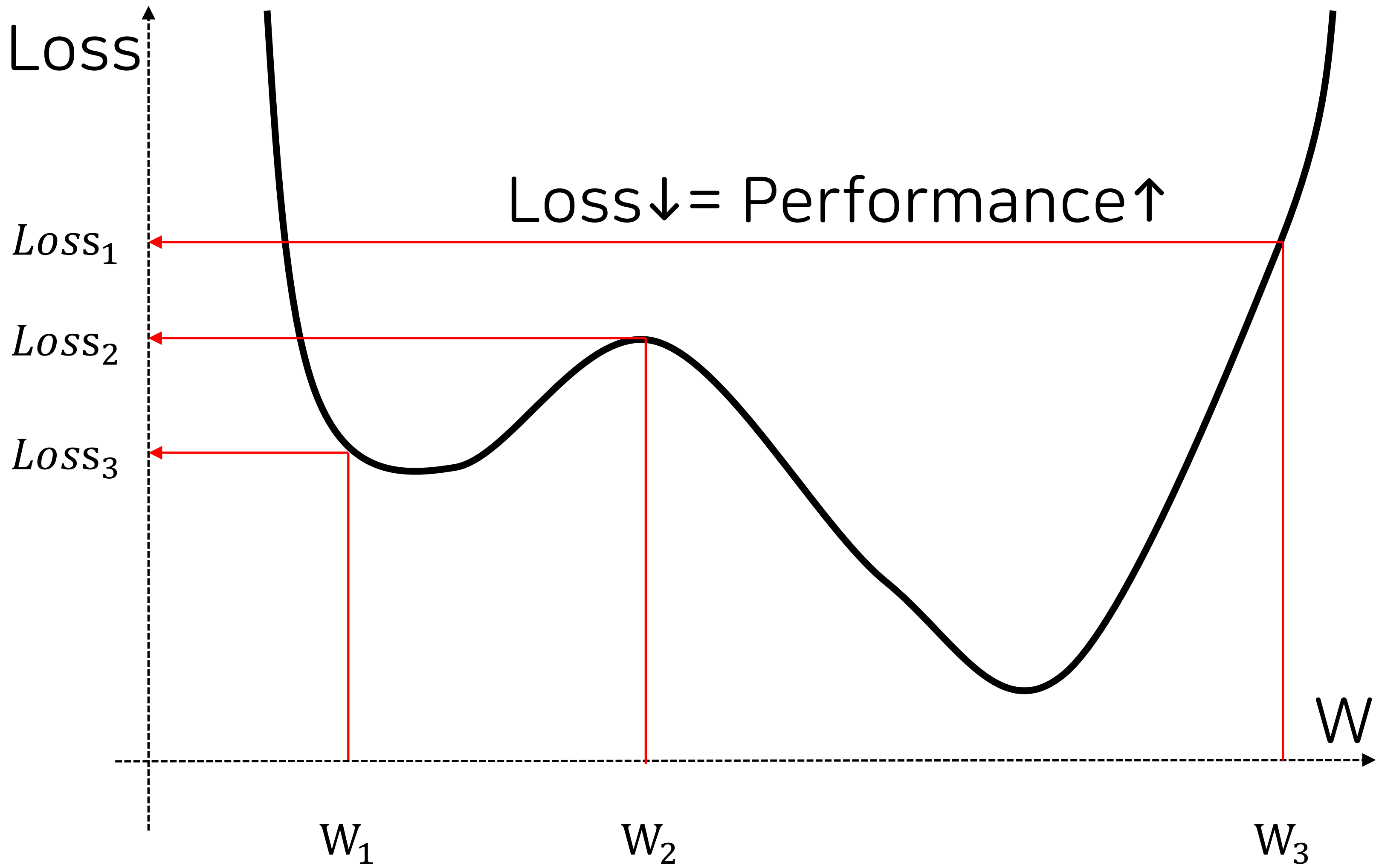
Optimizer



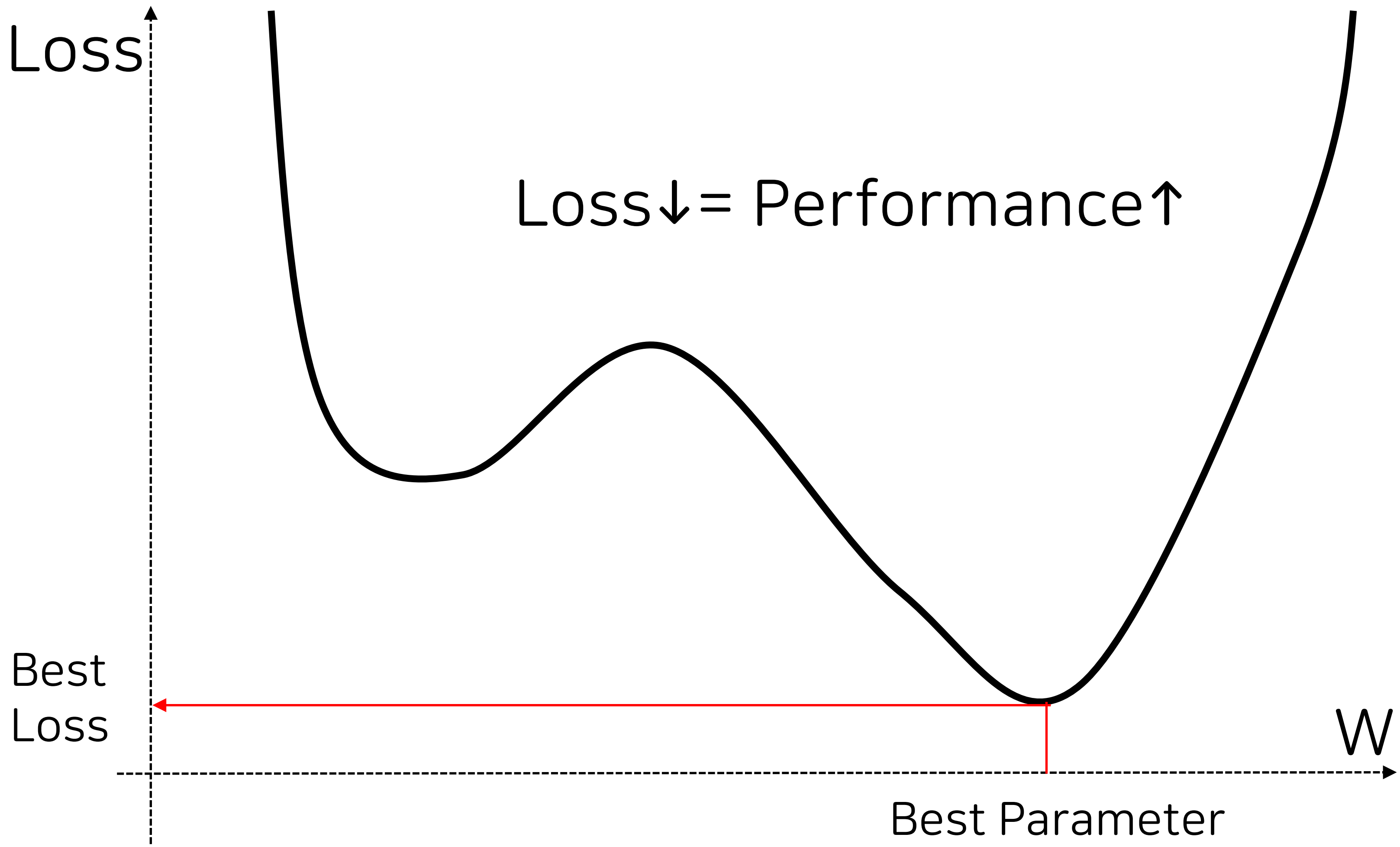
Optimizer


W : Parameter
Data
Loss
Penalty for failing to
achieve a desired value

Optimizer



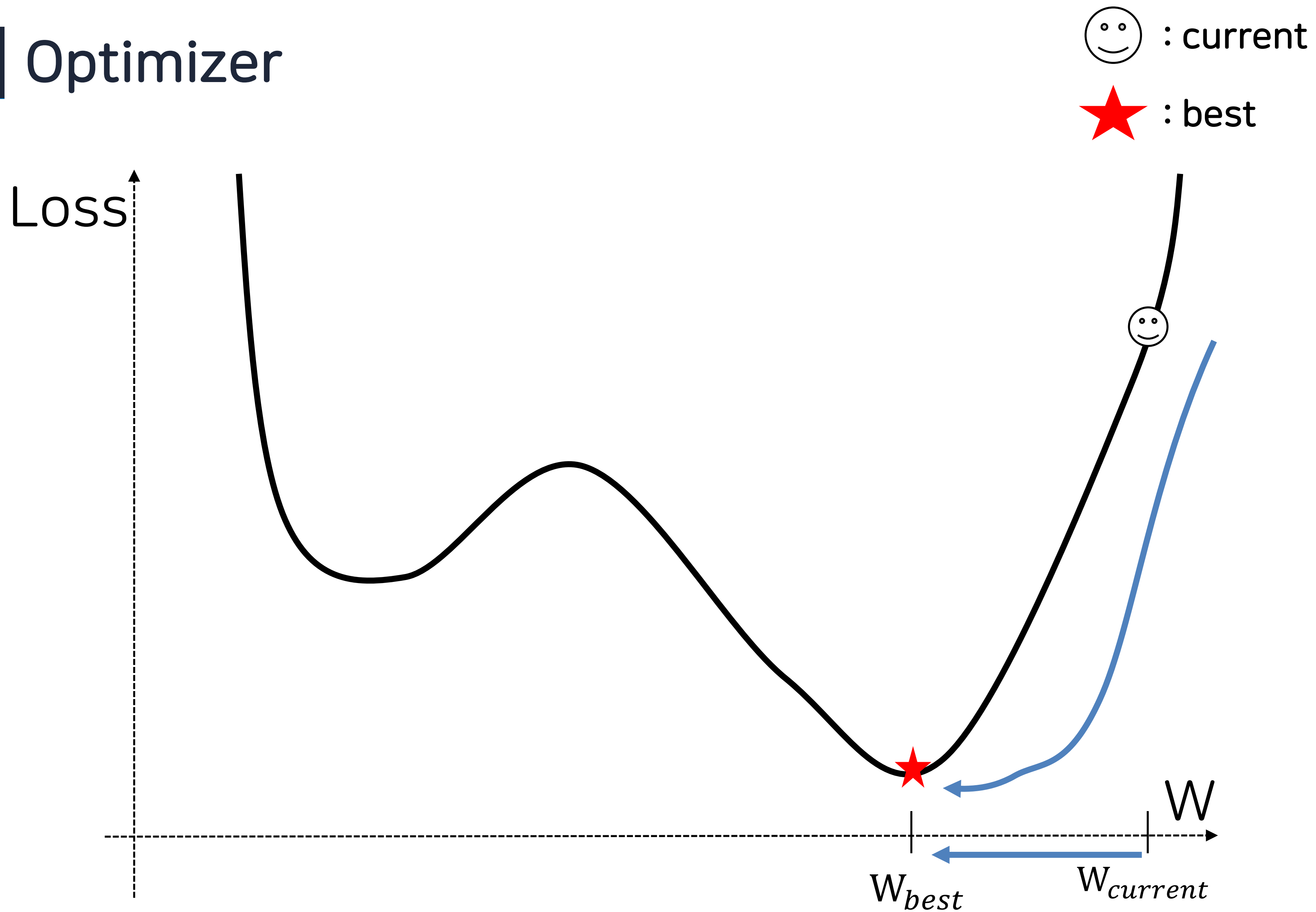
Optimizer



Optimizer

- $\underset{W}{\text{minimize}}$ Loss
- find $\underset{W}{\text{argmin}}$ Loss
- Training
- Who ? :Optimizer

