

2020년 9월 25일

Stochastic Gradient Descent

Data Mining & Quality Analytics Lab

발표자 : 배진수

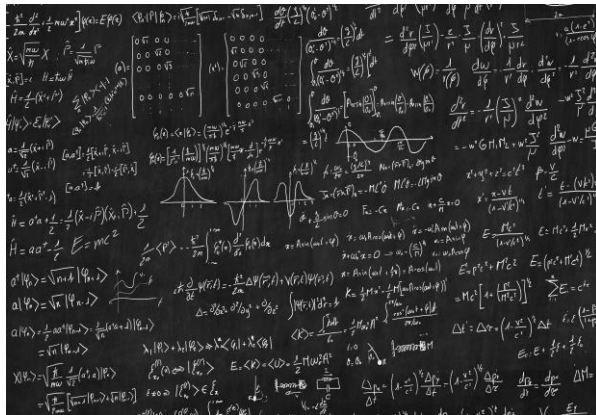
Presenter



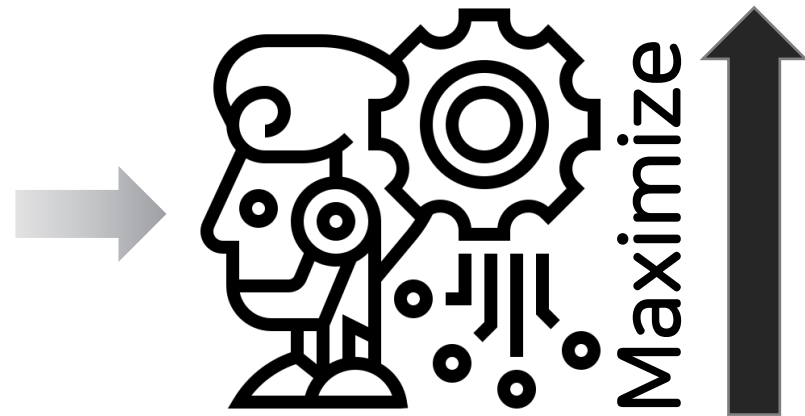
❖ 배진수(Jinsoo Bae)

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

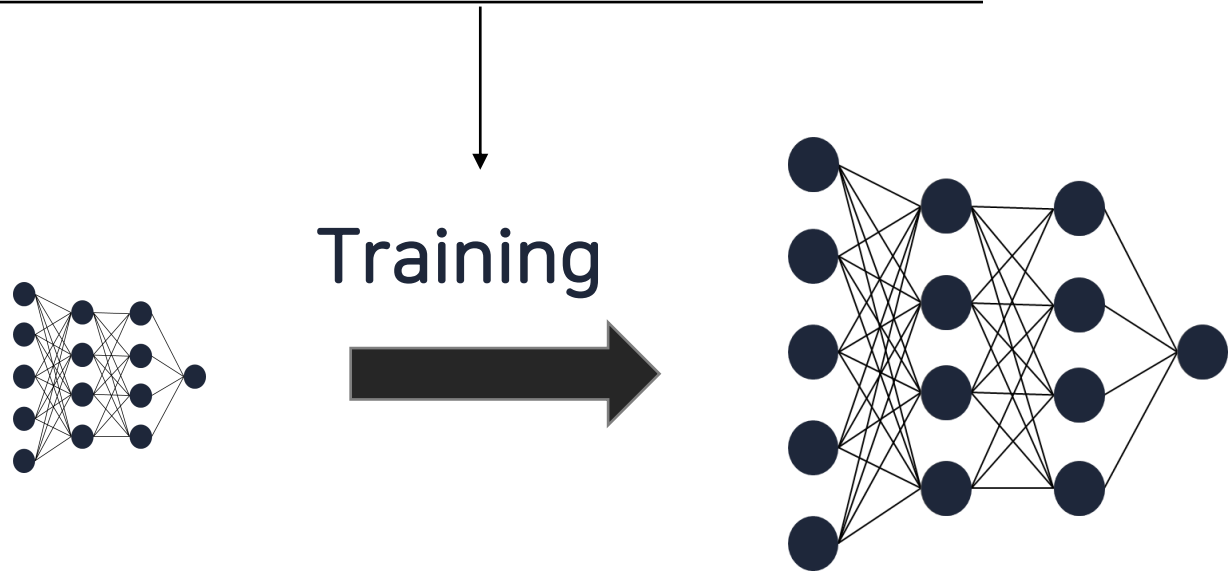
Math, statistical science



Machine learning and Optimization



Stochastic Gradient Descent



Stochastic Gradient Descent

Adam

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

Diederik P. Kingma^{*} Jimmy Lei Ba^{*}
University of Amsterdam, OpenAI University of Toronto
d.p.kingma@uva.nl jlb@cs.toronto.edu

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 hyperparameters 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 Adam's, a variant of Adam based on the infinity norm.

1 INTRODUCTION

Stochastic gradient-based optimization is of core practical importance in many fields of science and engineering. Many problems in these fields can be cast as the optimization of some scalar parameterized objective function requiring maximization or minimization with respect to its parameters. If the function is differentiable w.r.t. its parameters, gradient descent is a relatively efficient optimization method, since the computation of first-order partial derivatives w.r.t. all the parameters is of the same computational complexity as just evaluating the function. Often, objective functions are stochastic. For example, many objective functions are composed of a sum of subfunctions evaluated at different subregions of data, as this case optimization can be made more efficient by taking gradient steps w.r.t. individual subfunctions, i.e. stochastic gradient descent (SGD) or ascent. SGD proved itself as an efficient and effective optimization method that was central to many machine learning success stories, such as recent advances in deep learning (Peng et al., 2015; Krizhevsky et al., 2012; Hinton & Salakhutdinov, 2006; Hinton et al., 2012a; Graves et al., 2013). Objectives may also have other sources of noise than data subsampling, such as dropout (Hinton et al., 2012b) regularization. For all such noisy objectives, efficient stochastic optimization techniques are required. The focus of this paper is on the optimization of stochastic objectives with high-dimensional parameter spaces. In these cases, higher-order optimization methods are ill-suited, and discussion in this paper will be restricted to first-order methods.

We propose Adam, a method for efficient stochastic optimization that only requires first-order gradients with little memory requirement. The method computes individual adaptive learning rates for different parameters from estimates of first and second moments of the gradients, the name Adam is derived from adaptive moment estimation. Our method is designed to combine the advantages of two recently popular methods: Adagrad (Duchi et al., 2011), which works well with sparse gradients, and RMSProp (Tieleman & Hinton, 2012), which works well in on-line and non-stationary settings; important connections to these and other stochastic optimization methods are clarified in section 5. Some of Adam's advantages are that the magnitudes of parameter updates are invariant to rescaling of the gradients, its updates are approximately bounded by the sparse hyperparameter, it does not require a stationary objective, it works with sparse gradients, and it naturally performs a form of step size annealing.

^{*}Equal contribution. Author ordering determined by coin flip over a Google Hangout.

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
{mzhang, jylucas, hinton, jba}@cs.toronto.edu

Abstract

The most 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 directs a search direction by looking ahead at the sequence of "fast weights" generated by another optimizer. We show that Lookahead improves the training 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.

1 Introduction

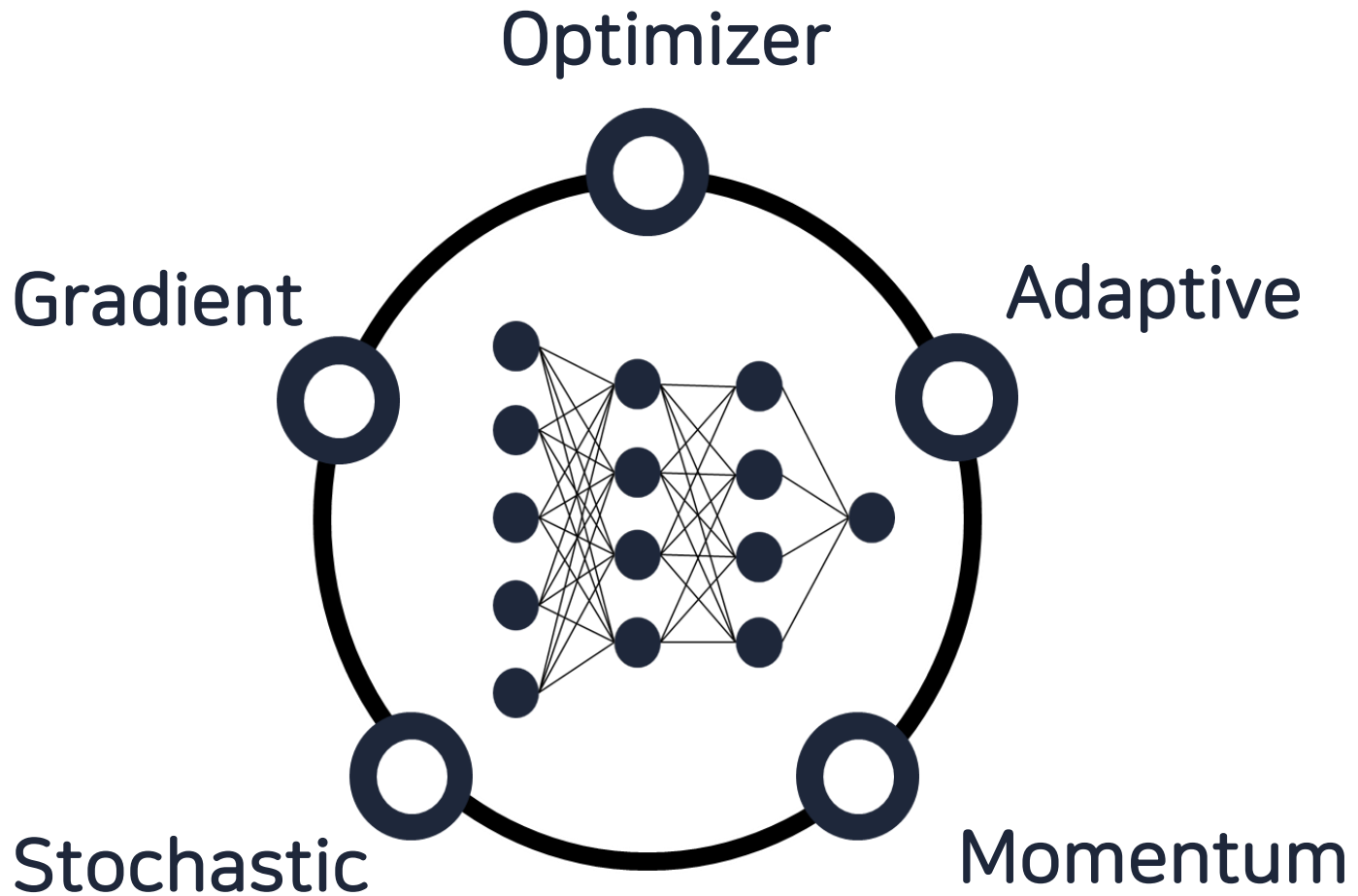
Despite their simplicity, SGD-like algorithms remain competitive for neural network training against advanced second-order optimization methods. Large-scale distributed optimization algorithms [10, 45] have shown impressive performance in combination with improved learning rate scheduling schemes [42, 31], yet variants of SGD remain the core algorithm in the distributed systems. The recent improvements to SGD can be broadly categorized into two approaches: (1) adaptive learning rate schemes, such as Adagrad [7] and Adam [16], and (2) accelerated schemes, such as Polyak heavy ball [33] and Nesterov momentum [29]. Both approaches make use of the accumulated past gradient information to achieve faster convergence. However, to obtain their improved performance in neural networks often require costly hyperparameter tuning [28].

In this work, we present Lookahead, a new optimization method, that is orthogonal to these previous approaches. Lookahead first updates the "fast weights" [12] it uses using any standard optimizer as its inner loop before updating the "slow weights" once in the direction of the final fast weights. We show that this update reduces the variance. We find that Lookahead has low sensitivity to hyperparameter and iteratively reduces the need for extensive hyperparameter tuning. By using hyperparameters and iteration counts the same for equivalent hyperparameter tuning. By using different deep learning tasks with minimal computational overhead.

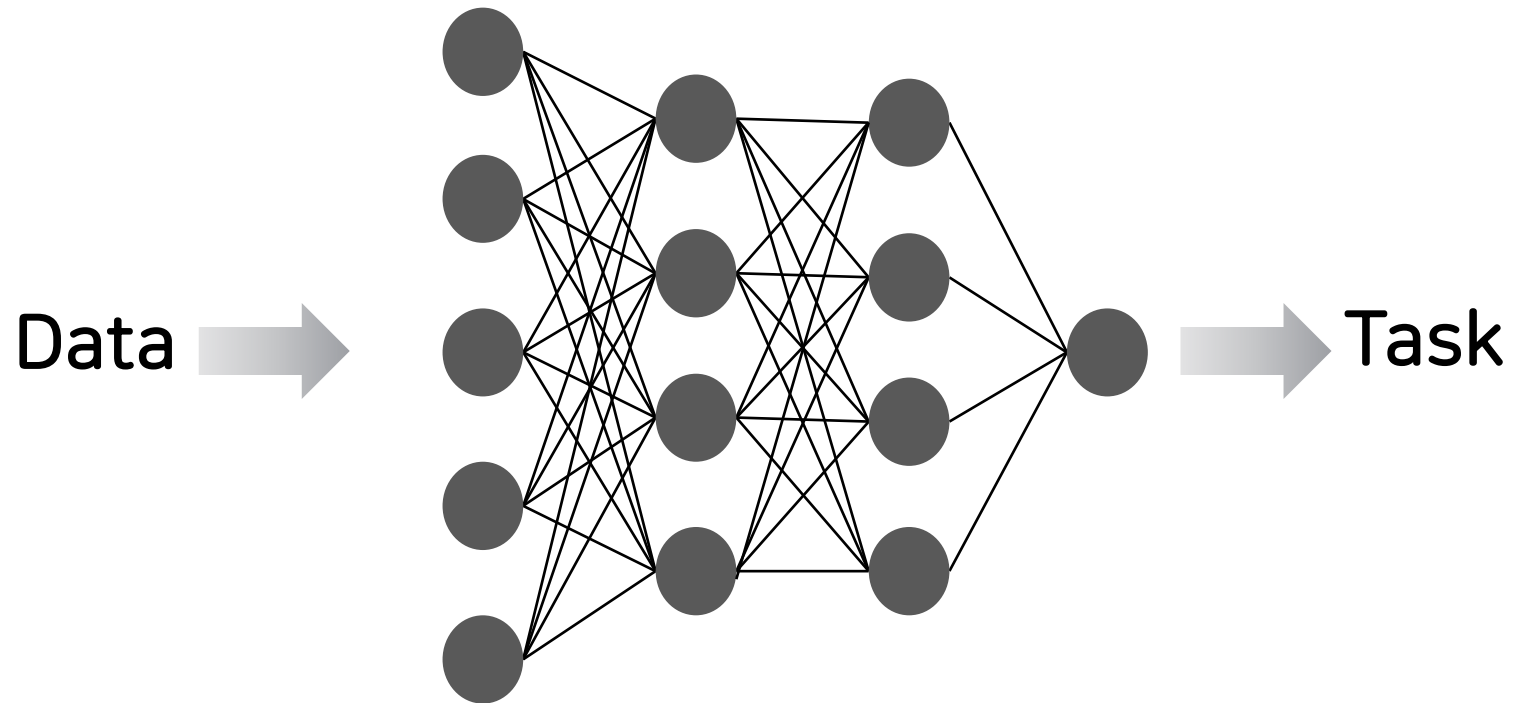
Empirically, we evaluate Lookahead by training classifiers on the CIFAR-100 and ImageNet datasets [5], observing faster convergence on the ResNet-50 and ResNet-152 architectures [11]. We also trained LSTM language models on the Penn Treebank dataset [24] and Transformer-based [12] neural machine translation models on the WMT 2014 English-to-German dataset. For all tasks, using Lookahead leads to improved convergence over the inner optimizer and often improved generalization performance while being robust to hyperparameter changes. Our experiments demonstrate that Lookahead is robust to changes in the inner loop optimizer, the number of fast weight updates, and the slow weight learning rate.

3rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

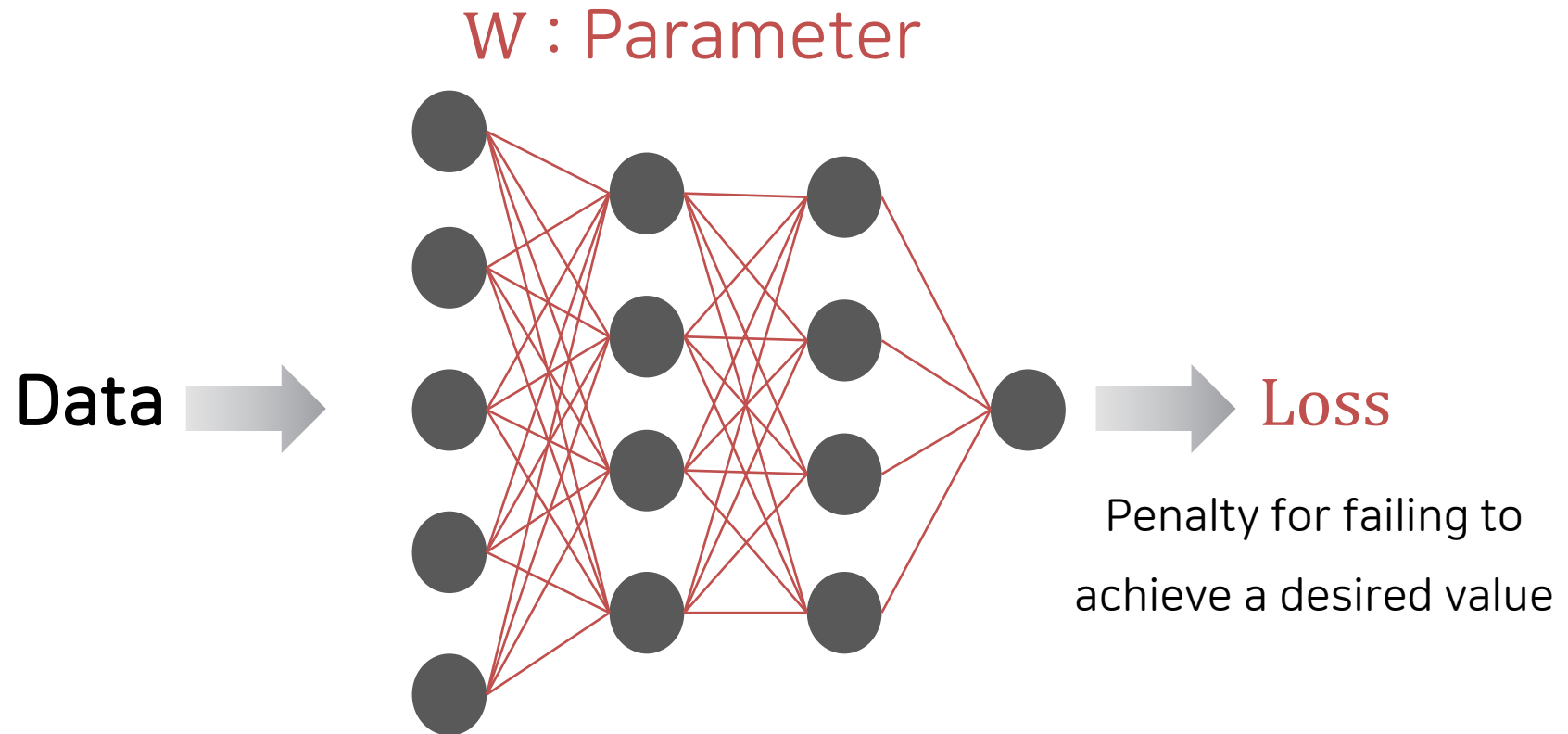
Keyword



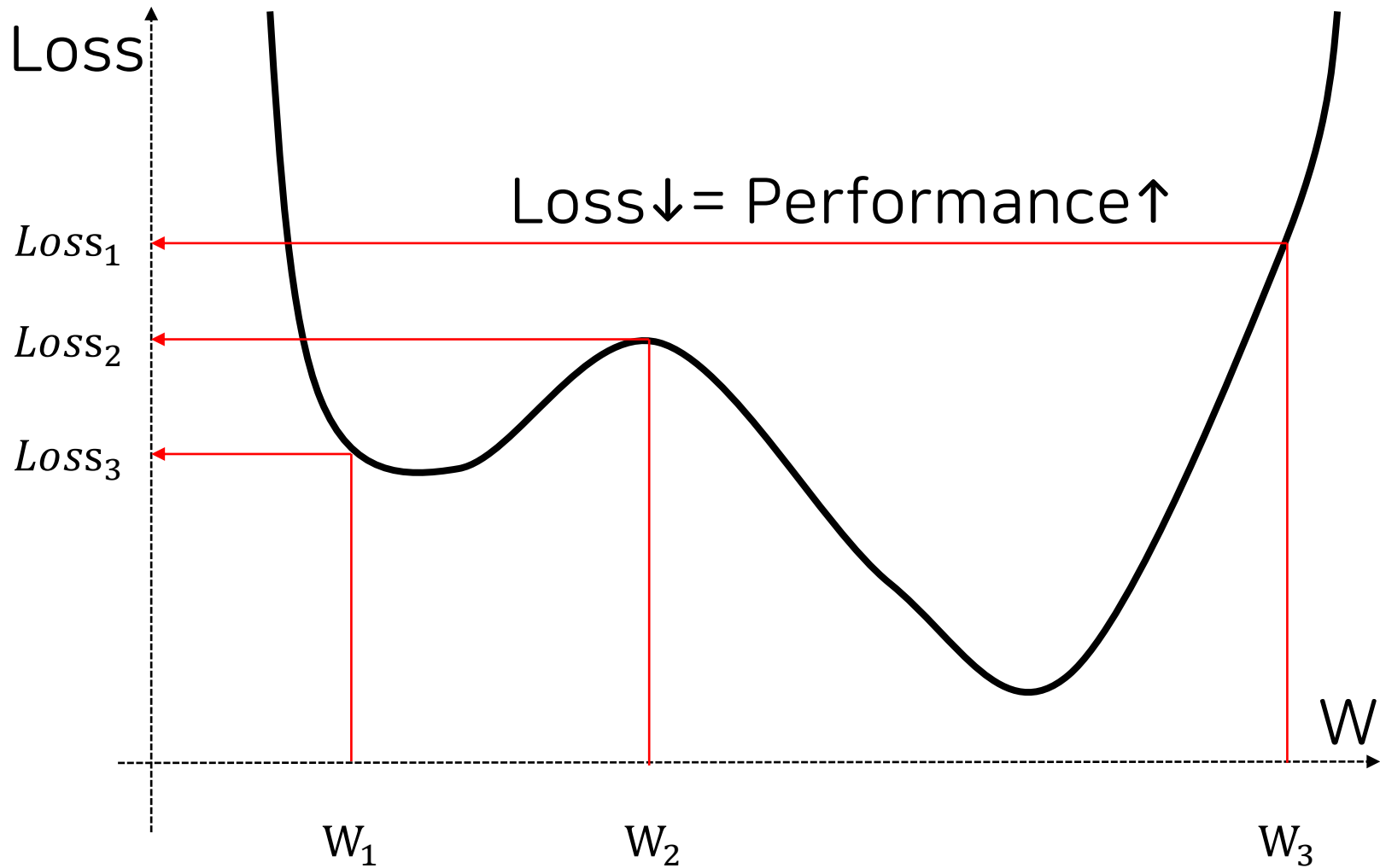
Optimizer



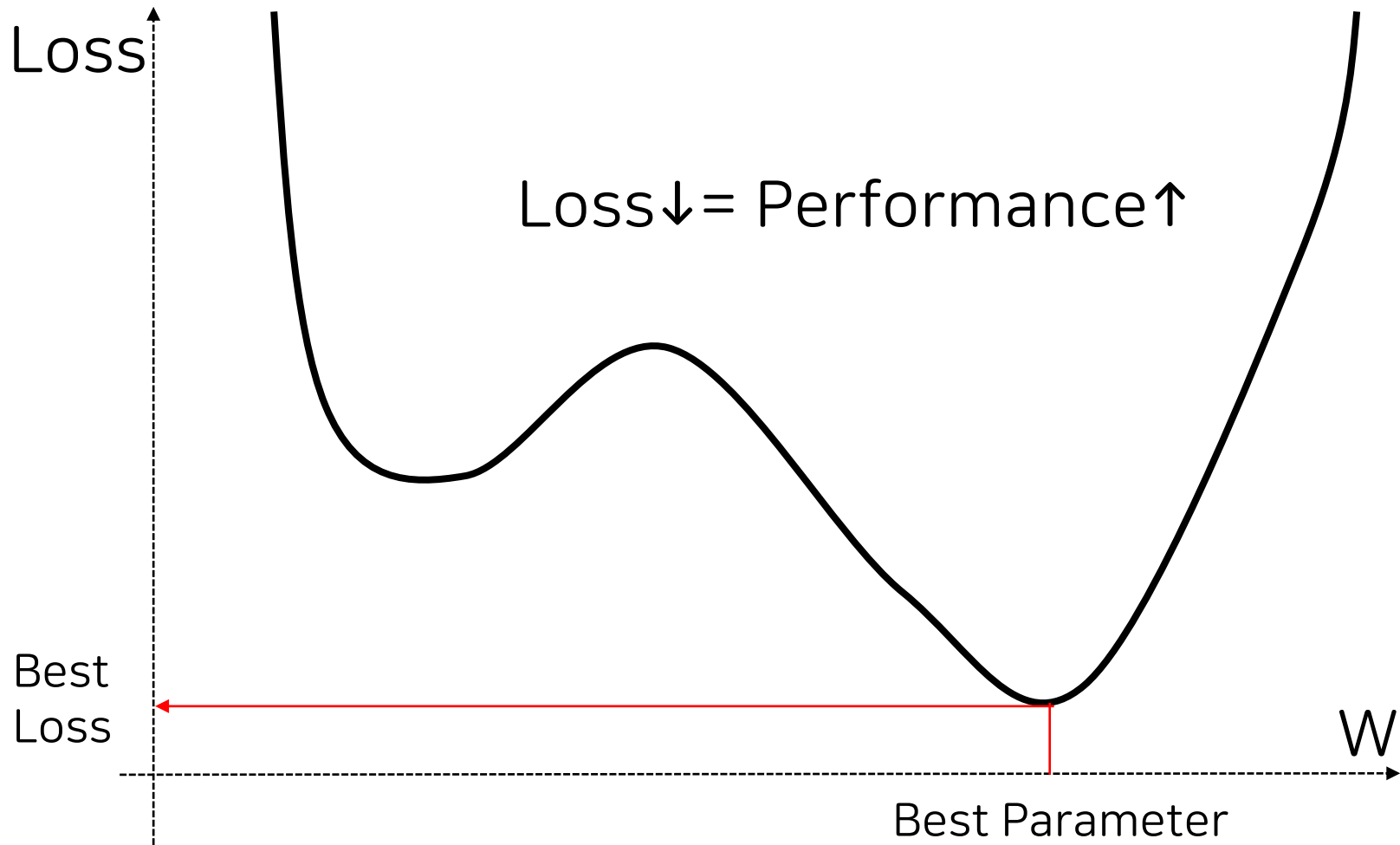
Optimizer



Optimizer



Optimizer

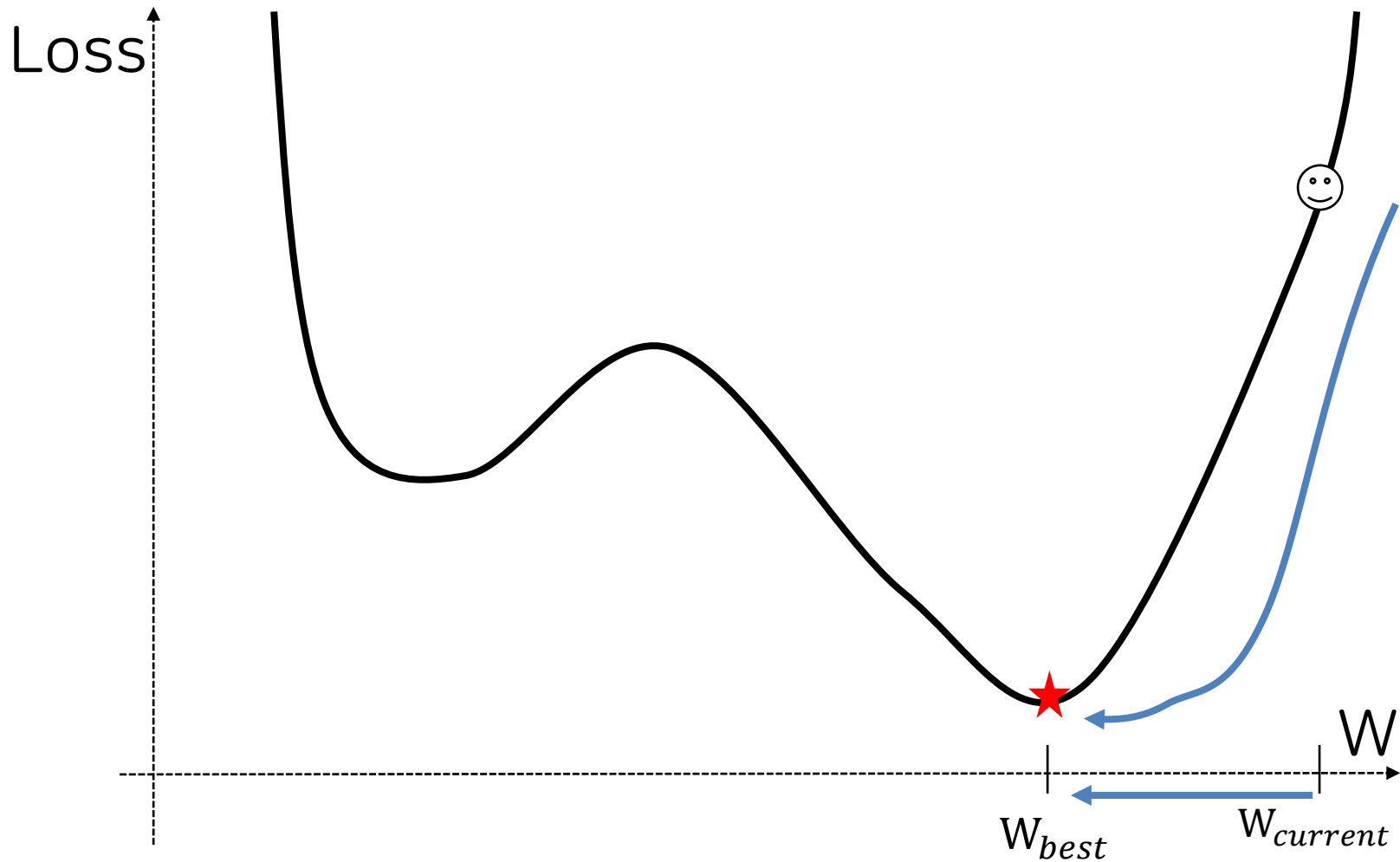


Optimizer

- minimize Loss_W
- find $\text{argmin}_W \text{Loss}_W$
- Training
- Who ? :Optimizer