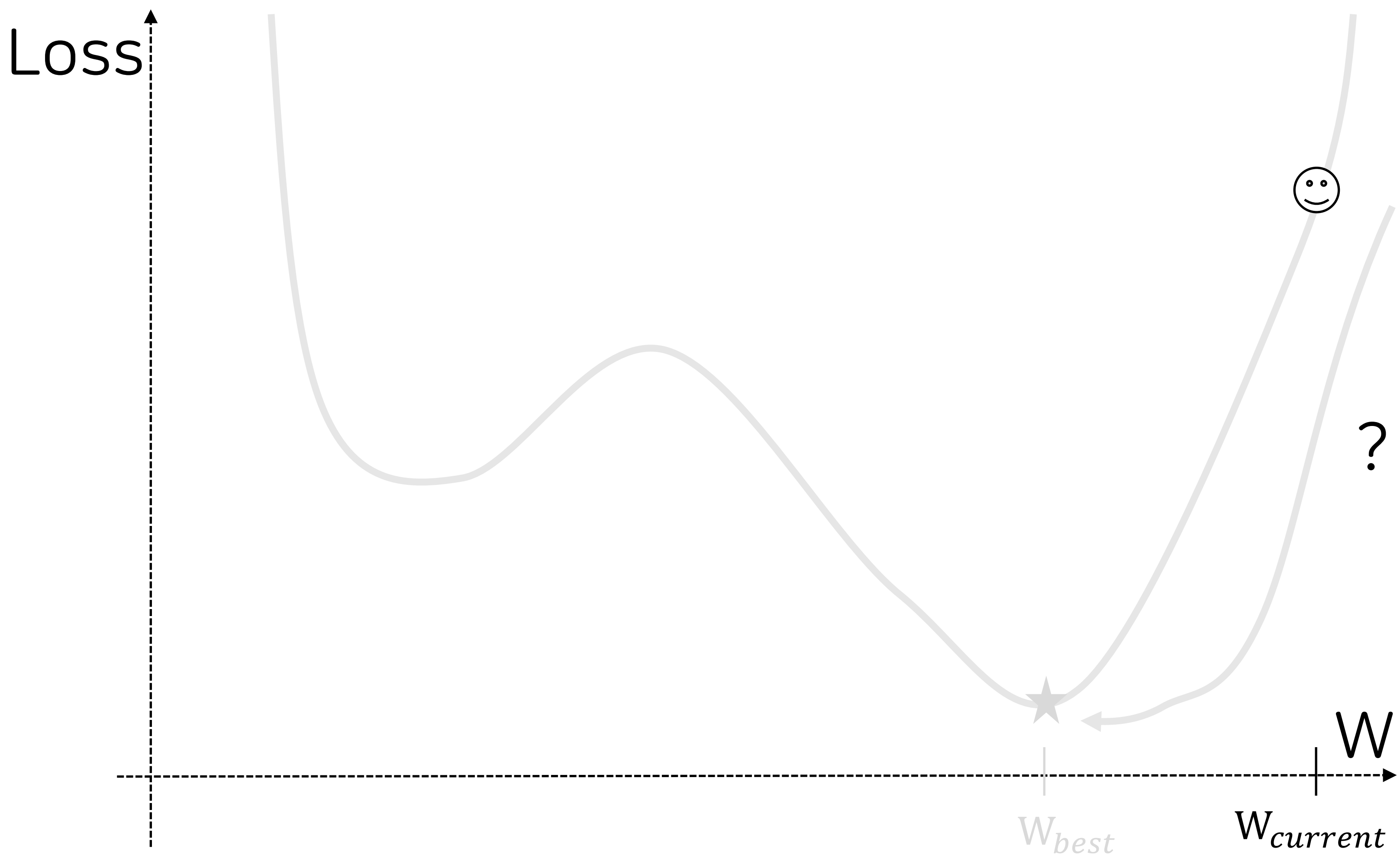
Optimizer



Gradient

: Unknown

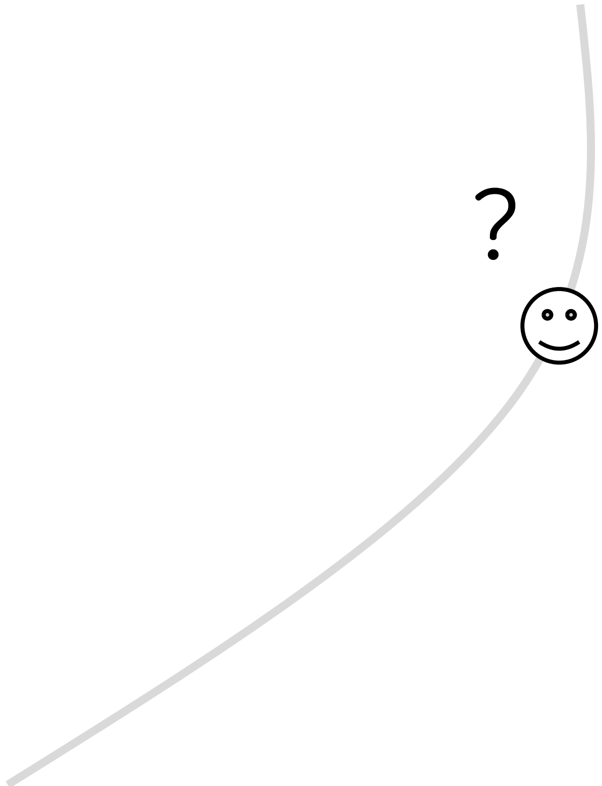
: Known



Gradient

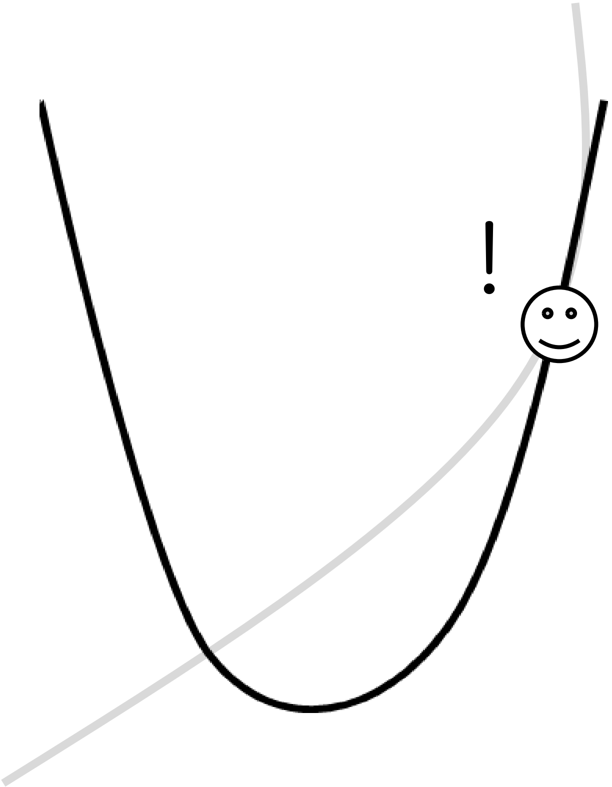
: Unknown

: Known



Brook Taylor

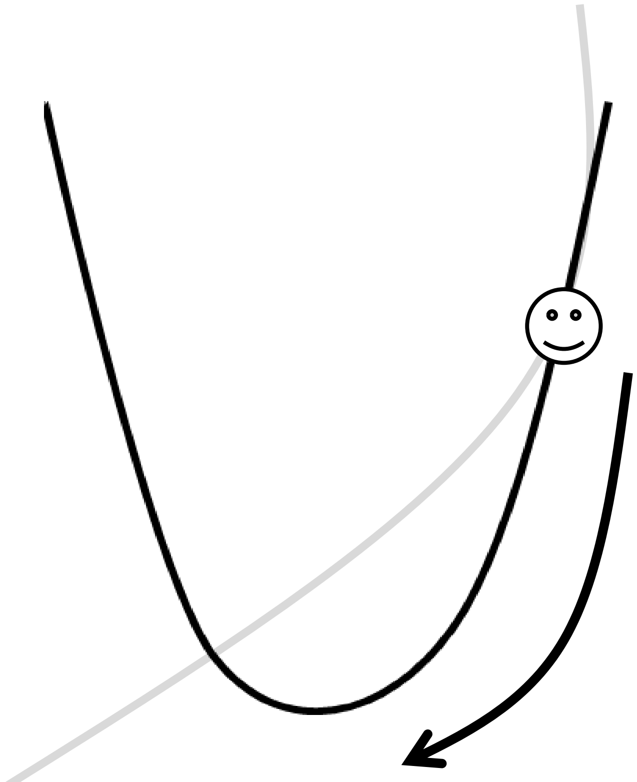
Gradient



Quadratic approximation

Brook Taylor

Gradient



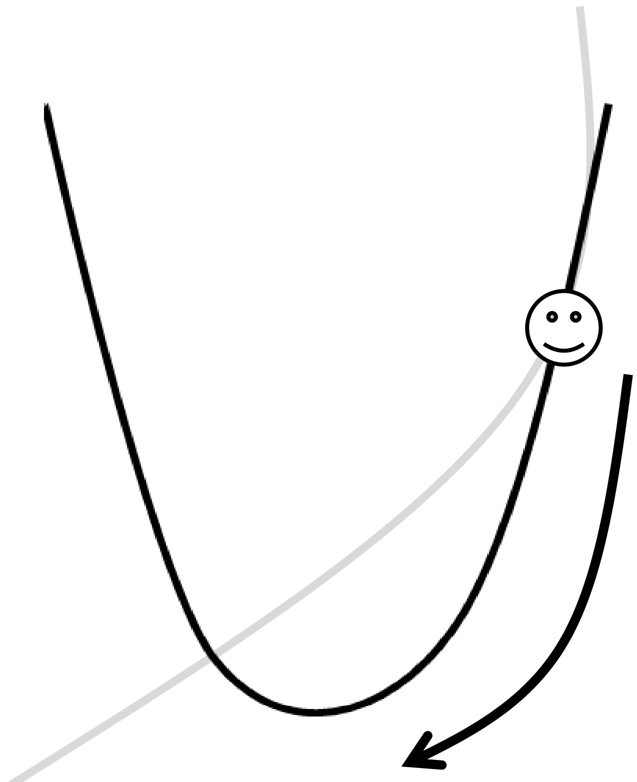
Gradient Descent

경사 하강법

Gradient

: Unknown

: Known

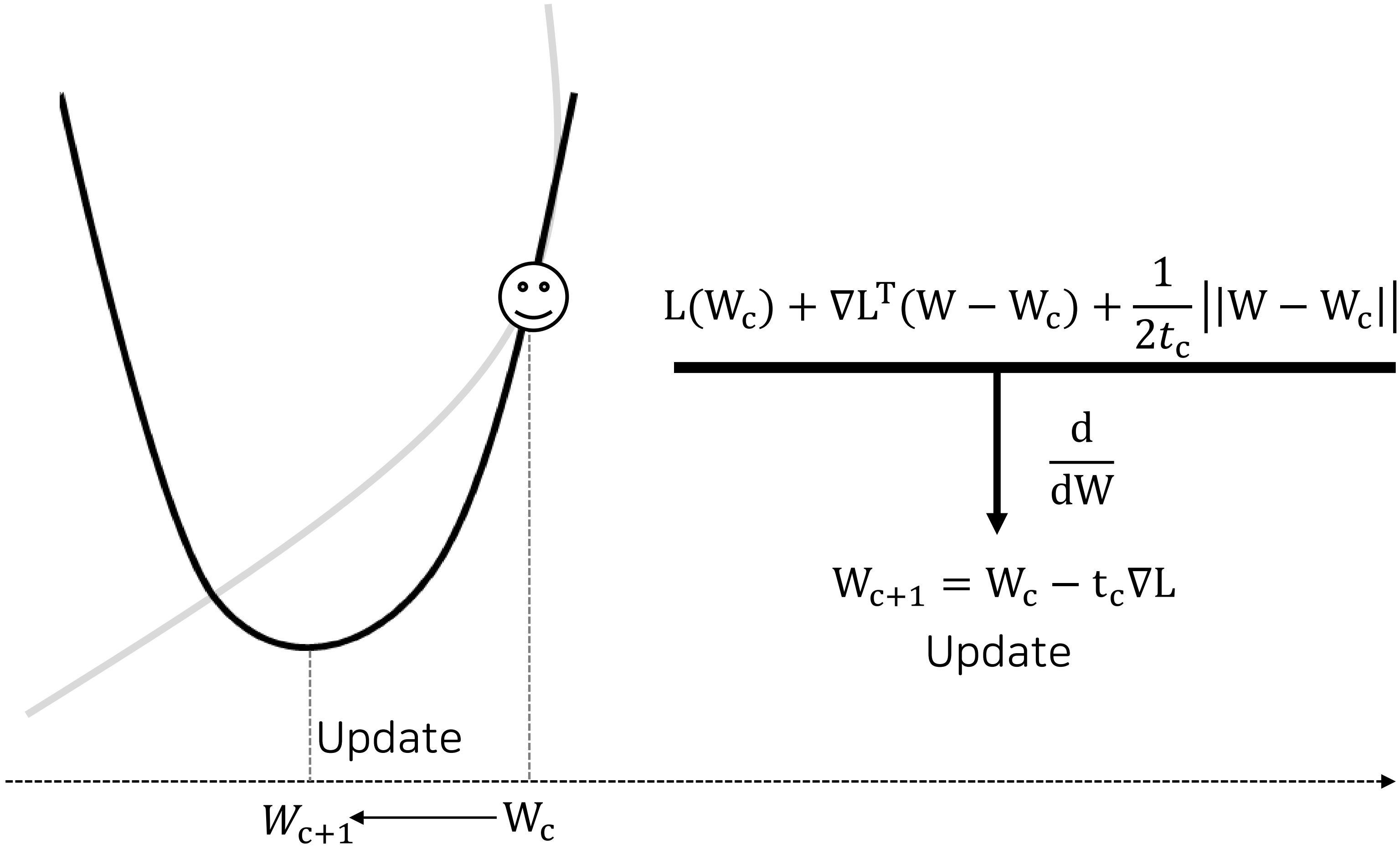


Gradient

$$\mathrm{L}(\mathrm{W_c}) + \nabla \mathrm{L}^{\mathrm{T}}(\mathrm{W} - \mathrm{W_c}) + \frac{1}{2t_c}||\mathrm{W} - \mathrm{W_c}||^2$$

Quadratic approximation

Gradient



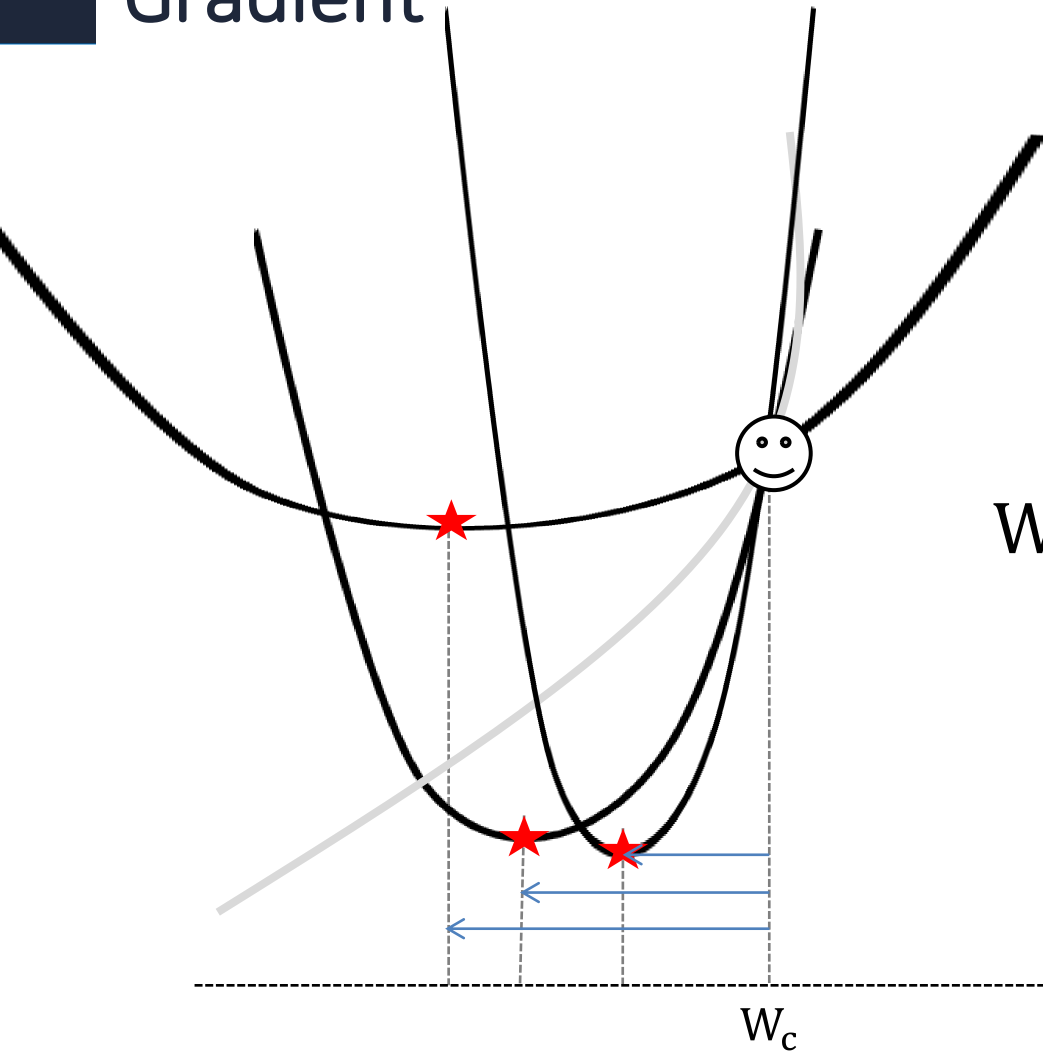
$$L(W_c) + \nabla L^T (W - W_c) + \frac{1}{2t_c} ||W - W_c||^2$$

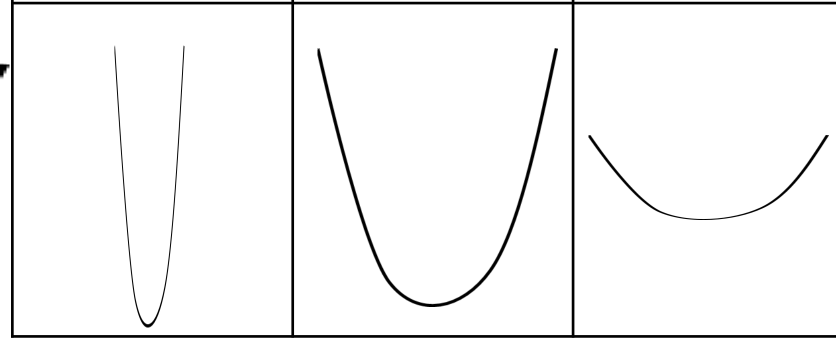
$$\frac{d}{dW}$$

$$W_{c+1} = W_c - t_c \nabla L$$

Update

Gradient



$\frac{1}{2t}\|\|W - W_0\|\|^2 + \nabla L^T(W - W_0) + L(W_0)$		
$t = 0.1$	$t = 1$	$t = 10$
		

$$W_{c+1} = W_c - t_c \nabla L$$

Learning rate
Step size

Gradient

Gradient Descent

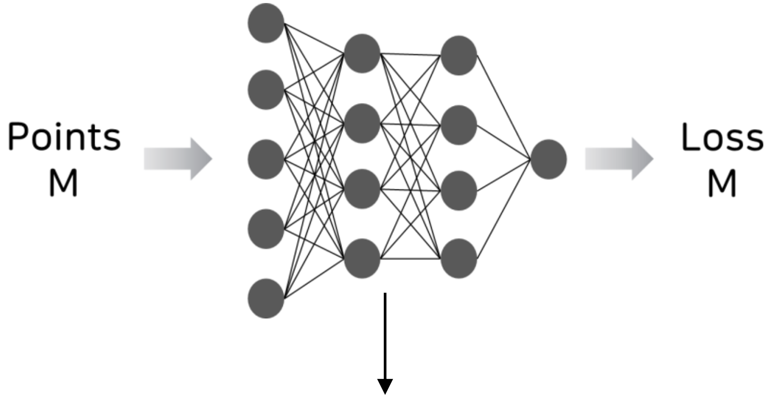
$$W_{c+1} = W_c - t_c \nabla L$$

Towards a lower Loss than the present

Move as much as Stepsize

Stochastic

❖ Consider minimizing an average of functions



$$\min_{W} \frac{1}{m} \sum_{i=1}^{m} L^i(W)$$

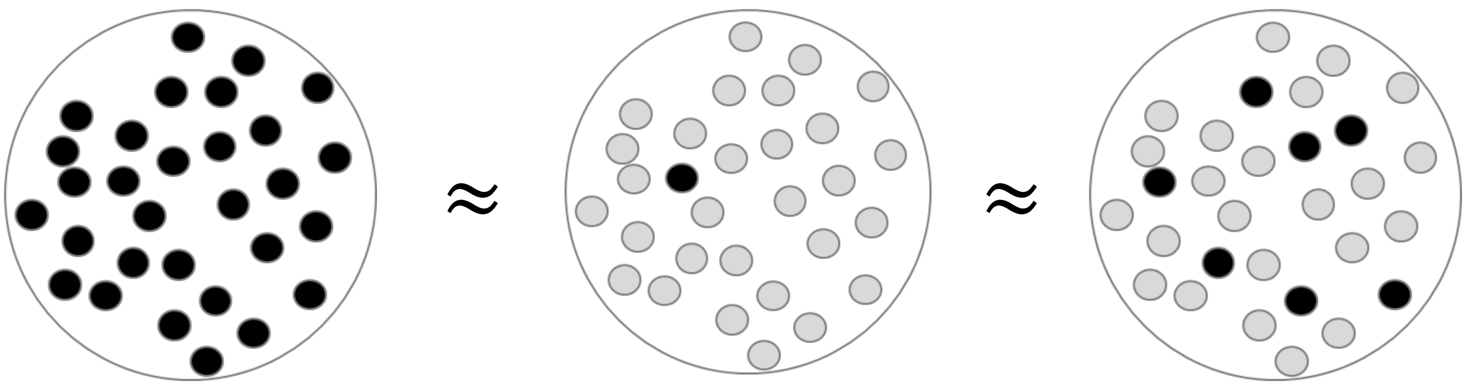
$$W_{c+1} = W_c - t_c \frac{1}{m} \sum_{i=1}^{m} \nabla L^i(W_c)$$

Stochastic

$$W_{c+1} = W_c - t_c \frac{1}{m}\sum_{i=1}^{m} \nabla L^i(W_c)$$

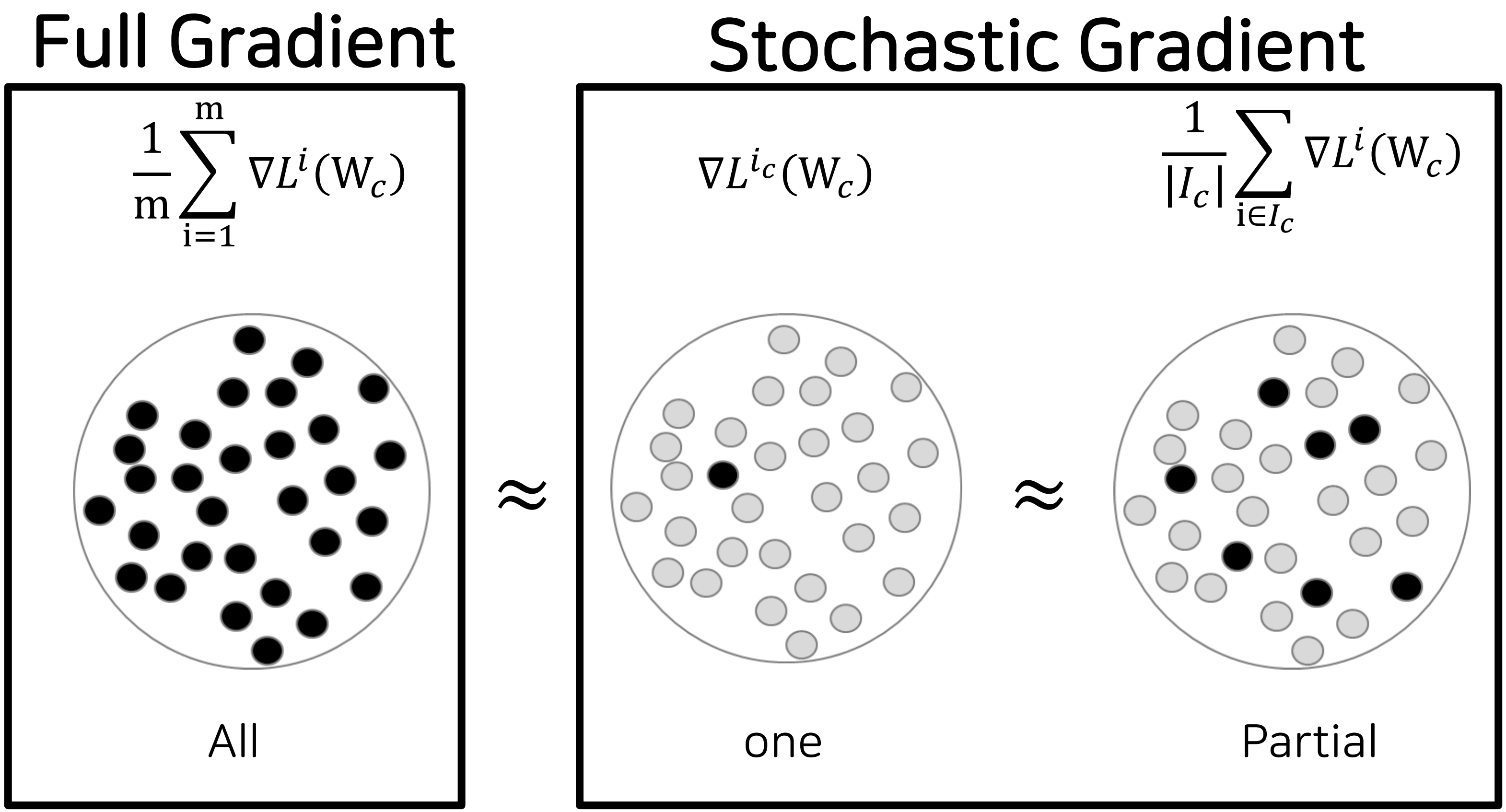
Gradient

$$\frac{1}{m}\sum_{i=1}^{m} \nabla L^i(W_c) \qquad \nabla L^{i_c}(W_c) \qquad \frac{1}{|I_c|}\sum_{i \in I_c} \nabla L^i(W_c)$$



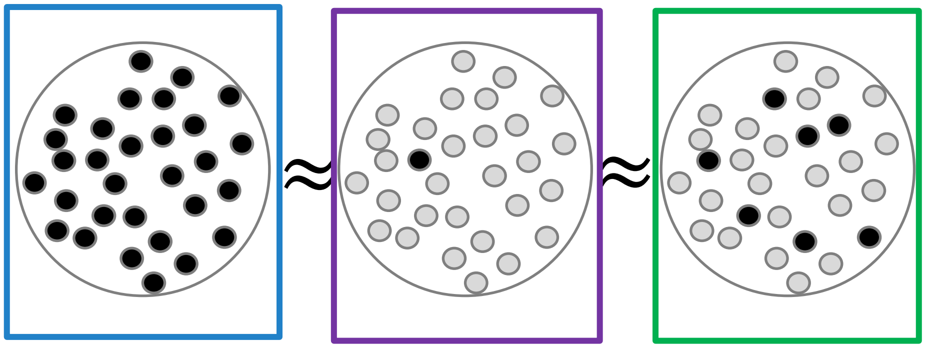
All ≈ one ≈ Partial

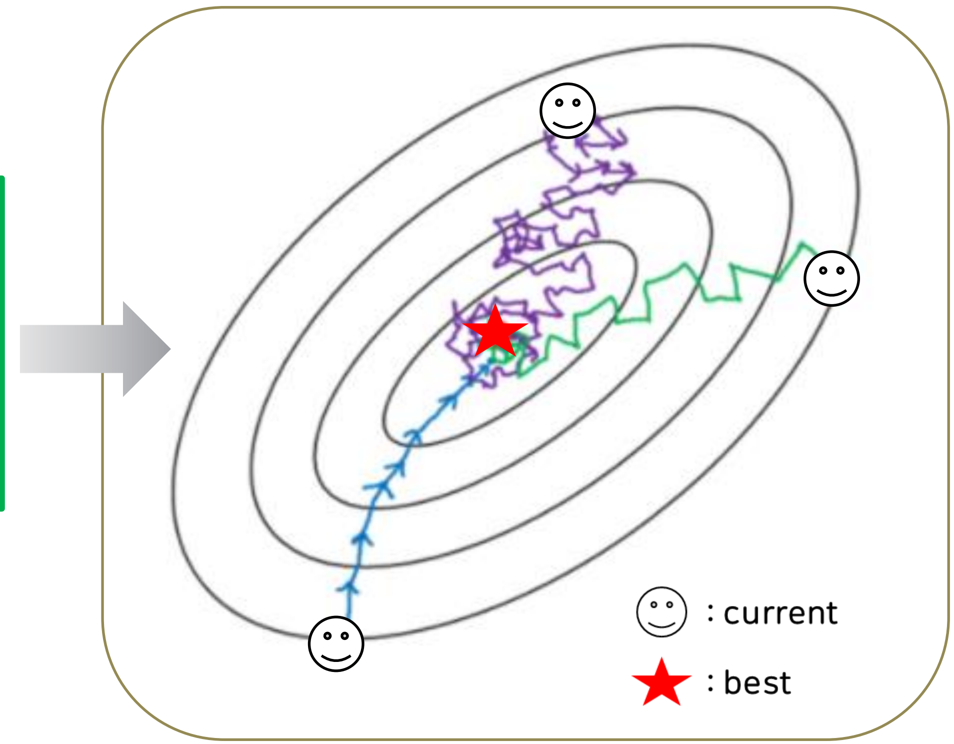
Stochastic


Full Gradient
Stochastic Gradient
$\frac{1}{m}\sum_{i=1}^{m} \nabla L^{i}(W_c)$
$\nabla L^{i_c}(W_c)$
$\frac{1}{|I_c|}\sum_{i \in I_c} \nabla L^{i}(W_c)$
≈
≈
All
one
Partial

Stochastic

$$\frac{1}{m}\sum_{i=1}^{m}\nabla L^{i}(W_c) \qquad \nabla L^{i_c}(W_c) \qquad \frac{1}{|I_c|}\sum_{i\in I_c}\nabla L^{i}(W_c)$$





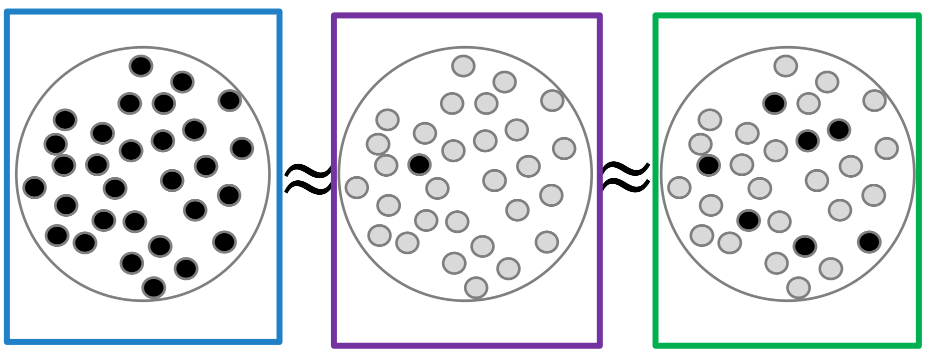
Source : https://suniljangirblog.wordpress.com/2018/12/13/variants-of-gradient-descent/

Stochastic

$$\frac{1}{m}\sum_{i=1}^{m}\nabla L^{i}(W_c)$$

$$\nabla L^{i_c}(W_c)$$

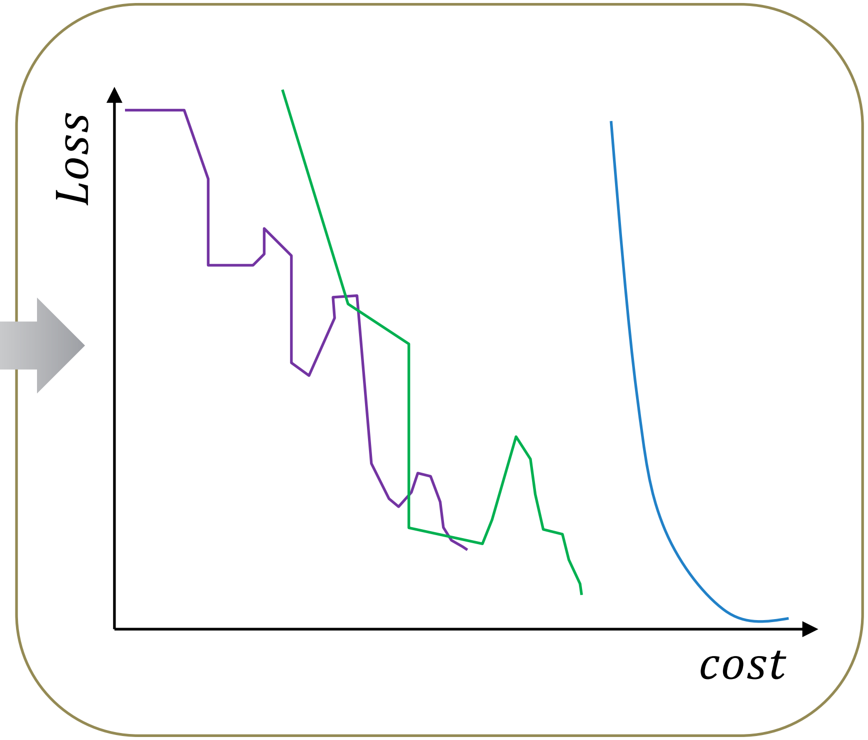
$$\frac{1}{|I_c|}\sum_{i\in I_c}\nabla L^{i}(W_c)$$