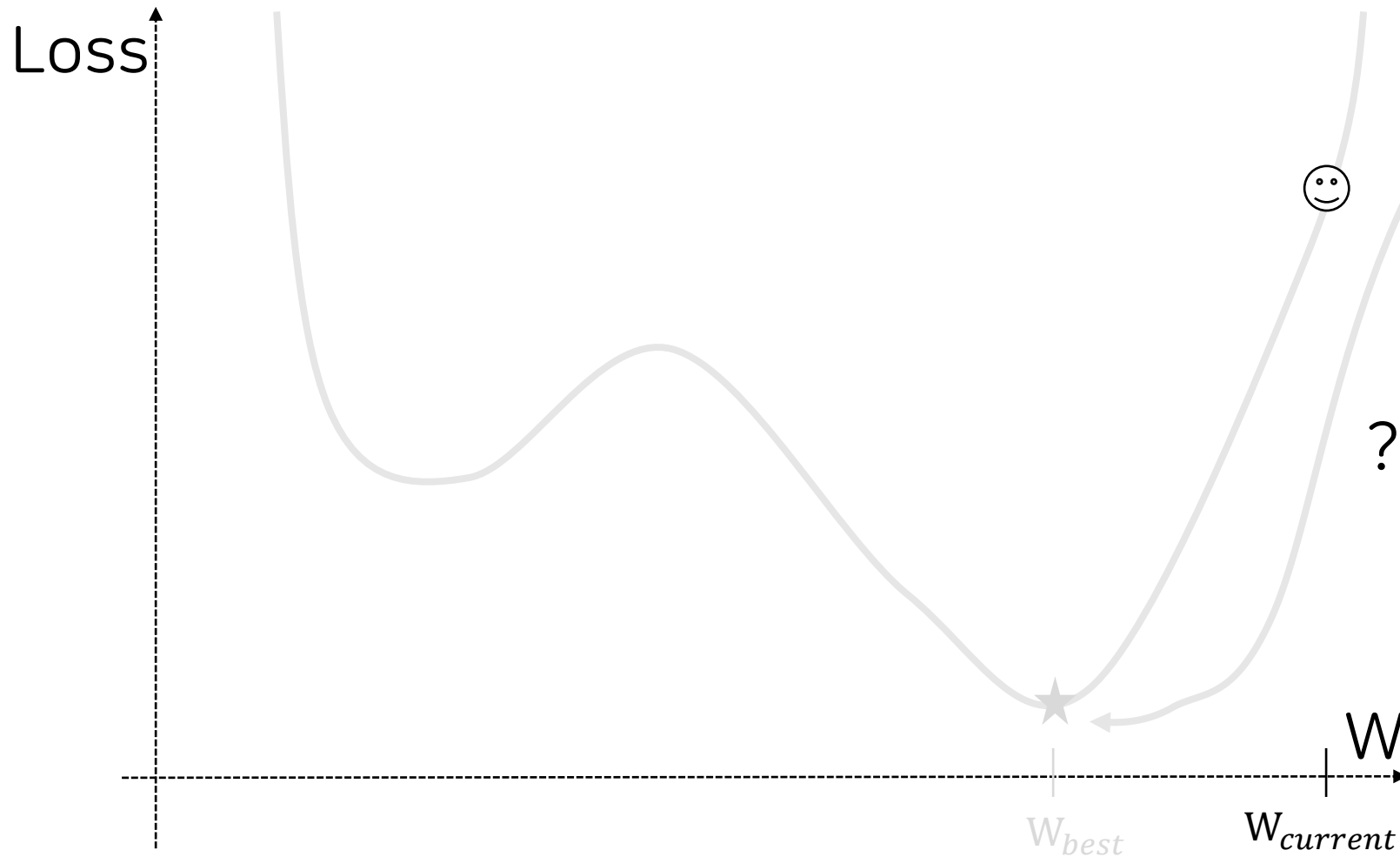
Optimizer

😊 : current
★ : best



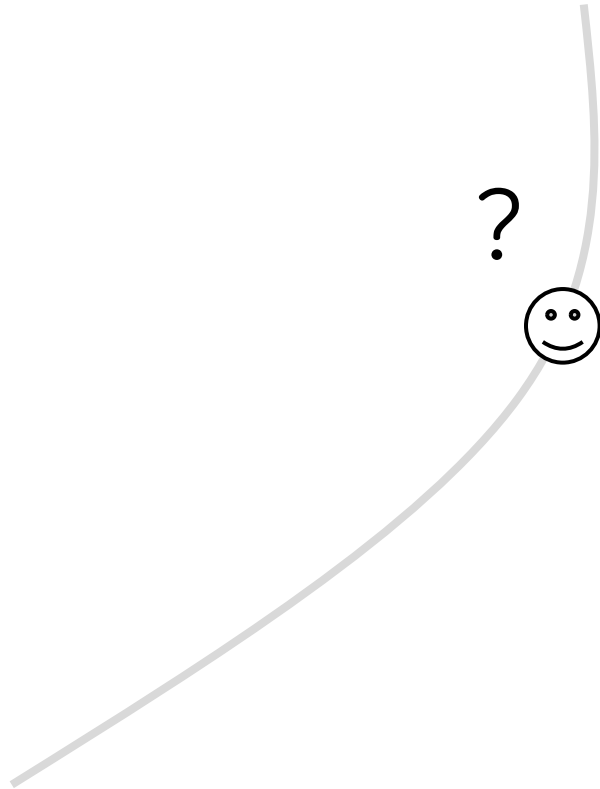
Gradient

□ : Unknown
■ : Known



Gradient

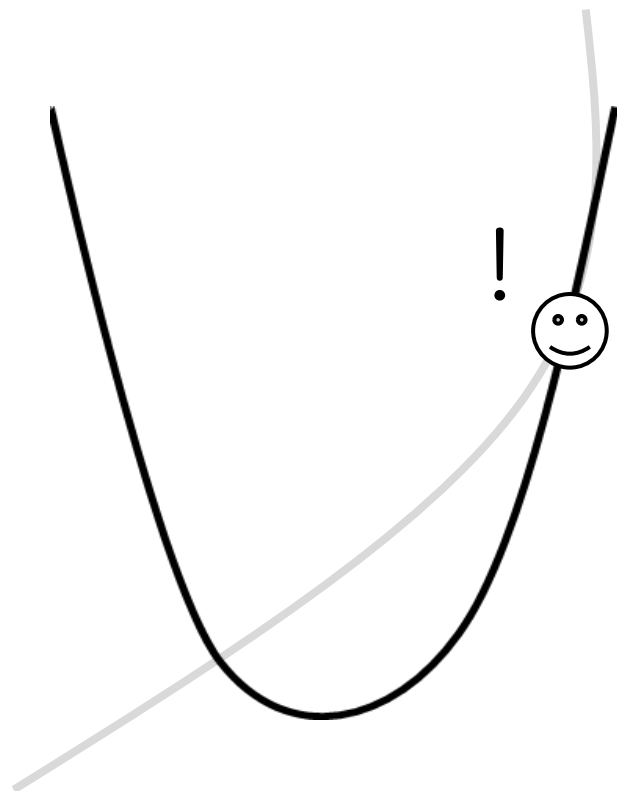
 : Unknown
 : Known



Brook Taylor

Gradient

□ : Unknown
■ : Known



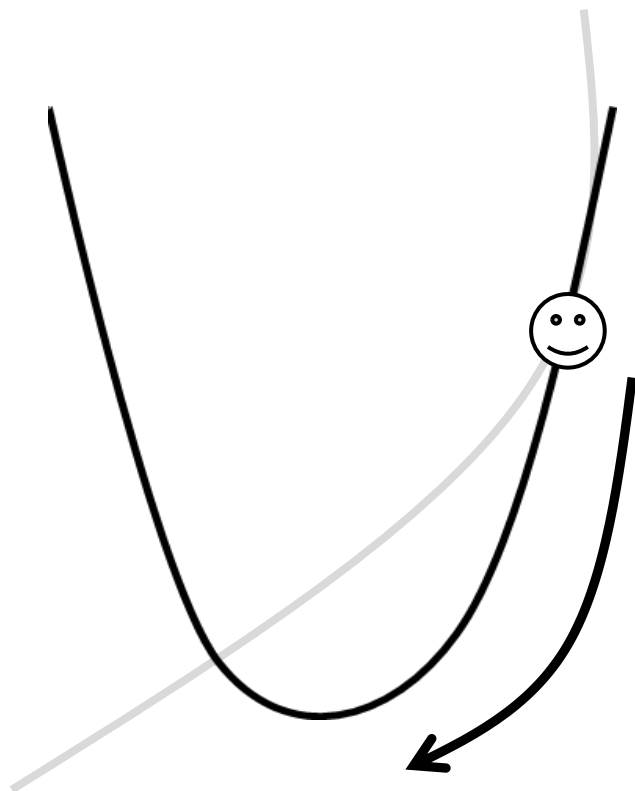
Quadratic approximation



Brook Taylor

Gradient

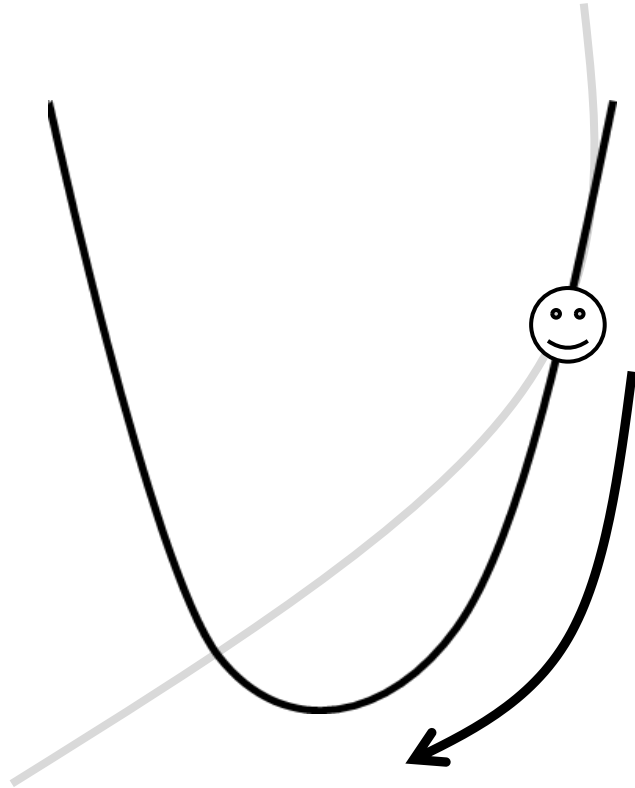
 : Unknown
 : Known



Gradient Descent
경사 하강법

Gradient

 : Unknown
 : Known

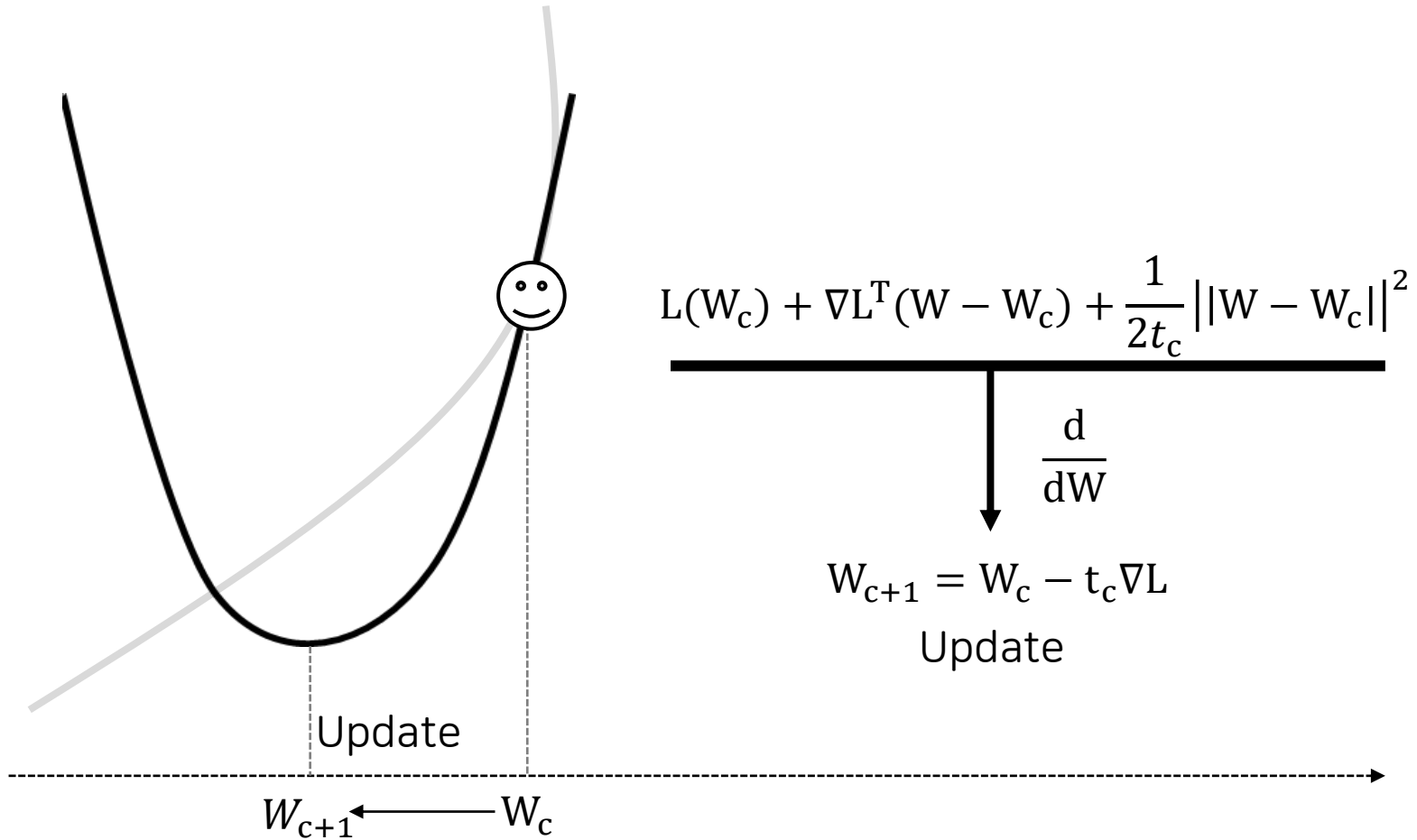


Gradient

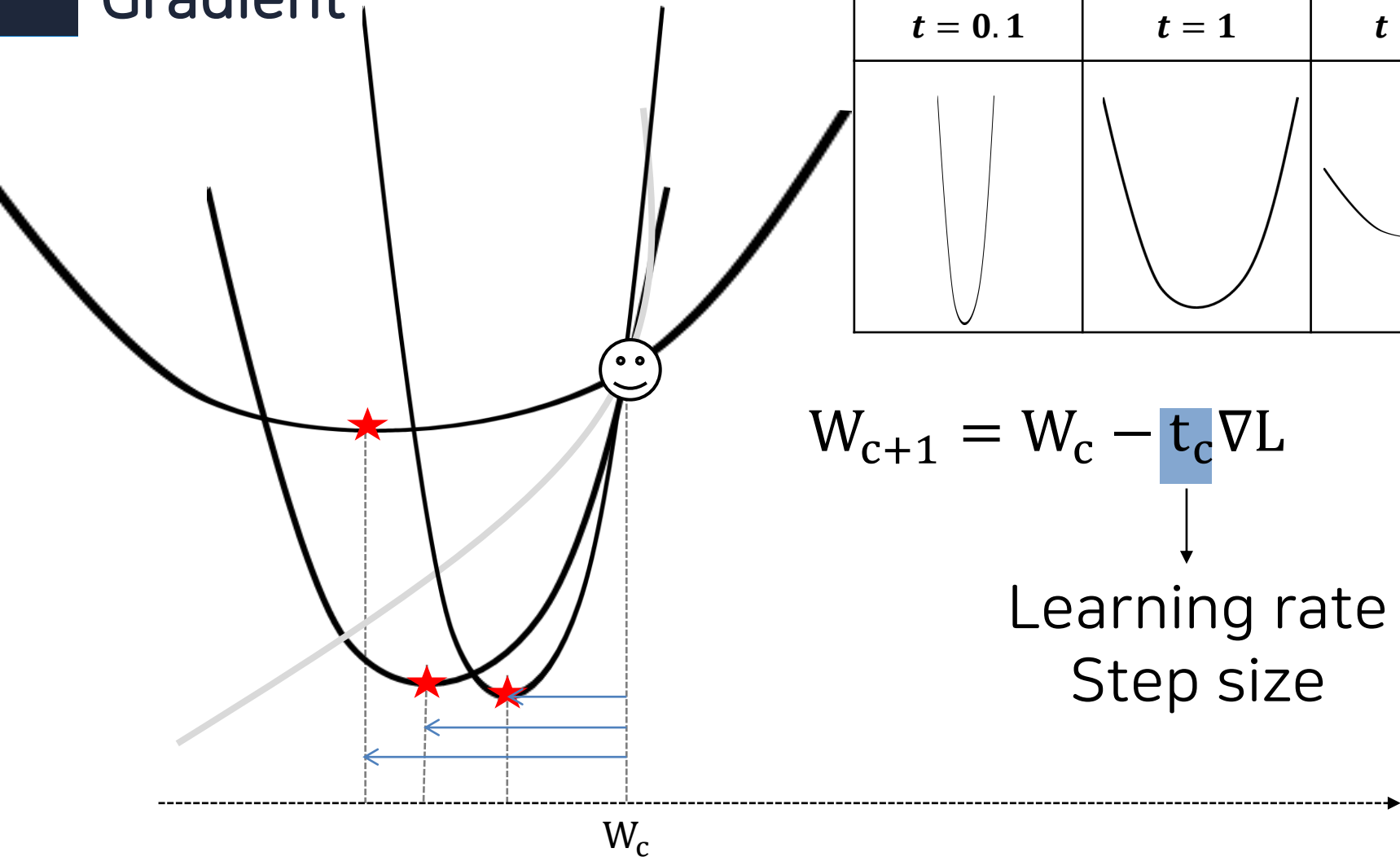
$$L(W_c) + \nabla L^T(W - W_c) + \frac{1}{2t_c} \|W - W_c\|^2$$