Batch gradient descent

Stochastic gradient descent

Mini-batch gradient descent

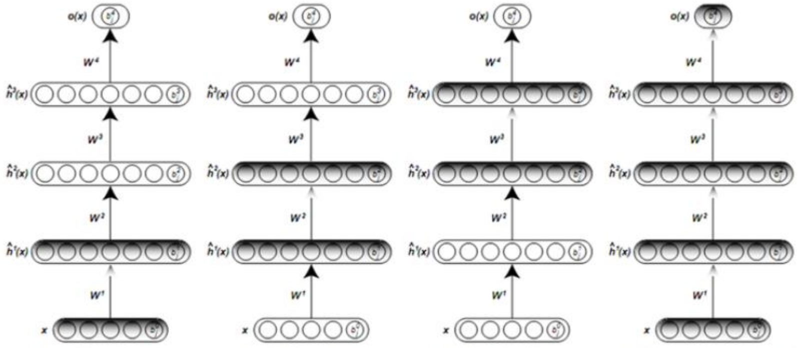


Momentum

Mini-batch gradient descent

$$W_{c+1} = W_c - t_c \frac{1}{|I_c|} \sum_{i \in I_c} \nabla L^i(W_c)$$

Greedy Layer-Wise Training



(a) First hidden layer pre-training (b) Second hidden layer pre-training (c) Third hidden layer pre-training (d) Fine-tuning of whole network

Second-order Optimization

$$W_{c+1} = W_c - H_L \nabla L$$

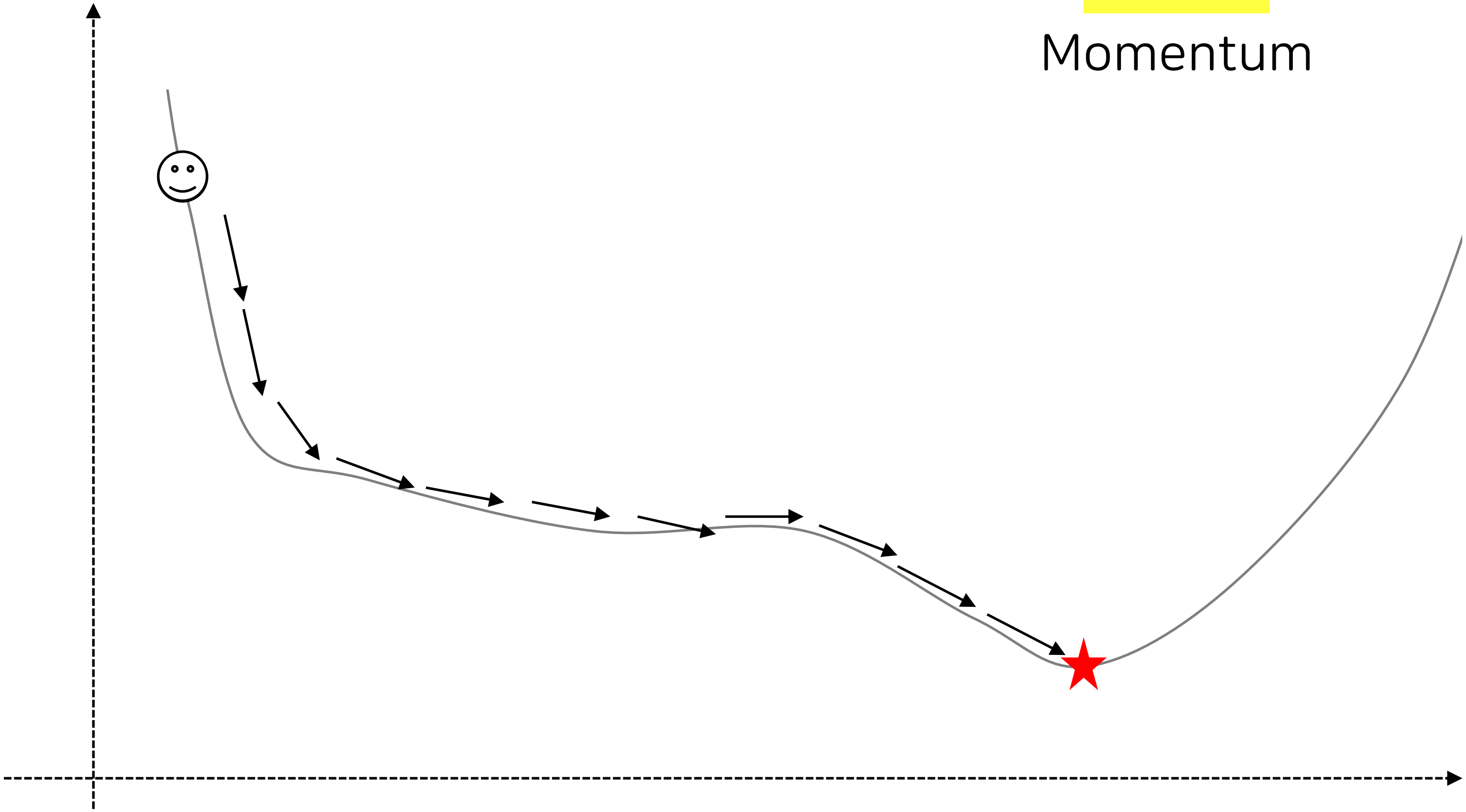
Hessian

Momentum

$$W_{c+1} = W_c + v_c$$

$$v_c = \mu v_{c-1} - t_c \nabla L$$

Momentum

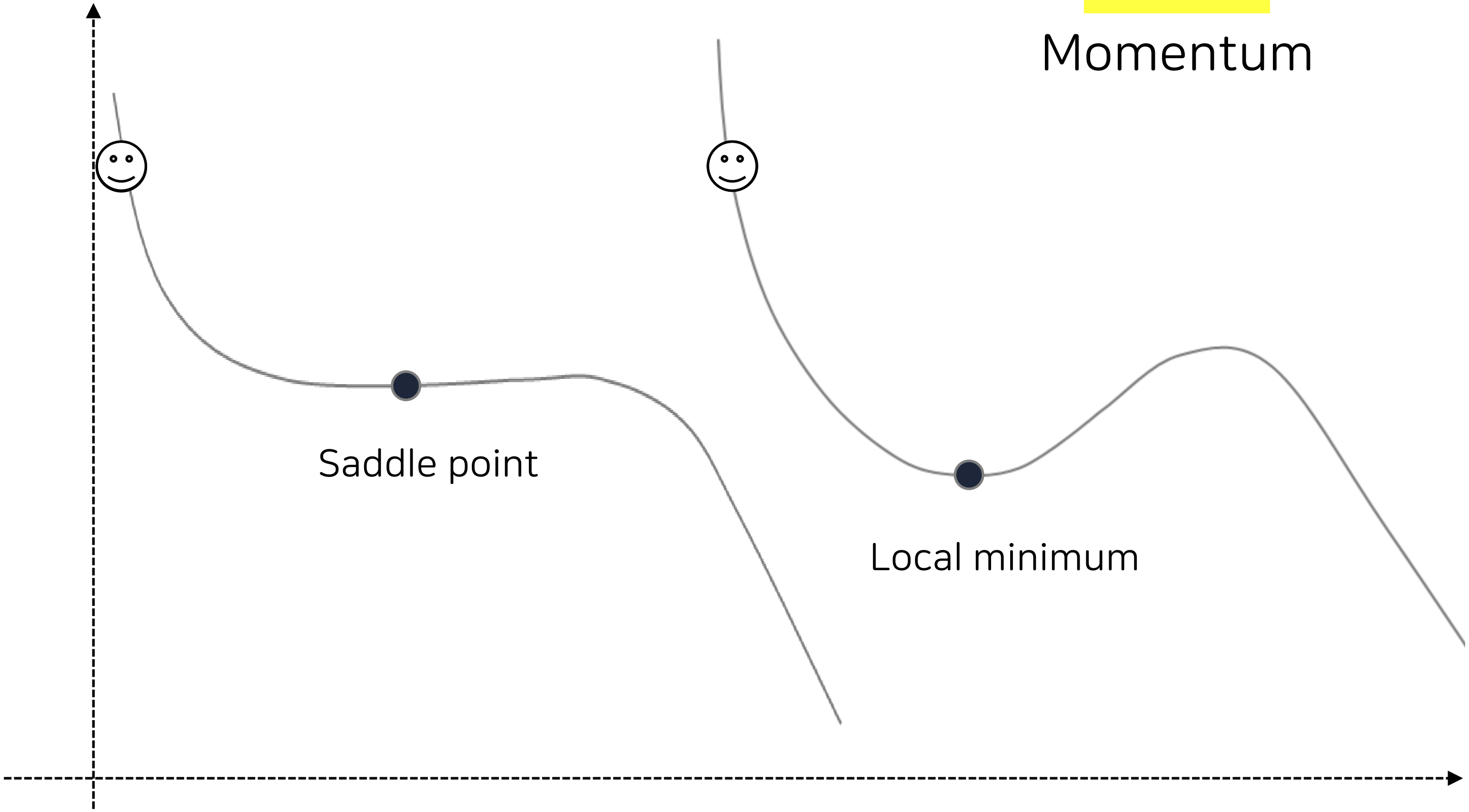


Momentum

$$W_{c+1} = W_c + v_c$$

$$v_c = \mu v_{c-1} - t_c \nabla L$$

Momentum



Adaptive

n: # parameters

$$v_c = \mu v_{c-1} - t_c \nabla L(W_c)$$

$$\begin{bmatrix} v_{1,c} \\ v_{2,c} \\ v_{3,c} \\ \vdots \\ v_{n,c} \end{bmatrix} = \mu \begin{bmatrix} v_{1,c-1} \\ v_{2,c-1} \\ v_{3,c-1} \\ \vdots \\ v_{n,c-1} \end{bmatrix} - t_c \begin{bmatrix} \nabla_1 L(W_{1,c}) \\ \nabla_2 L(W_{2,c}) \\ \nabla_3 L(W_{3,c}) \\ \vdots \\ \nabla_n L(W_{n,c}) \end{bmatrix}$$

n by 1 n by 1 n by 1

Adaptive

n: # parameters

$$v_c = \mu v_{c-1} - t_c \nabla L(W_c)$$

Global stepsize

$$\begin{bmatrix} v_{1,c} \\ v_{2,c} \\ v_{3,c} \\ \vdots \\ v_{n,c} \end{bmatrix} = \mu \begin{bmatrix} v_{1,c-1} \\ v_{2,c-1} \\ v_{3,c-1} \\ \vdots \\ v_{n,c-1} \end{bmatrix} - \begin{bmatrix} t_c \nabla_1 L(W_{1,c}) \\ t_c \nabla_2 L(W_{2,c}) \\ t_c \nabla_3 L(W_{3,c}) \\ \vdots \\ t_c \nabla_n L(W_{n,c}) \end{bmatrix}$$

n by 1 n by 1 n by 1

Adaptive

$\boldsymbol{n}$: # parameters

$$v_c = \mu v_{c-1} - \mathrm{t_c}\nabla \mathrm{L}(\mathrm{W_c})$$

To have its own dynamic stepsize

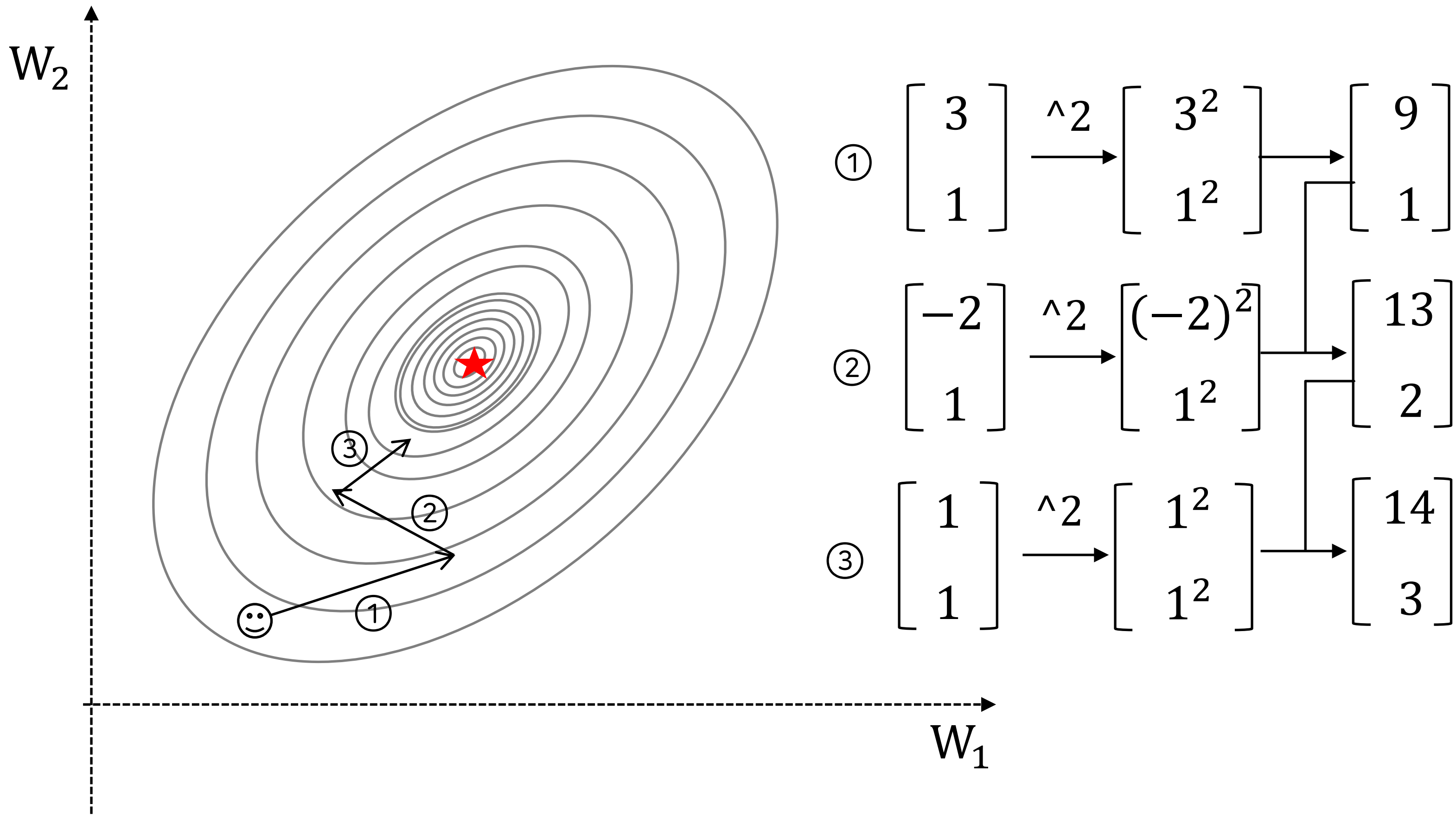
$$\begin{bmatrix} v_{1,c} \\ v_{2,c} \\ v_{3,c} \\ \vdots \\ v_{\mathrm{n},c} \end{bmatrix} = \mu \begin{bmatrix} v_{1,c-1} \\ v_{2,c-1} \\ v_{3,c-1} \\ \vdots \\ v_{\mathrm{n},c-1} \end{bmatrix} - \begin{bmatrix} \widetilde{\mathrm{t_{1,c}}}\nabla_1 L(\mathrm{W_{1,c}}) \\ \widetilde{\mathrm{t_{2,c}}}\nabla_2 L(\mathrm{W_{2,c}}) \\ \widetilde{\mathrm{t_{3,c}}}\nabla_3 L(\mathrm{W_{3,c}}) \\ \vdots \\ \widetilde{\mathrm{t_{n,c}}}\nabla_n L(\mathrm{W}_{n,\mathrm{c}}) \end{bmatrix}$$