Quadratic approximation

Gradient



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 Descent

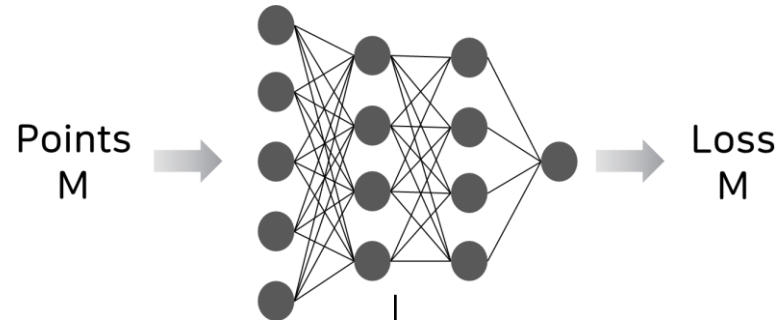
$$W_{c+1} = W_c - t_c \nabla L$$

Towards a lower Loss than the present

Move as much as Stepsize

Stochastic

- ❖ Consider minimizing an average of functions



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

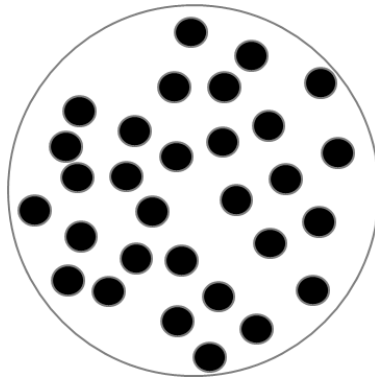
$$W_{c+1} = W_c - t_c \frac{1}{m} \sum_{i=1}^m \nabla L^i(W_c)$$

Stochastic

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

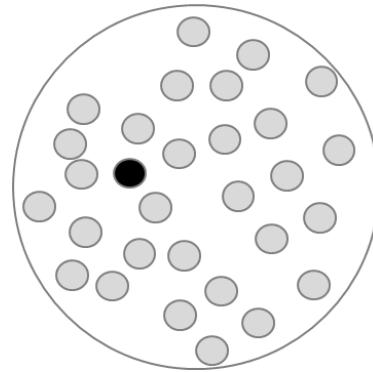
Gradient

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



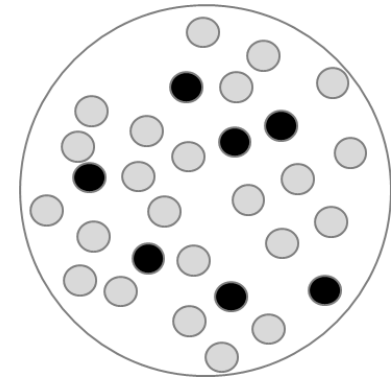
All

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



one

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



Partial

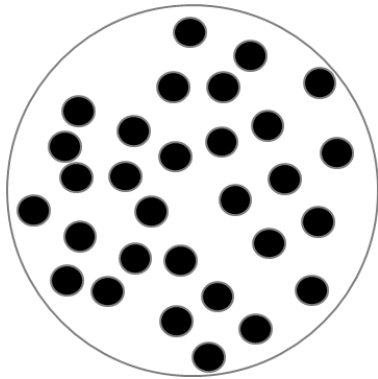
≈

≈

Stochastic

Full Gradient

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

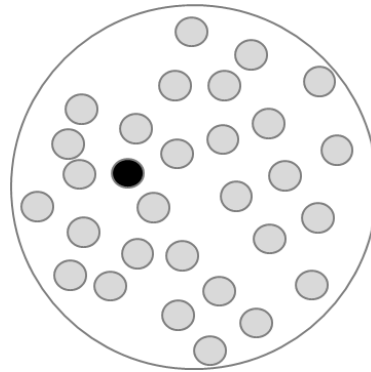


All

Stochastic Gradient

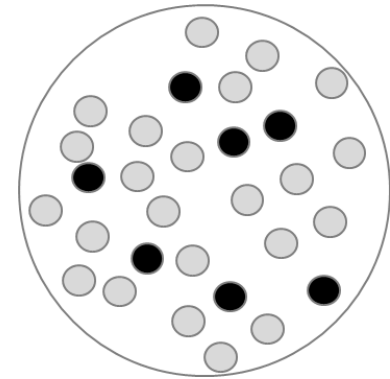
$$\nabla L^{i_c}(W_c)$$

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



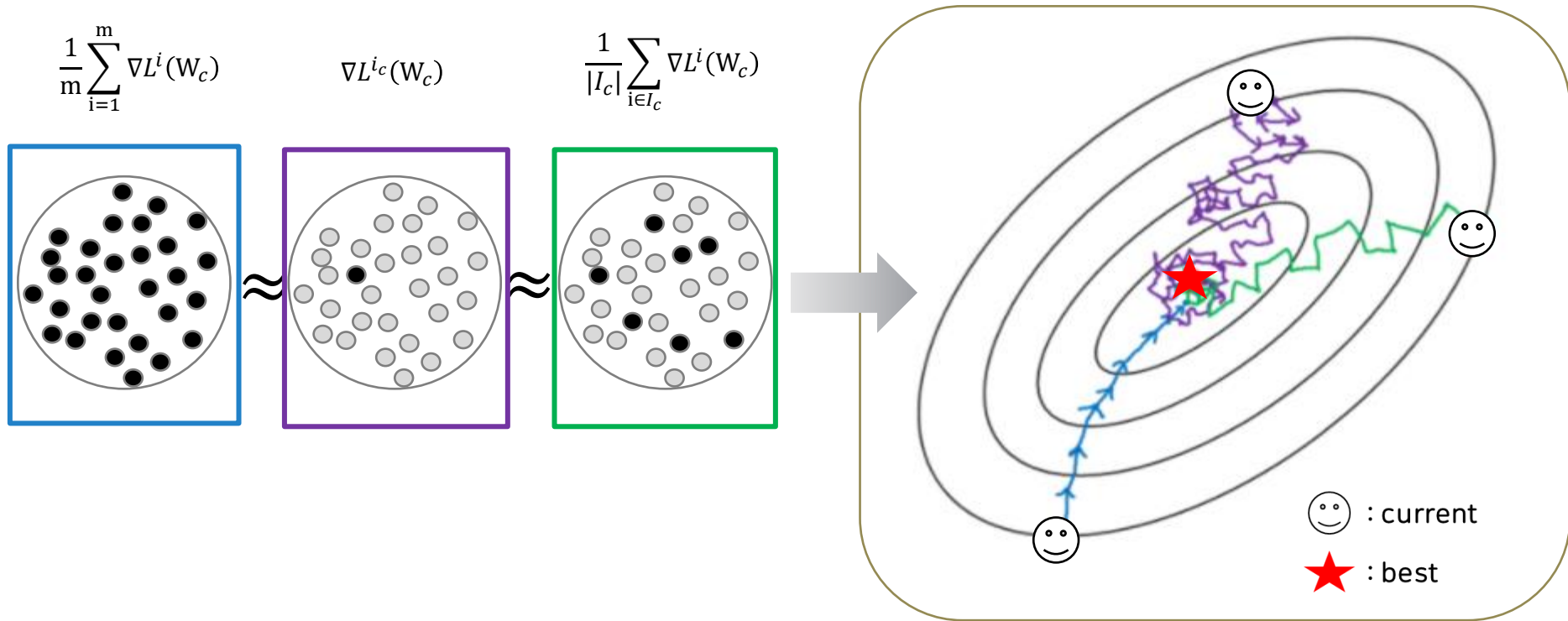
one

≈



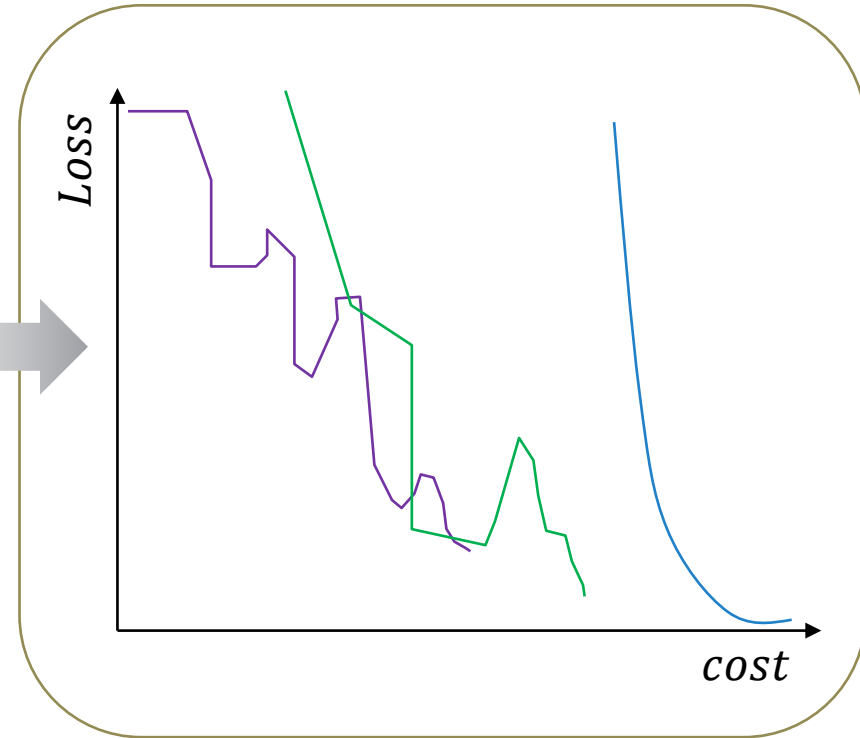
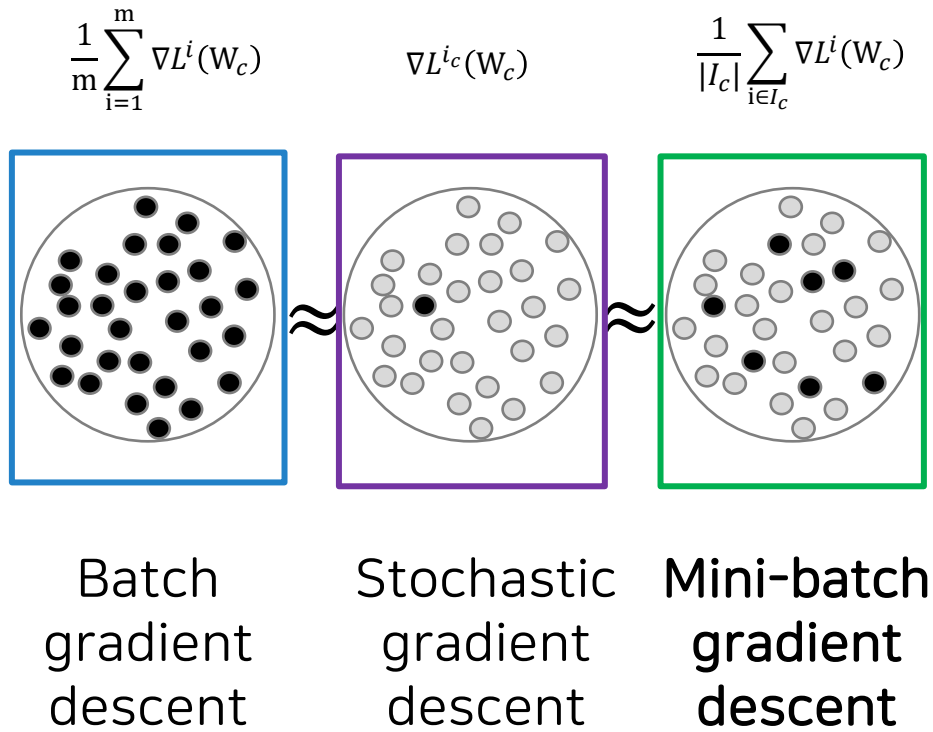
Partial

Stochastic



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

Stochastic



Momentum

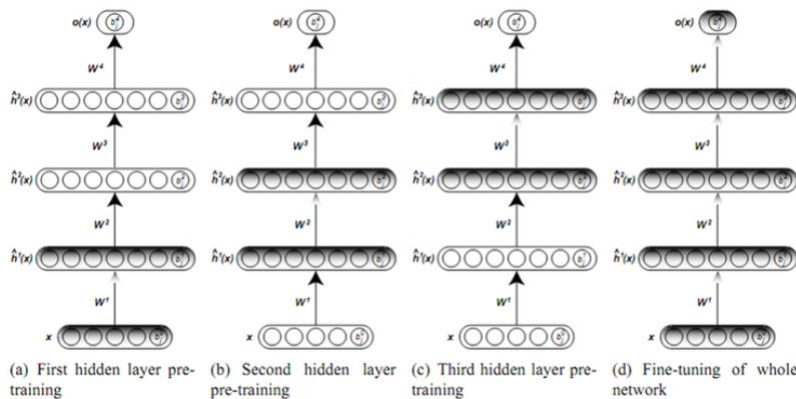
Mini-batch gradient descent

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



Greedy Layer-Wise Training

Second-order Optimization



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

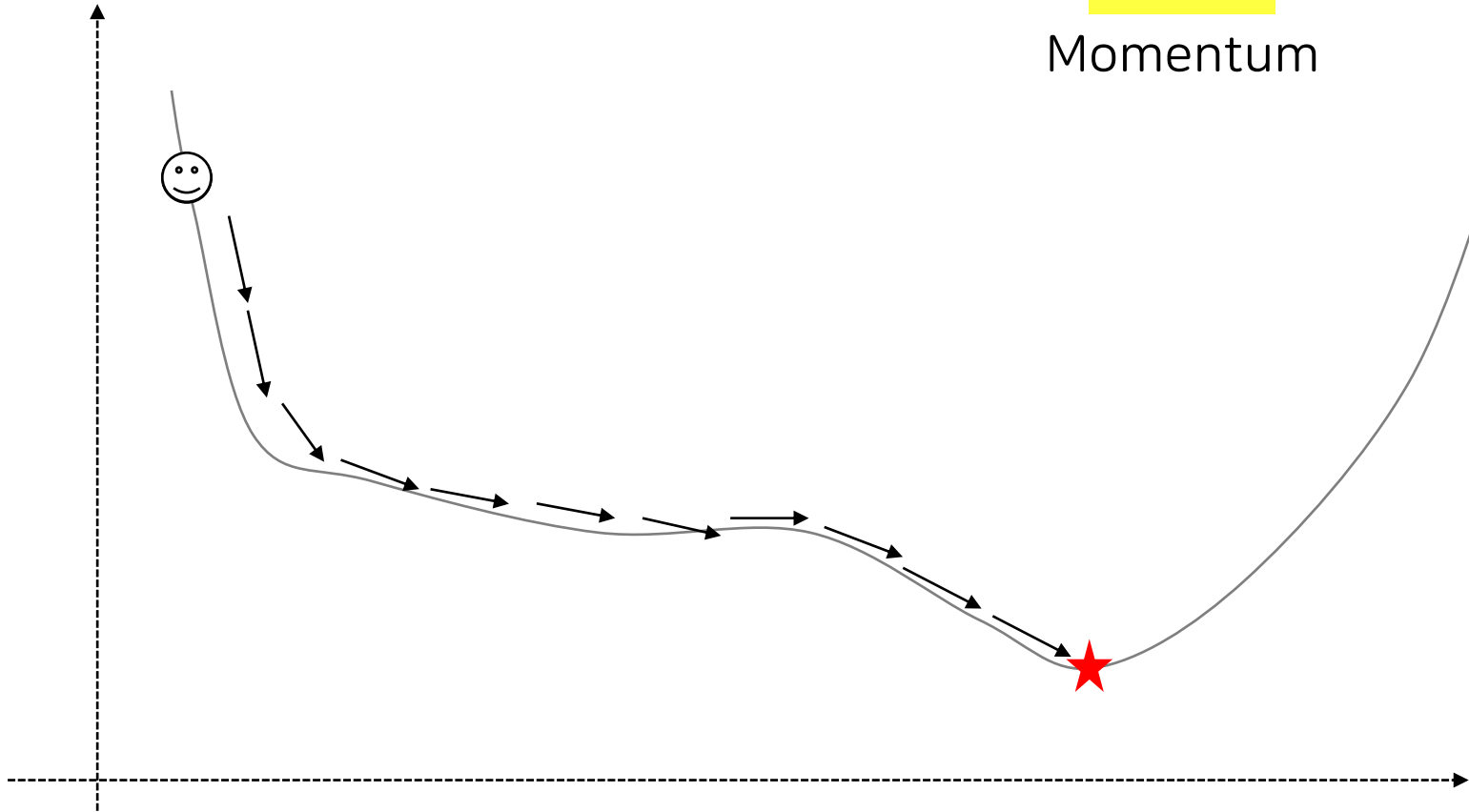
↓
Hessian

Momentum

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

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

Momentum

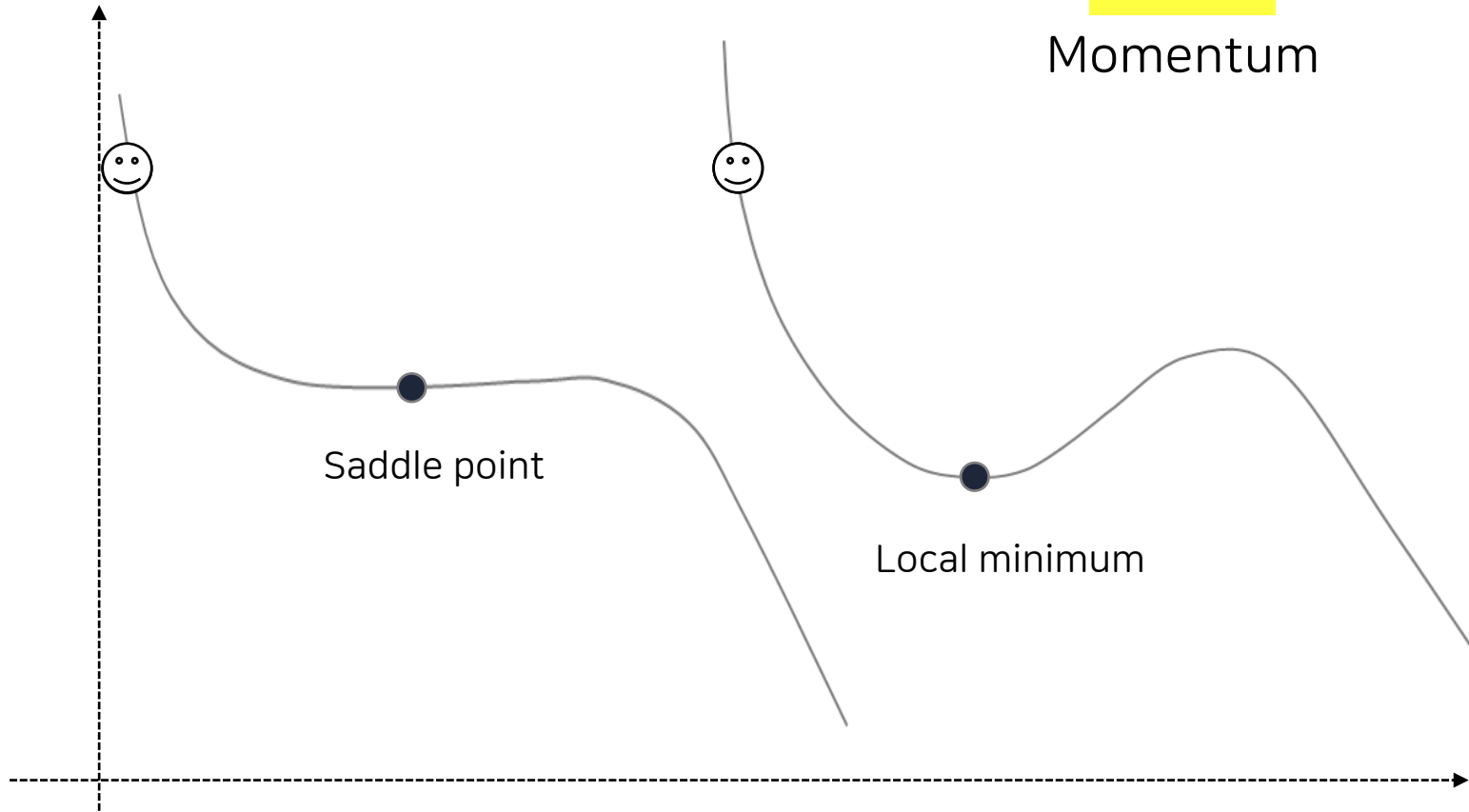


Momentum

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

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

Momentum



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

$$\begin{array}{c}
 \left[\begin{array}{c} v_{1,c} \\ v_{2,c} \\ v_{3,c} \\ \vdots \\ v_{n,c} \end{array} \right] \\
 n \text{ by } 1
 \end{array}
 = \mu
 \begin{array}{c}
 \left[\begin{array}{c} v_{1,c-1} \\ v_{2,c-1} \\ v_{3,c-1} \\ \vdots \\ v_{n,c-1} \end{array} \right] \\
 n \text{ by } 1
 \end{array}
 - t_c
 \begin{array}{c}
 \left[\begin{array}{c} \nabla_1 L(W_{1,c}) \\ \nabla_2 L(W_{2,c}) \\ \nabla_3 L(W_{3,c}) \\ \vdots \\ \nabla_n L(W_{n,c}) \end{array} \right] \\
 n \text{ by } 1
 \end{array}$$

Adaptive

n : # parameters

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

↓

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

n by 1 *n by 1* *n by 1*

Global stepsize

Adaptive

n : # parameters

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



To have its own dynamic stepsize

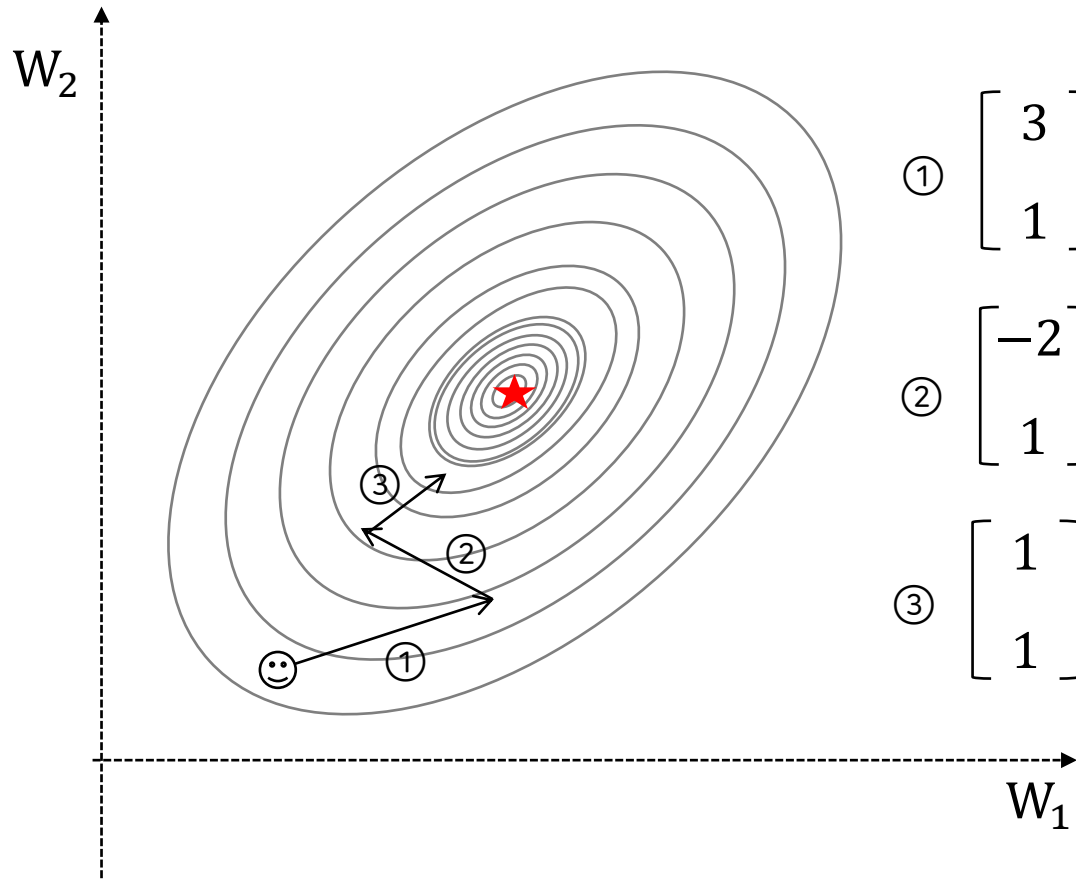
$$\begin{bmatrix} v_{1,c} \\ v_{2,c} \\ v_{3,c} \\ \vdots \\ v_{n,c} \end{bmatrix} = \mu \begin{bmatrix} v_{1,c-1} \\ v_{2,c-1} \\ v_{3,c-1} \\ \vdots \\ v_{n,c-1} \end{bmatrix} - \begin{bmatrix} \widetilde{t}_{1,c} \nabla_1 L(W_{1,c}) \\ \widetilde{t}_{2,c} \nabla_2 L(W_{2,c}) \\ \widetilde{t}_{3,c}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