n by 1 $\quad$ n by 1 $\quad$ n by 1

Adaptive

: current

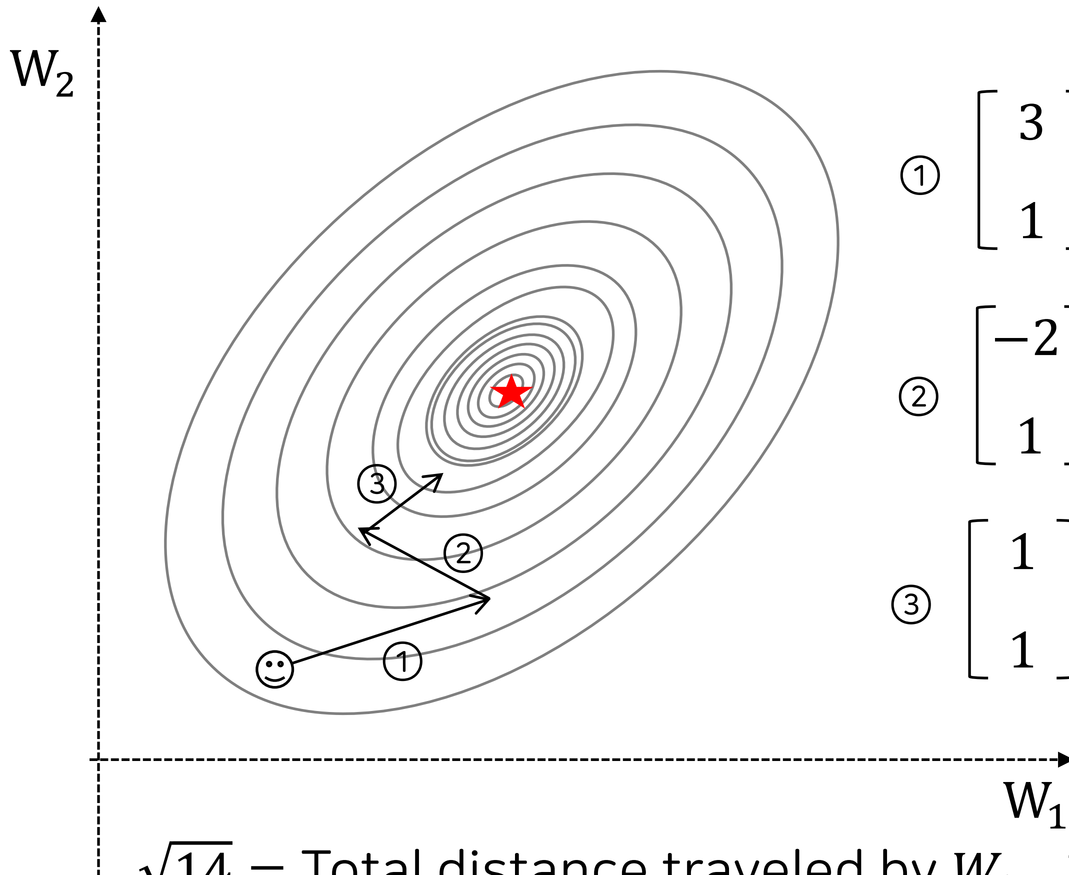
: best



Adaptive

: current

: best



① $\begin{bmatrix} 3 \\ 1 \end{bmatrix} \xrightarrow{\wedge 2} \begin{bmatrix} 3^2 \\ 1^2 \end{bmatrix} \rightarrow \begin{bmatrix} 9 \\ 1 \end{bmatrix}$

② $\begin{bmatrix} -2 \\ 1 \end{bmatrix} \xrightarrow{\wedge 2} \begin{bmatrix} (-2)^2 \\ 1^2 \end{bmatrix} \rightarrow \begin{bmatrix} 13 \\ 2 \end{bmatrix}$

③ $\begin{bmatrix} 1 \\ 1 \end{bmatrix} \xrightarrow{\wedge 2} \begin{bmatrix} 1^2 \\ 1^2 \end{bmatrix} \rightarrow \begin{bmatrix} 14 \\ 3 \end{bmatrix}$

$\sqrt{14}$ = Total distance traveled by W_1

$\sqrt{3}$ = Total distance traveled by W_2

→ $\boldsymbol{W_1}$ **stepsize ↓**

$\boldsymbol{W_2}$ **stepsize ↑**

Adaptive

$$\uparrow \sum_{q=1}^{c} \{\nabla_k L(W_{k,q})\}^2 \propto \frac{1}{\widetilde{t_{k,c}}} \downarrow$$

- Let large gradients have small step size
- Let small gradients have large step size

Adaptive stepsize

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

❖ ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

- 2015 International Conference on Learning Representations(ICLR)
- OpenAI, University of Toronto
- September 22, 2020 : 53545 citation

Published as a conference paper at ICLR 2015

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Diederik P. Kingma*
University of Amsterdam, OpenAI
dpkingma@openai.com

Jimmy Lei Ba*
University of Toronto
jimmy@psi.utoronto.ca

ABSTRACT

We introduce *Adam*, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-parameters have intuitive interpretations and typically require little tuning. Some connections to related algorithms, on which *Adam* was inspired, are discussed. We also analyze the theoretical convergence properties of the algorithm and provide a regret bound on the convergence rate that is comparable to the best known results under the online convex optimization framework. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. Finally, we discuss *AdaMax*, a variant of *Adam* based on the infinity norm.

Algorithm 1: *Adam*, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. g_t^2 indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With β_1^t and β_2^t we denote β_1 and β_2 to the power t.

Require: α: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters θ
Require: θ_0: Initial parameter vector
  $m_0 \leftarrow 0$ (Initialize 1st moment vector)
  $v_0 \leftarrow 0$ (Initialize 2nd moment vector)
  $t \leftarrow 0$ (Initialize timestep)
  while θ_t not converged **do**
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep t)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\widehat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\widehat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \widehat{m}_t / (\sqrt{\widehat{v}_t} + \epsilon)$ (Update parameters)
  end while
  return θ_t (Resulting parameters)

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Require : α: Stepsize

Require : $\beta_1, \beta_2 \in [0,1)$

Require : $f(\theta)$: Loss function

Require : θ_0 Initial parameter vector

$m_0, v_0, t_0 \leftarrow [0,0,0]$

while θ_t not converge do

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$

$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

$\widehat{m}_t \leftarrow \dfrac{m_t}{1 - \beta_1^t}$

$\widehat{v}_t \leftarrow \dfrac{v_t}{1 - \beta_2^t}$

$\theta_t \leftarrow \theta_{t-1} - \widehat{m}_t \alpha / (\sqrt{\widehat{v}_t} + \epsilon)$

end *while*

return θ_t

- High-dimensional parameter
- Little memory requirement
- Naturally step size annealing
- Invariant to re-scaling of gradient

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Require : α: Stepsize

Require : $\beta_1, \beta_2 \in [0,1)$

Require : $f(\theta)$: Loss function

Require : θ_0 Initial parameter vector

$m_0, v_0, t_0 \leftarrow [0,0,0]$

while θ_t not converge do

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$

$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

$\widehat{m_t} \leftarrow \dfrac{m_t}{1 - \beta_1^t}$

$\widehat{v_t} \leftarrow \dfrac{v_t}{1 - \beta_2^t}$

$\theta_t \leftarrow \theta_{t-1} - \widehat{m_t}\alpha / (\sqrt{\widehat{v_t}} + \epsilon)$

end *while*

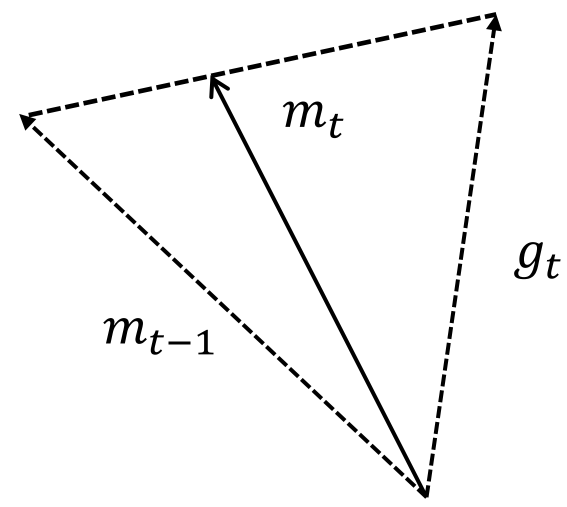
return θ_t

gradient

$$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$$

Gradient with momentum

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$$



ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Require : α: Stepsize

Require : $\beta_1, \beta_2 \in [0,1)$

Require : $f(\theta)$: Loss function

Require : θ_0 Initial parameter vector

$m_0, v_0, t_0 \leftarrow [0,0,0]$

while θ_t not converge do

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$

$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

$\widehat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$

$\widehat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$

$\theta_t \leftarrow \theta_{t-1} - \widehat{m}_t \alpha / (\sqrt{\widehat{v}_t} + \epsilon)$

end *while*

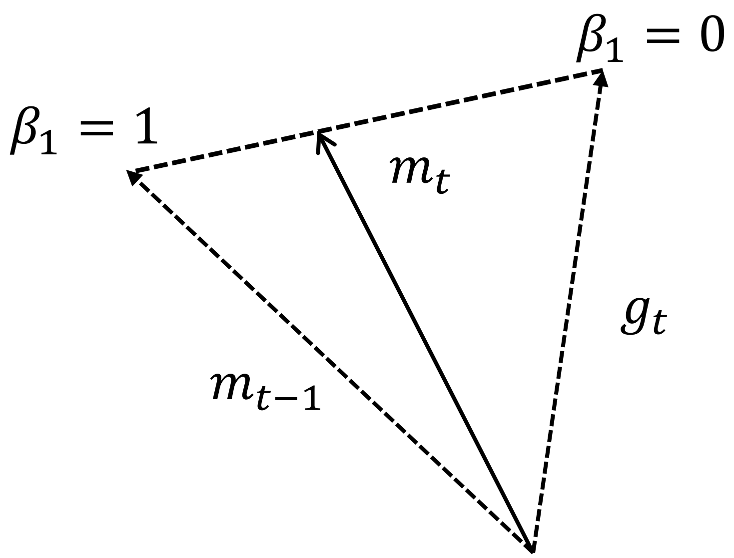
return θ_t

gradient

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

Gradient with momentum

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$



ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Require : α: Stepsize

Require : $\beta_1, \beta_2 \in [0,1)$

Require : $f(\theta)$: Loss function

Require : θ_0 Initial parameter vector

$m_0, v_0, t_0 \leftarrow [0,0,0]$

while θ_t not converge do

$$t \leftarrow t+1$$

$$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$$

$$m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$$

$$v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$

$$\widehat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$$

$$\widehat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$$

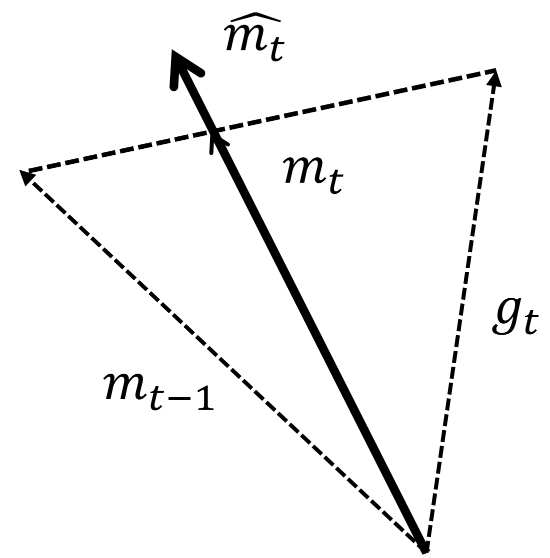
$$\theta_t \leftarrow \theta_{t-1} - \widehat{m_t}\alpha/(\sqrt{\widehat{v}_t} + \epsilon)$$

end *while*

return θ_t

$$m_0 \leftarrow 0$$

$$\widehat{m_t} \leftarrow \frac{m_t}{1-\beta_1^t}, \beta_1 \in [0,1)$$



ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Require : α: Stepsize

Require : $\beta_1, \beta_2 \in [0,1)$

Require : $f(\theta)$: Loss function

Require : θ_0 Initial parameter vector

$m_0, v_0, t_0 \leftarrow [0,0,0]$

while θ_t not converge do

$$t \leftarrow t + 1$$

$$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$$

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\widehat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$$

$$\widehat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$$

$$\theta_t \leftarrow \theta_{t-1} - \widehat{m}_t \alpha / (\sqrt{\widehat{v}_t} + \epsilon)$$

end *while*

return θ_t

- Accumulated sum(g_t^2) continues to grow.
- stepsize will become infinitesimally small.
- Decaying the squared gradients

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

$$\frac{1}{stepsize} \propto \sum_{q=1}^{c} \{\nabla_k L(W_{k,q})\}^2$$

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Require : α: Stepsize

Require : $\beta_1, \beta_2 \in [0,1)$

Require : $f(\theta)$: Loss function

Require : θ_0 Initial parameter vector

$m_0, v_0, t_0 \leftarrow [0,0,0]$

while θ_t not converge do

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$

$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

$\widehat{m_t} \leftarrow \frac{m_t}{1 - \beta_1^t}$

$\widehat{v_t} \leftarrow \frac{v_t}{1 - \beta_2^t}$

$\theta_t \leftarrow \theta_{t-1} - \widehat{m_t}\alpha / (\sqrt{\widehat{v_t}} + \epsilon)$

end *while*

return θ_t

$$\widehat{v_t} \leftarrow \frac{v_t}{1 - \beta_2^t}, \beta_2 \in [0,1)$$

$$\theta_t \leftarrow \theta_{t-1} - \widehat{m_t}\alpha / (\sqrt{\widehat{v_t}} + \epsilon)$$

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Require : α: Stepsize

Require : $\beta_1, \beta_2 \in [0,1)$

Require : $f(\theta)$: Loss function

Require : θ_0 Initial parameter vector

$m_0, v_0, t_0 \leftarrow [0,0,0]$

while θ_t not converge do

$t \leftarrow t + 1$

$g_t \leftarrow \nabla_\theta f_t(\theta_{t-1})$

$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$

$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$

$\widehat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$

$\widehat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$

$\theta_t \leftarrow \theta_{t-1} - \widehat{m}_t \alpha / (\sqrt{\widehat{v}_t} + \epsilon)$

end *while*

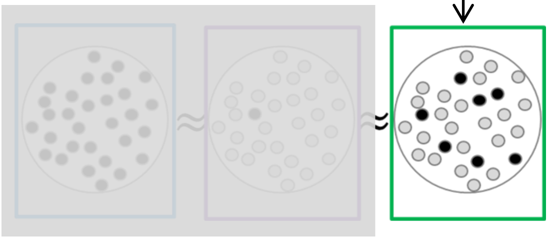
return θ_t

Gradient and Momentum

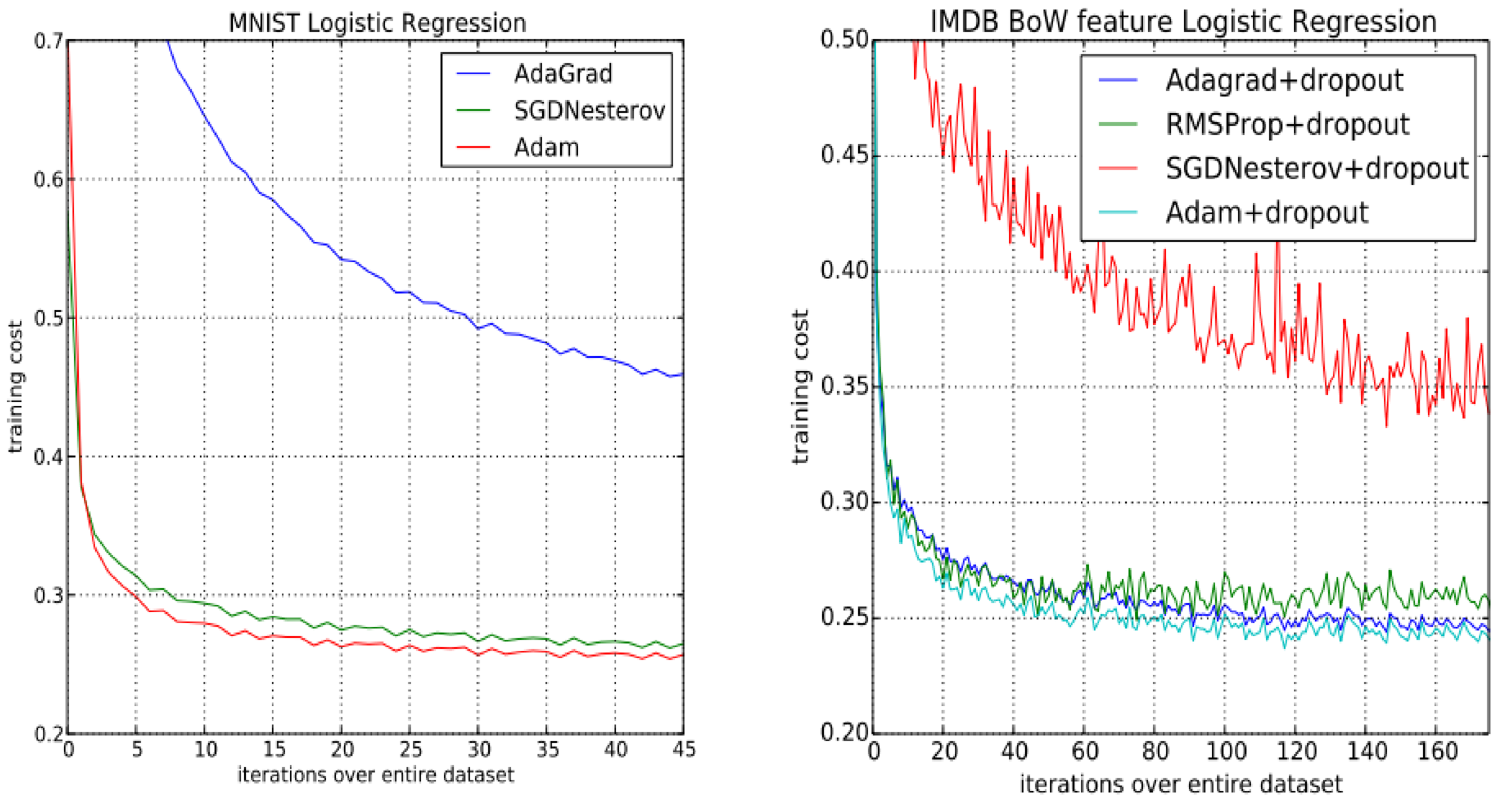
Adaptive

$$\theta_t \leftarrow \theta_{t-1} - \widehat{m}_t \alpha / (\sqrt{\widehat{v}_t} + \epsilon)$$

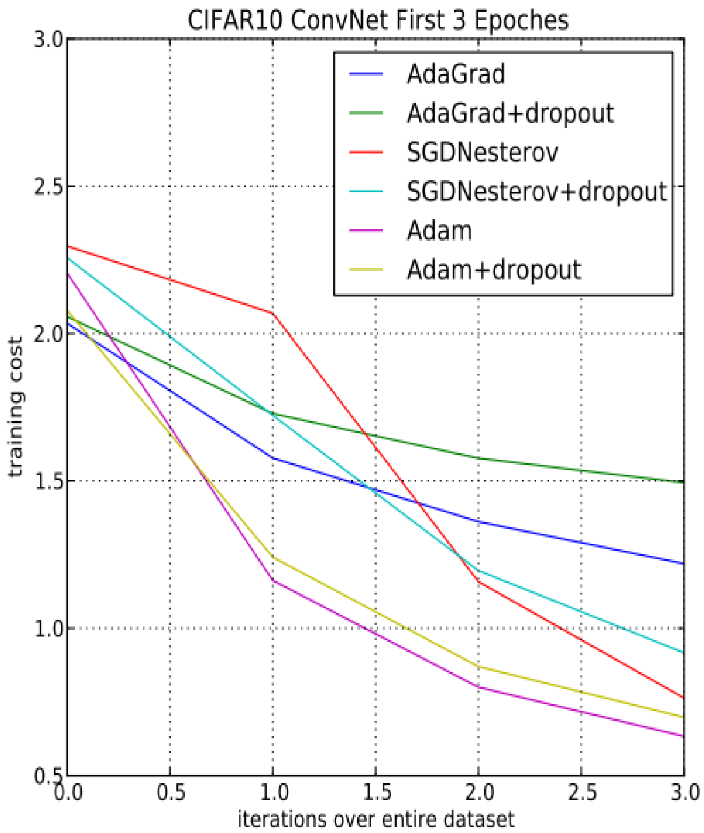
Stochastic

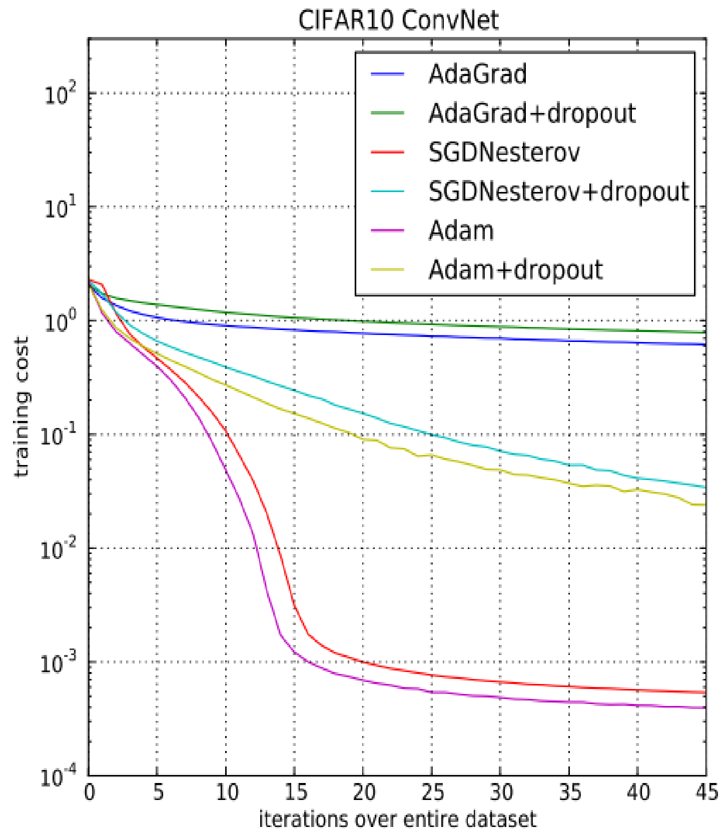


ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION



ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION





Lookahead Optimizer: k steps forward, 1 step back

❖ Lookahead Optimizer: k steps forward, 1 step back

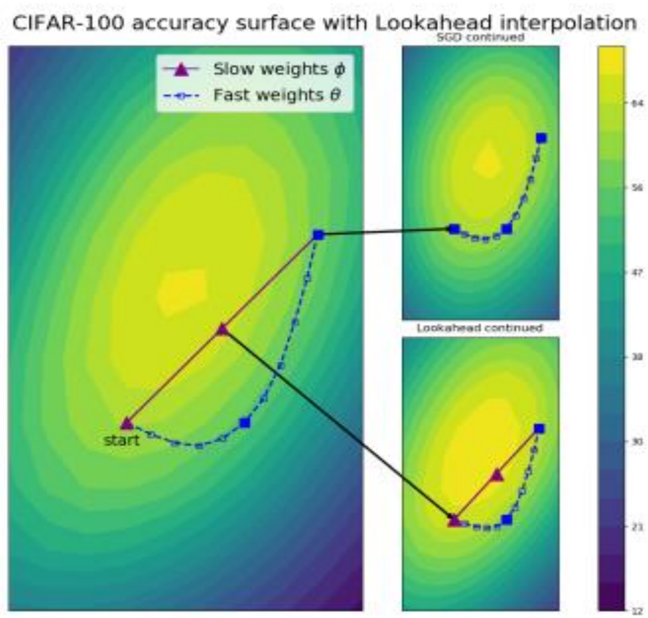
- 2019 Neural Information Processing Systems(NeurIPS)
- Michael R. Zhang, James Lucas, Geoffrey E Hinton, Jimmy Ba
- September 22, 2020 : 83 citation

Lookahead Optimizer: k steps forward, 1 step back

Michael R. Zhang, James Lucas, Geoffrey Hinton, Jimmy Ba
Department of Computer Science, University of Toronto, Vector Institute
{michael, jlucas, hinton,jba}@cs.toronto.edu

Abstract

The vast majority of successful deep neural networks are trained using variants of stochastic gradient descent (SGD) algorithms. Recent attempts to improve SGD can be broadly categorized into two approaches: (1) adaptive learning rate schemes, such as AdaGrad and Adam, and (2) accelerated schemes, such as heavy-ball and Nesterov momentum. In this paper, we propose a new optimization algorithm, Lookahead, that is orthogonal to these previous approaches and iteratively updates two sets of weights. Intuitively, the algorithm chooses a search direction by *looking ahead* at the sequence of "fast weights" generated by another optimizer. We show that Lookahead improves the learning stability and lowers the variance of its inner optimizer with negligible computation and memory cost. We empirically demonstrate Lookahead can significantly improve the performance of SGD and Adam, even with their default hyperparameter settings on ImageNet, CIFAR-10/100, neural machine translation, and Penn Treebank.