n by 1 n by 1 n by 1

Adaptive

😊 : current

★ : best

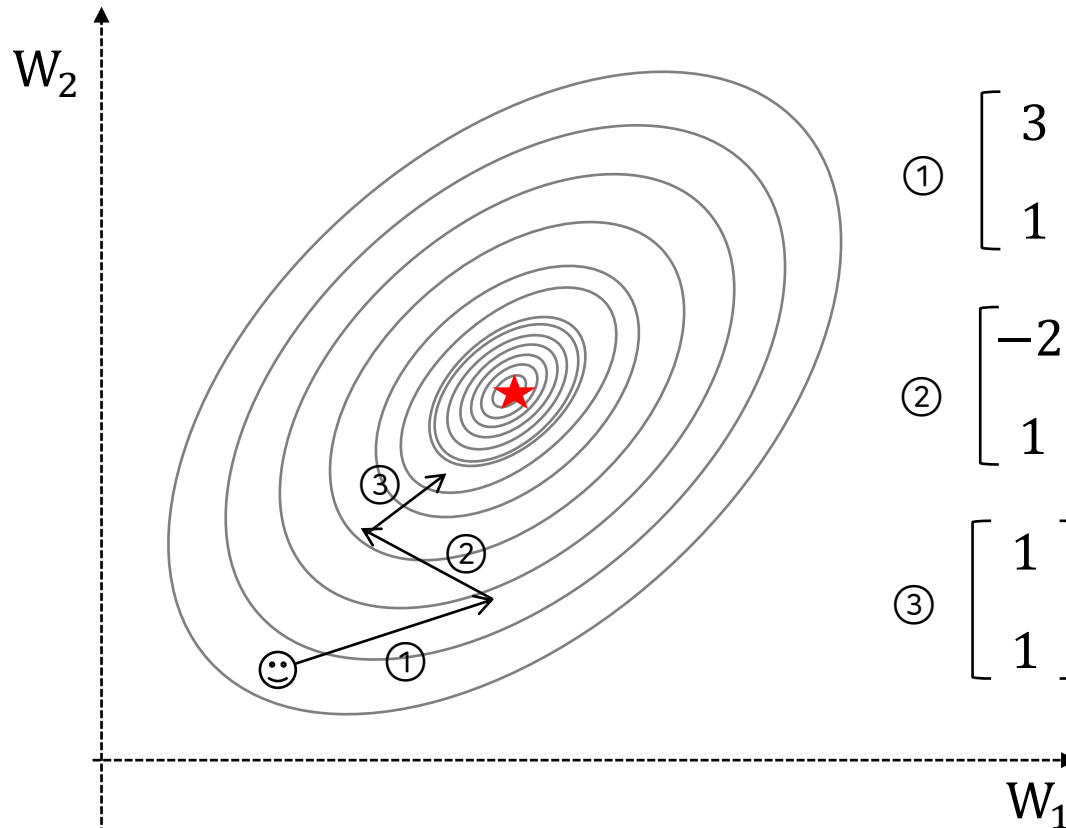


$$\begin{array}{l} \textcircled{1} \begin{bmatrix} 3 \\ 1 \end{bmatrix} \xrightarrow{\wedge 2} \begin{bmatrix} 3^2 \\ 1^2 \end{bmatrix} \rightarrow \begin{bmatrix} 9 \\ 1 \end{bmatrix} \\ \textcircled{2} \begin{bmatrix} -2 \\ 1 \end{bmatrix} \xrightarrow{\wedge 2} \begin{bmatrix} (-2)^2 \\ 1^2 \end{bmatrix} \rightarrow \begin{bmatrix} 13 \\ 2 \end{bmatrix} \\ \textcircled{3} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \xrightarrow{\wedge 2} \begin{bmatrix} 1^2 \\ 1^2 \end{bmatrix} \rightarrow \begin{bmatrix} 14 \\ 3 \end{bmatrix} \end{array}$$

Adaptive

😊 : current

★ : best



$$\begin{array}{l}
 \textcircled{1} \begin{bmatrix} 3 \\ 1 \end{bmatrix} \xrightarrow{\wedge 2} \begin{bmatrix} 3^2 \\ 1^2 \end{bmatrix} \rightarrow \begin{bmatrix} 9 \\ 1 \end{bmatrix} \\
 \textcircled{2} \begin{bmatrix} -2 \\ 1 \end{bmatrix} \xrightarrow{\wedge 2} \begin{bmatrix} (-2)^2 \\ 1^2 \end{bmatrix} \rightarrow \begin{bmatrix} 13 \\ 2 \end{bmatrix} \\
 \textcircled{3} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \xrightarrow{\wedge 2} \begin{bmatrix} 1^2 \\ 1^2 \end{bmatrix} \rightarrow \begin{bmatrix} 14 \\ 3 \end{bmatrix}
 \end{array}$$

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

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

W_1 stepsize ↓
 W_2 stepsize ↑

Adaptive

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

- 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)

$\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)

$\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)

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

end while

return θ_t (Resulting parameters)

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

```
Require :  $\alpha$ : Stepsize
Require :  $\beta_1, \beta_2 \in [0,1)$ 
Require :  $f(\theta)$ : Loss function
Require :  $\theta_0$  Initial parameter vector
 $m_0, v_0, t_0 \leftarrow [0,0,0]$ 
while  $\theta_t$  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  $\theta_t$ 
```

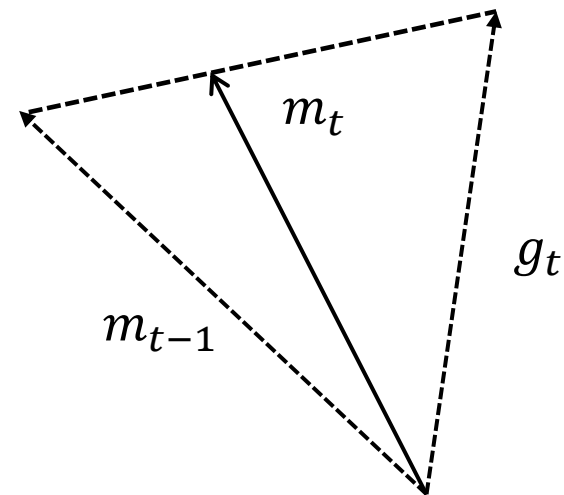
- High-dimensional parameter
- Little memory requirement
- Naturally step size annealing
- Invariant to re-scaling of gradient

ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

```
Require :  $\alpha$ : Stepsize
Require :  $\beta_1, \beta_2 \in [0,1)$ 
Require :  $f(\theta)$ : Loss function
Require :  $\theta_0$  Initial parameter vector
 $m_0, v_0, t_0 \leftarrow [0,0,0]$ 
while  $\theta_t$  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  $\theta_t$ 
```

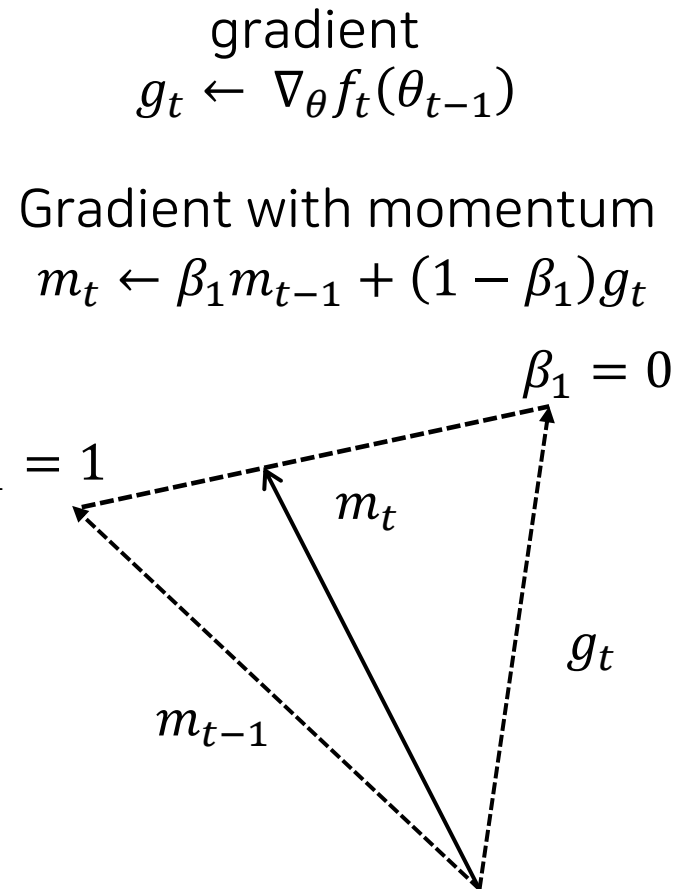
gradient
 $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$

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

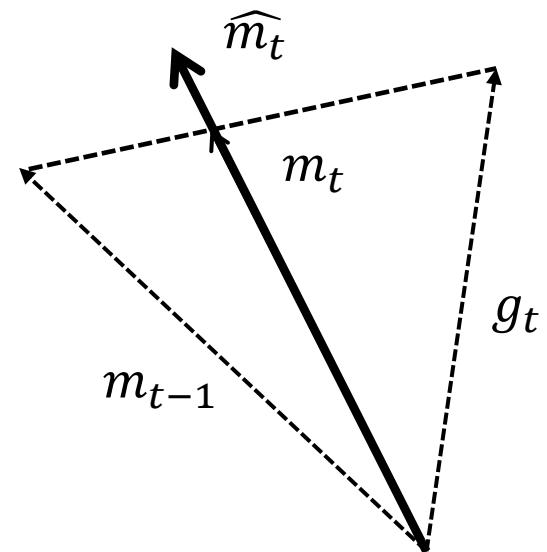


ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION

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

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$
$$\frac{1}{\text{stepsize}} \propto \sum_{q=1}^c \{\nabla_k L(W_{k,q})\}^2$$

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

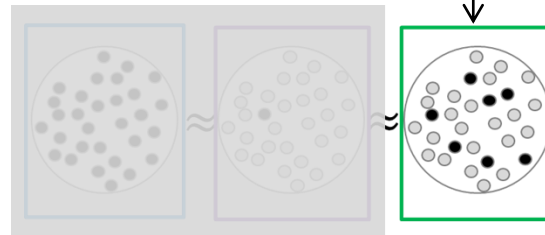
Require : α : Stepsize
Require : $\beta_1, \beta_2 \in [0,1)$
Require : $f(\theta)$: Loss function
Require : θ_0 Initial parameter vector
 $m_0, v_0, t_0 \leftarrow [0,0,0]$
while θ_t not converge do
 $t \leftarrow t + 1$
 $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$
 $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$
 $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
 $\widehat{m}_t \leftarrow \frac{m_t}{1 - \beta_1^t}$
 $\widehat{v}_t \leftarrow \frac{v_t}{1 - \beta_2^t}$
 $\theta_t \leftarrow \theta_{t-1} - \widehat{m}_t \alpha / (\sqrt{\widehat{v}_t} + \epsilon)$
end *while*
return θ_t

Gradient and Momentum

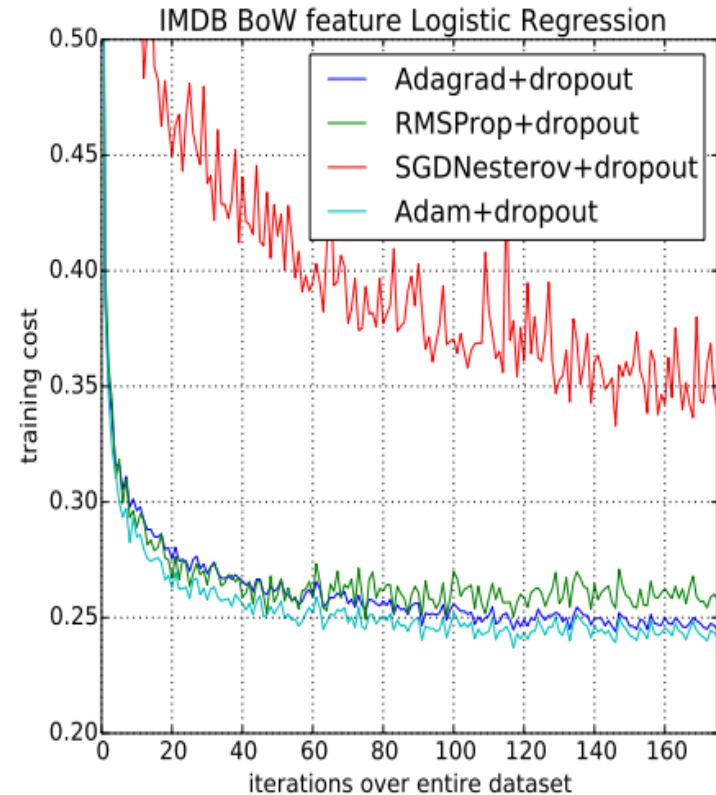
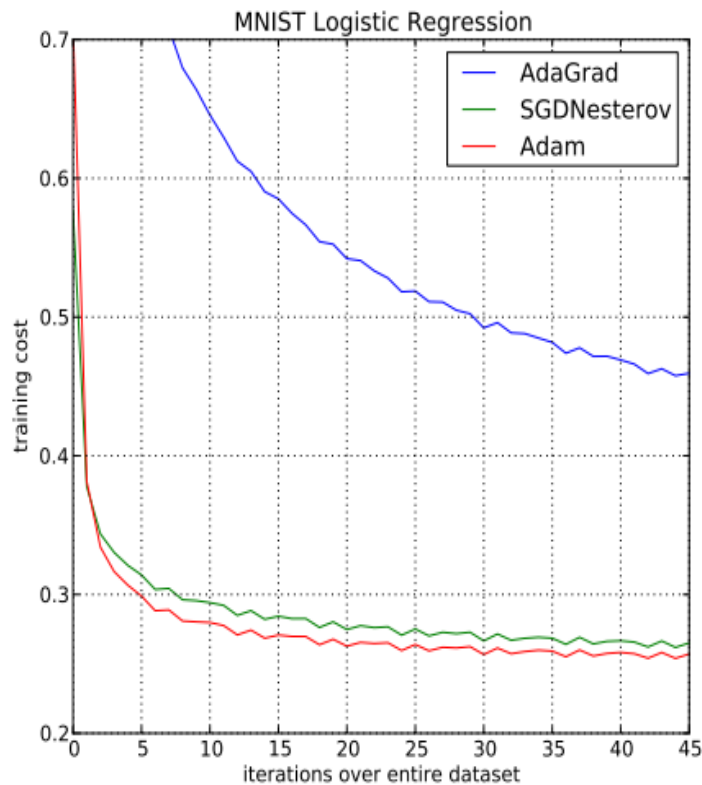
$$\theta_t \leftarrow \theta_{t-1} - \widehat{m}_t \alpha / (\sqrt{\widehat{v}_t} + \epsilon)$$