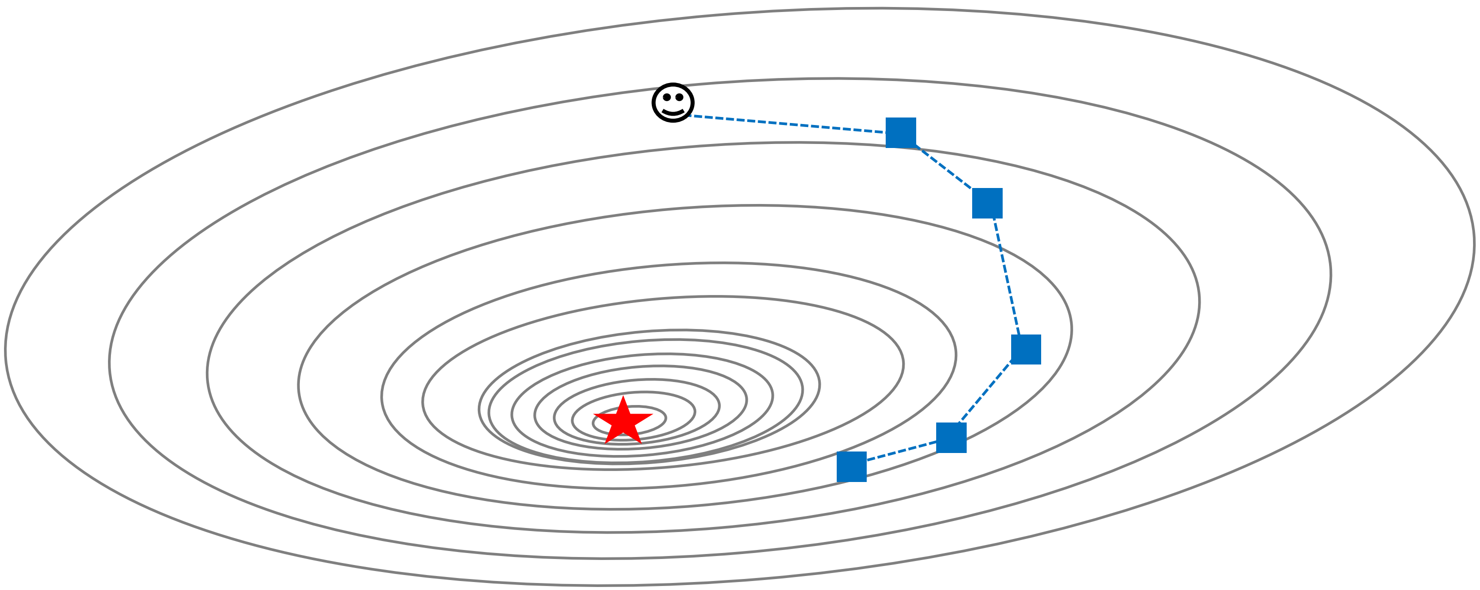


Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

$(\text{ex. } k = 5)$

Adam or SGD

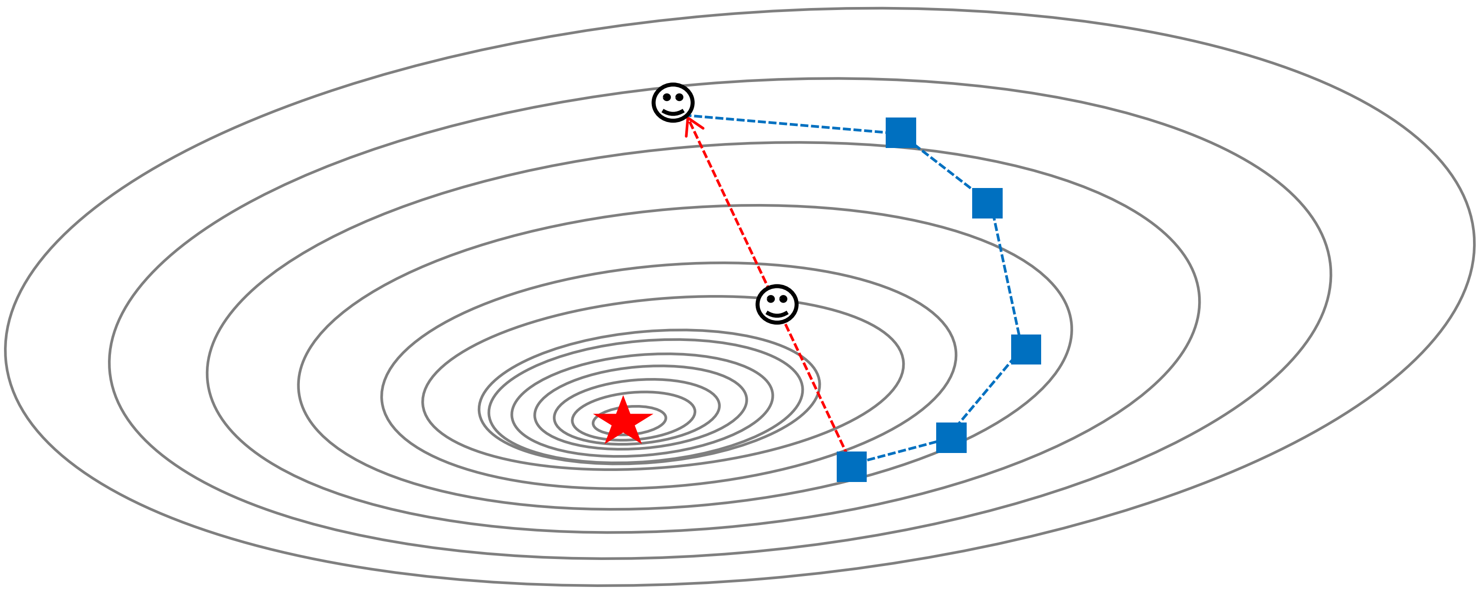


Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

(ex. k = 5)
Adam or SGD

interpolation

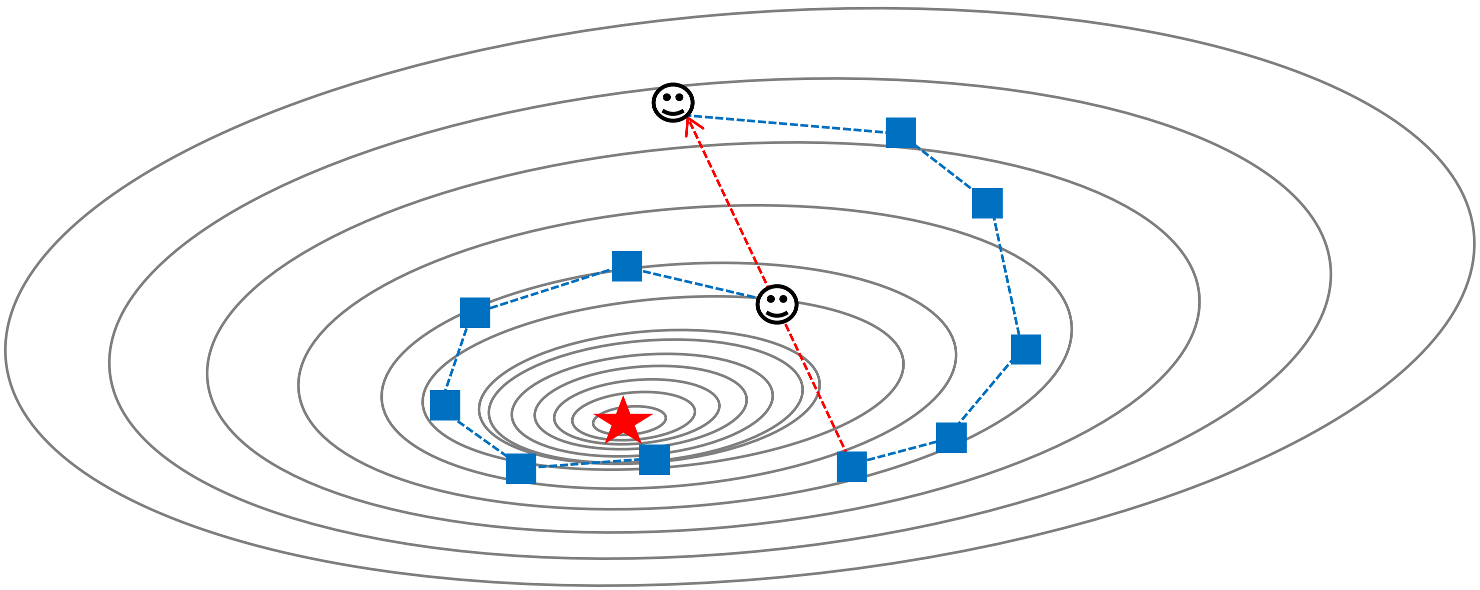


Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

$(\text{ex.}\,\mathrm{k}=5)$ Adam or SGD

interpolation



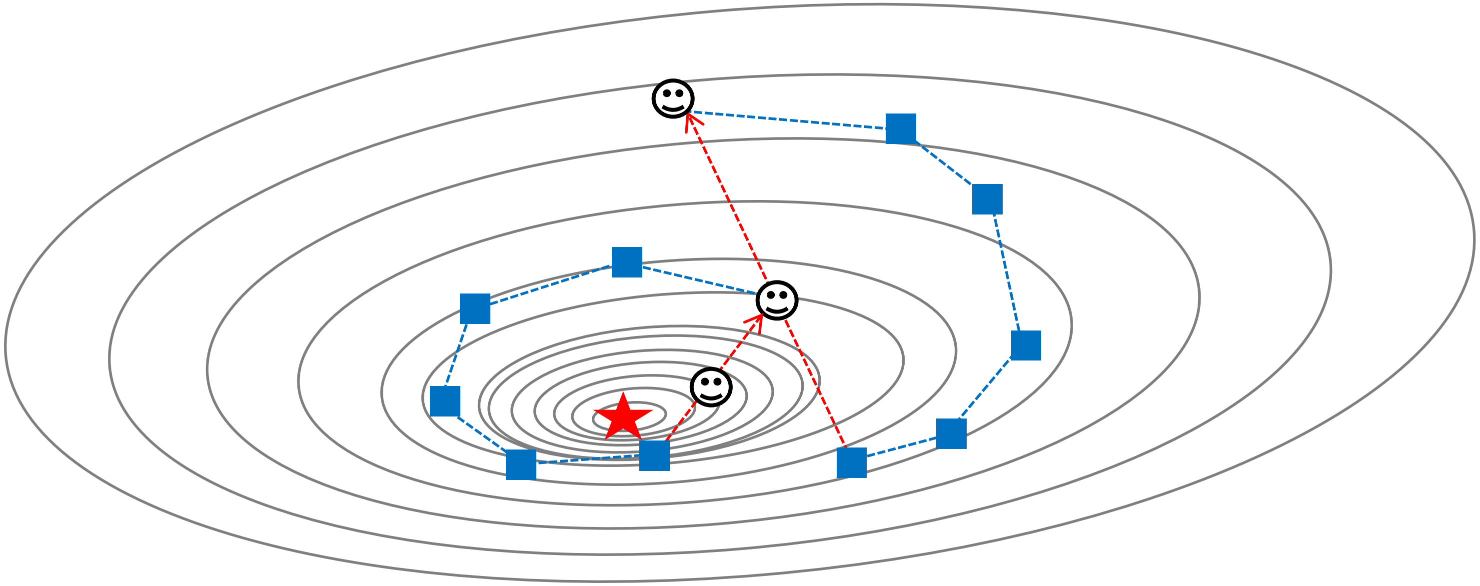
Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

(ex. k = 5)

Adam or SGD

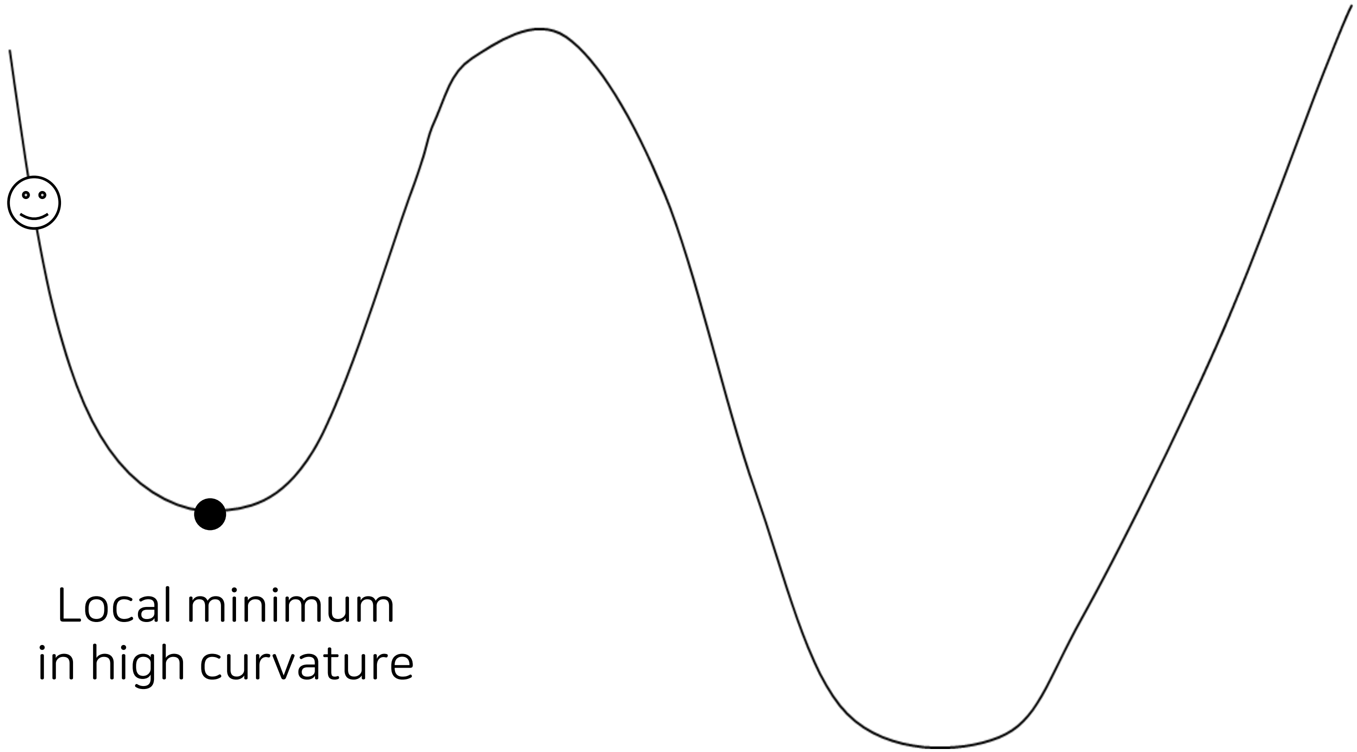
interpolation



Lookahead Optimizer: k steps forward, 1 step back

❖ SGD or Adam

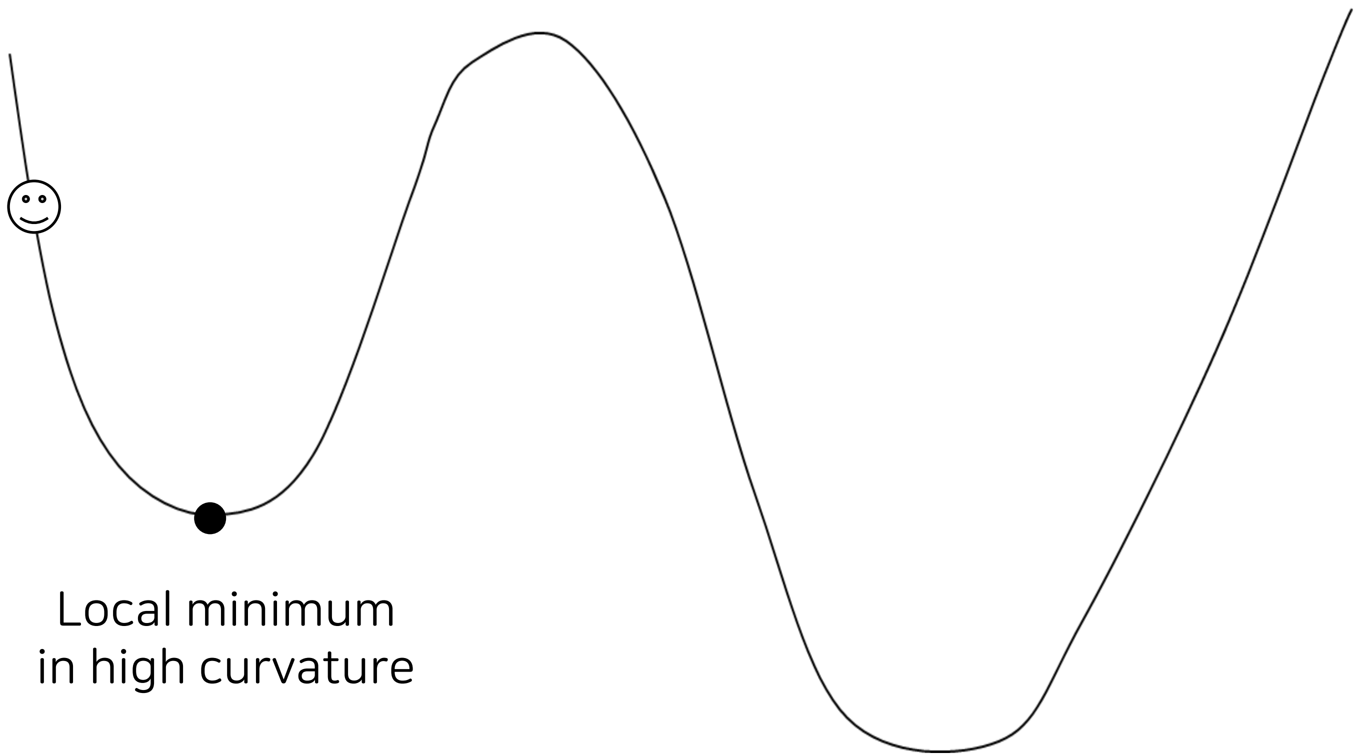
→ It is difficult to escape local minima in high curvature directions.



Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

- SGD or Adam
- Large stepsize

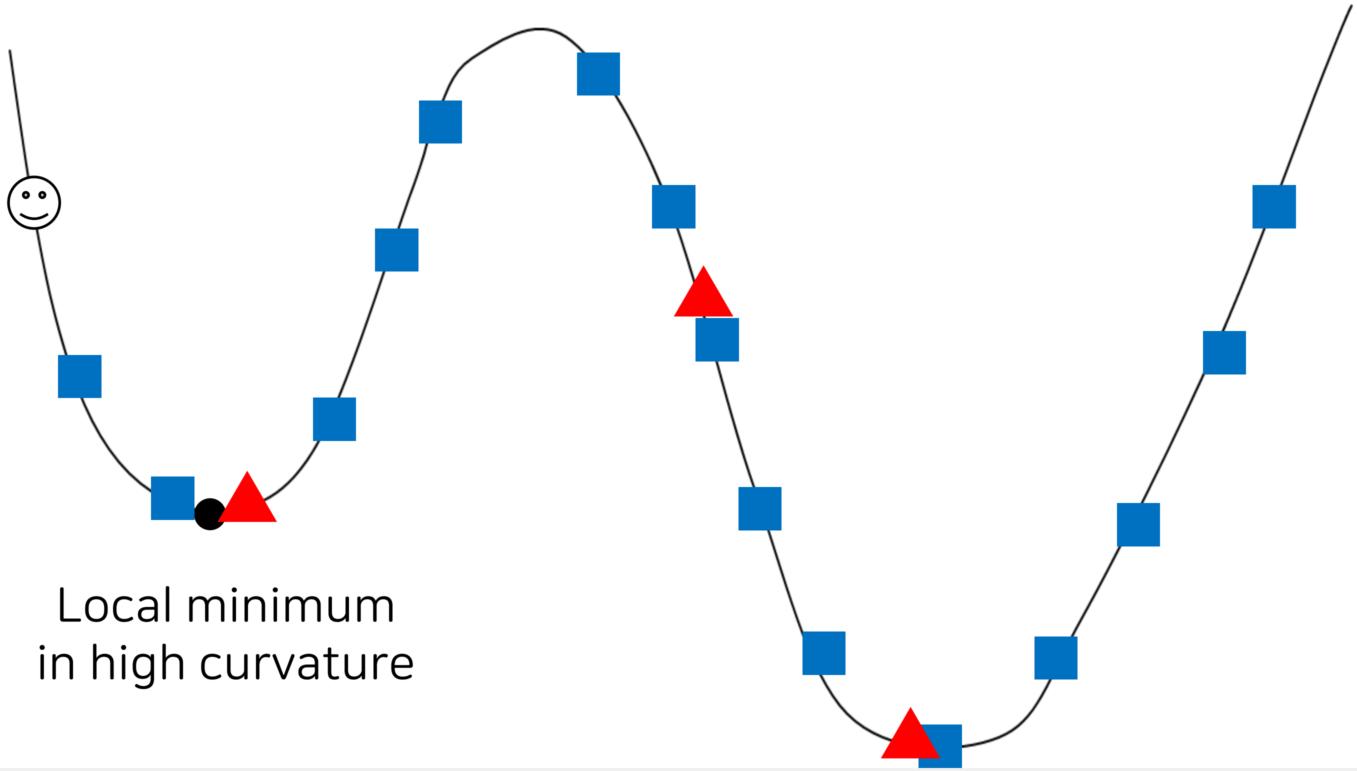


Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

- SGD or Adam
- Large stepsize

- Smooth out the oscillations
- Variance reduction



Local minimum
in high curvature

Lookahead Optimizer: k steps forward, 1 step back

Require : Initial parameter ϕ_0, Loss function L

Require : Synchronization period k,

slow eights step size α, optimizer A

for t = 1,2, ... do

Synchronize parameters $\theta_{t,0} \leftarrow \phi_{t-1}$

for i = 1,2, ... , k do

sample minibatch of data d $\sim\mathcal{D}$

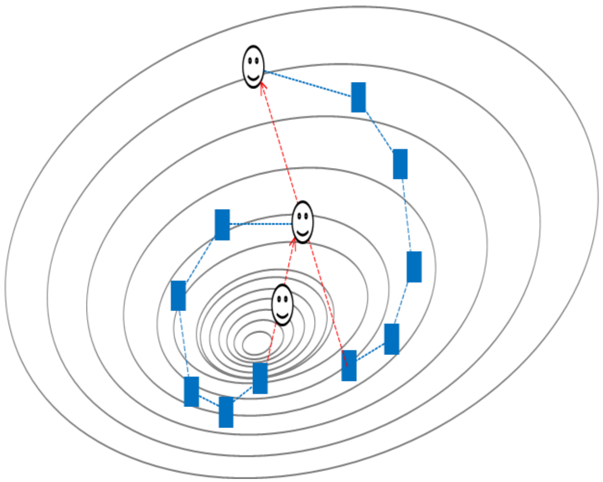
$\theta_{t,i} \leftarrow \theta_{t,i-1} + A(L, \theta_{t,i-1}, d)$

$end\ for$

$Perform\ outer\ update\ \phi_t \leftarrow \phi_{t-1} + \alpha(\theta_{t,k} - \phi_{t-1})$

$end\ for$

return $paramters\phi$



Lookahead Optimizer: k steps forward, 1 step back

Require : Initial parameter ϕ_0, Loss function L

Require : Synchronization period k,

slow eights step size α, optimizer A

for t = 1,2, … do

Slow weight

Synchronize parameters $\theta_{t,0} \leftarrow \phi_{t-1}$

for i = 1,2, … , k do

Fast weight

sample minibatch of data d $\sim\mathcal{D}$

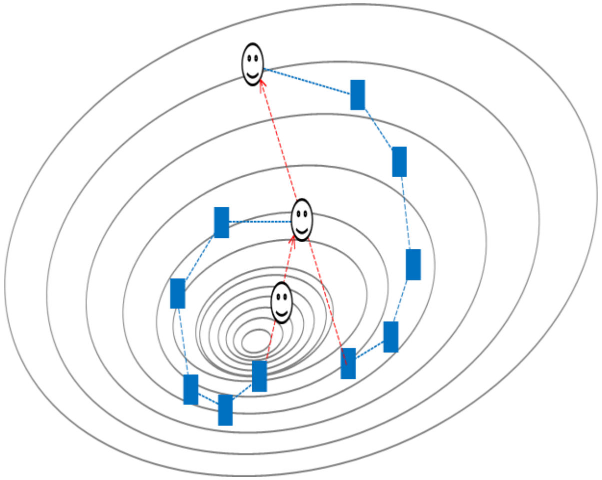
$\theta_{t,i} \leftarrow \theta_{t,i-1} + A(L, \theta_{t,i-1}, d)$

$end\ for$

$Perform\ outer\ update\ \phi_t \leftarrow \phi_{t-1} + \alpha(\theta_{t,k} - \phi_{t-1})$

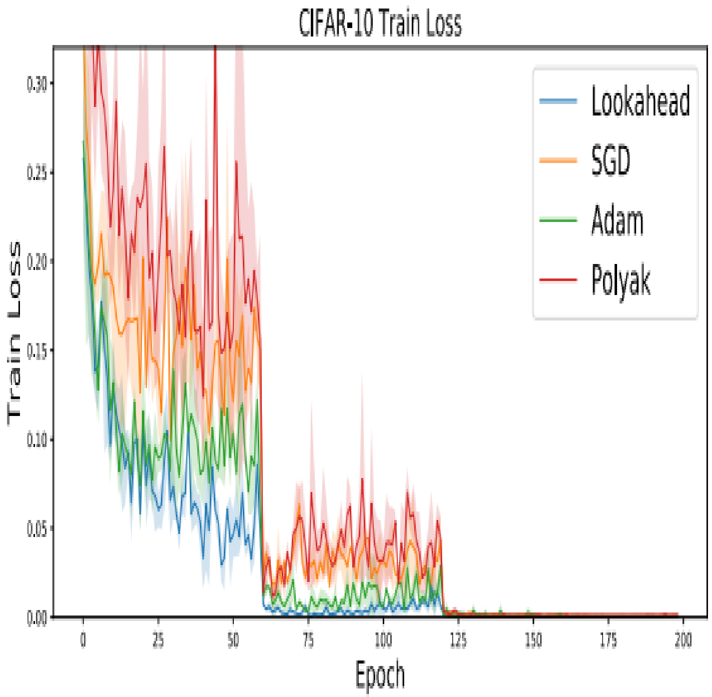
$end\ for$

return $paramters\phi$



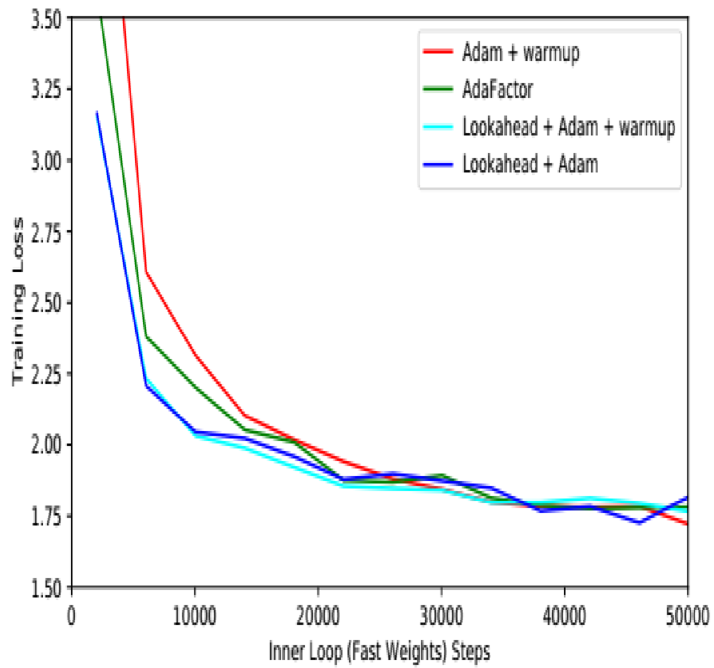
Lookahead Optimizer: k steps forward, 1 step back

ResNet-18
(CIFAR-10)



$\alpha = 0.8, k = 5$

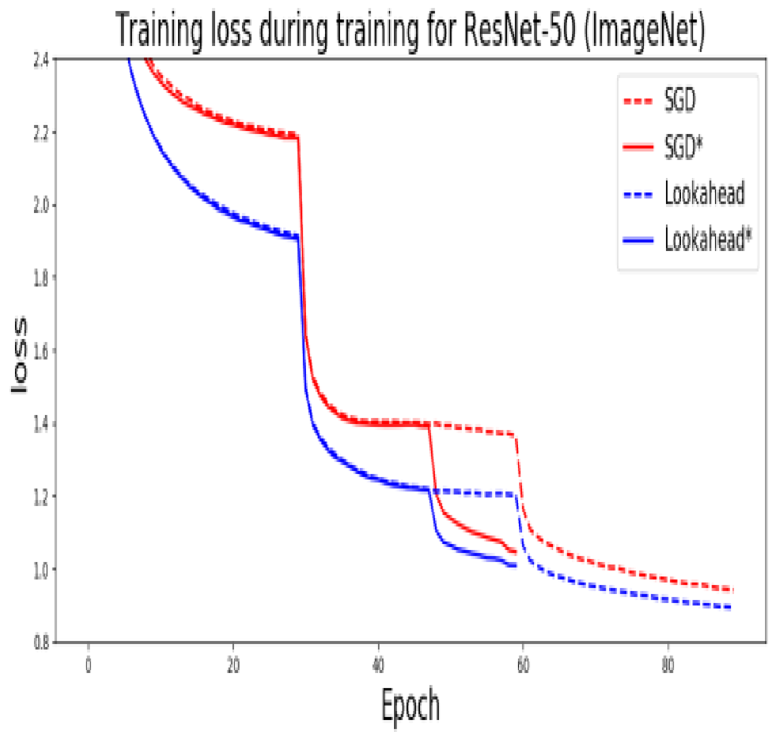
Transformer
(English to German)



$\alpha = 0.5, k = 10$

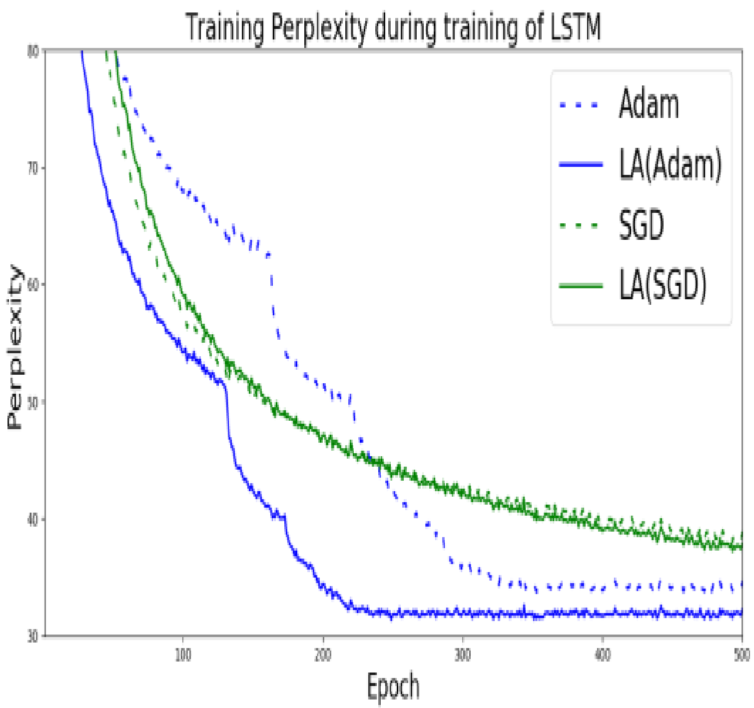
Lookahead Optimizer: k steps forward, 1 step back

ResNet-50
(ImageNet)



$\alpha = 0.5, k = 5$

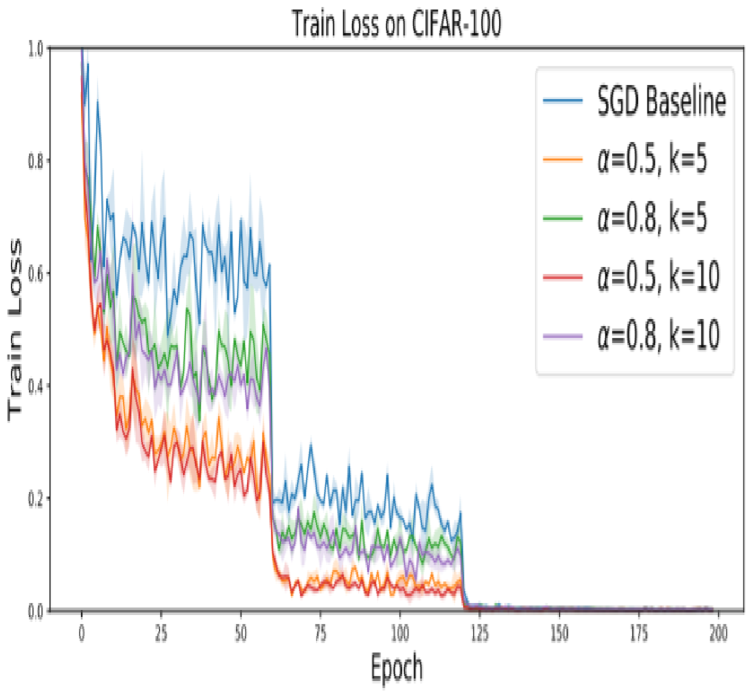
LSTM
(Penn Treebank)



$\alpha = 0.5, k = 5$

Lookahead Optimizer: k steps forward, 1 step back

ResNet-18
(CIFAR-10)

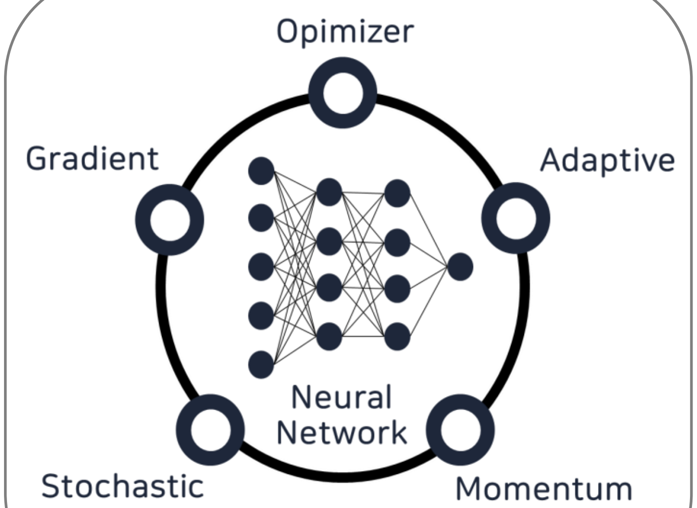


K \ α	0.5	0.8
5	$78.24 \pm .02$	$78.27 \pm .04$
10	$78.19 \pm .22$	$77.94 \pm .22$

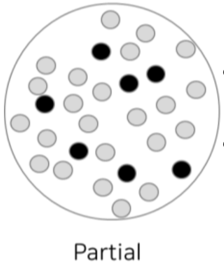
- Test Accuracy
- α, K Robust

Conclusion

~ Current



Current ~



- Noise reduction
- Variance reduction

Second-order Optimization

$$W_{c+1} = W_c - \underline{H_L} \nabla L$$

Hessian

Reference

감사합니다.

Reference

❖ Paper

- Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).
- Zhang, Michael, et al. "Lookahead optimizer: k steps forward, 1 step back." Advances in Neural Information Processing Systems. 2019.
- Bottou, Léon, Frank E. Curtis, and Jorge Nocedal. "Optimization methods for large-scale machine learning." Siam Review 60.2 (2018): 223-311.
- Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." International conference on machine learning. 2013.

❖ Blog

- https://www.deepideas.net/deep-learning-from-scratch-iv-gradient-descent-and-backpropagation/
- https://medium.com/@lessw/new-deep-learning-optimizer-ranger-synergistic-combination-of-radam-lookahead-for-the-best-of-2dc83f79a48d
- https://www.youtube.com/watch?v=Q2dewZweAtU
- https://www.youtube.com/watch?v=mdKjMPmcWjY
- https://algorithmia.com/blog/introduction-to-optimizers