Adaptive

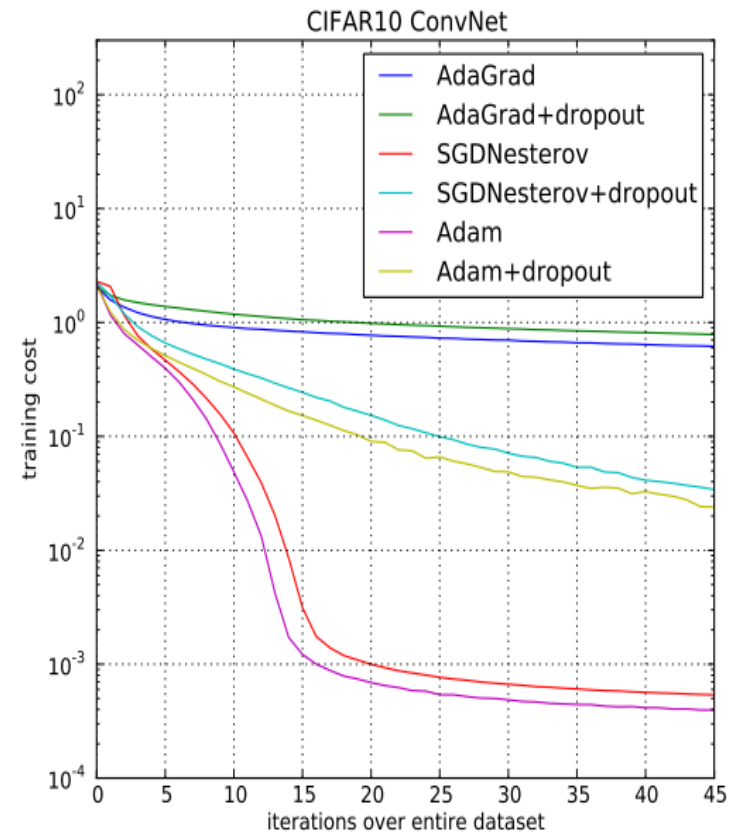
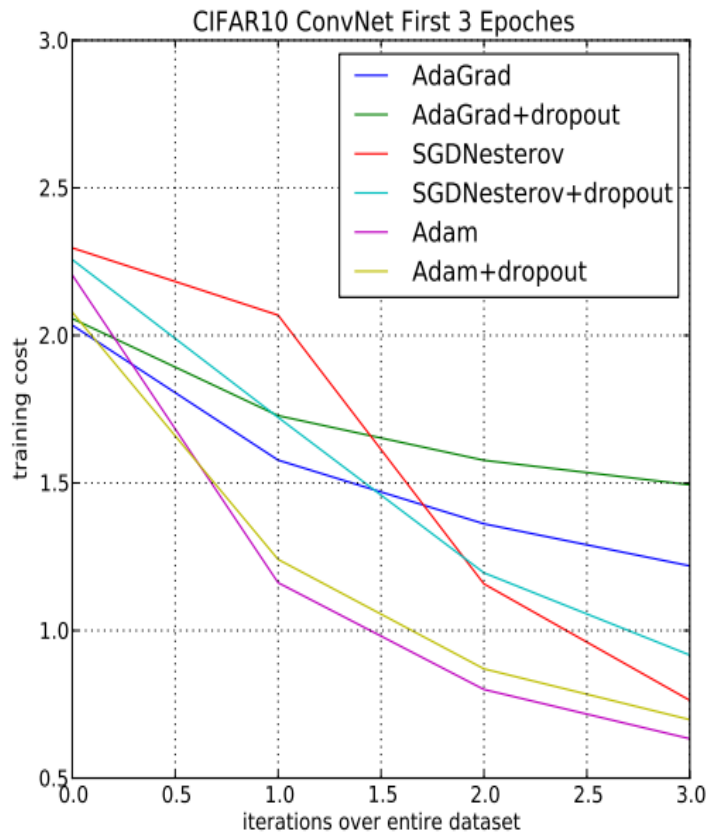
Stochastic



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

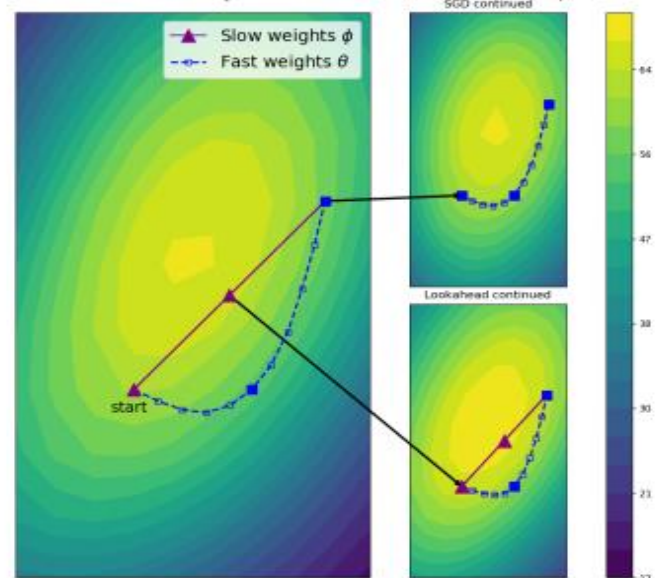
Lookahead Optimizer: k steps forward, 1 step back

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.

CIFAR-100 accuracy surface with Lookahead interpolation

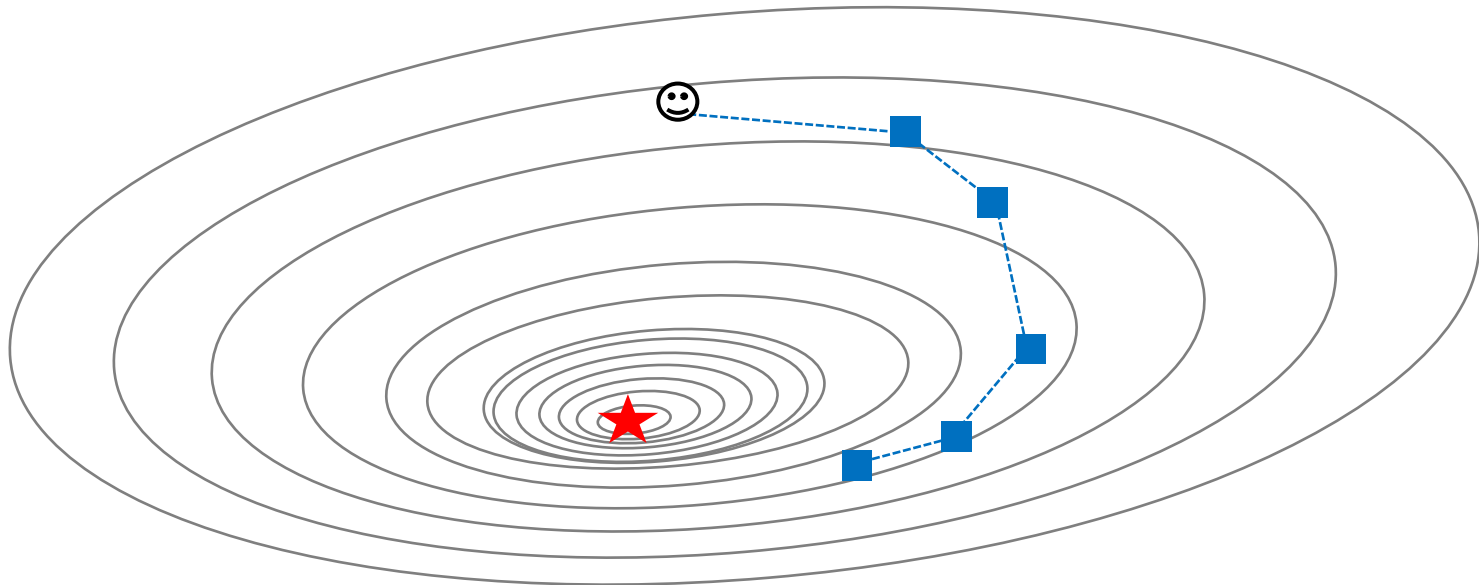


Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

(ex. $k = 5$)

Adam or SGD



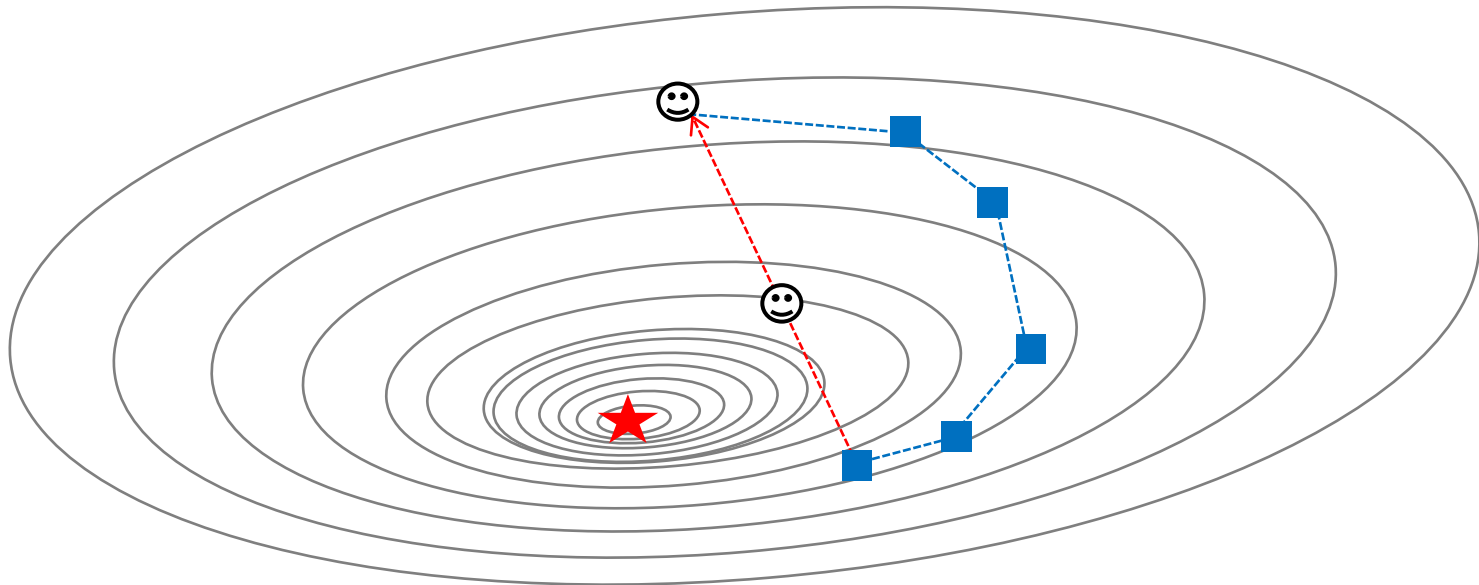
Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

(ex. $k = 5$)

Adam or SGD

interpolation



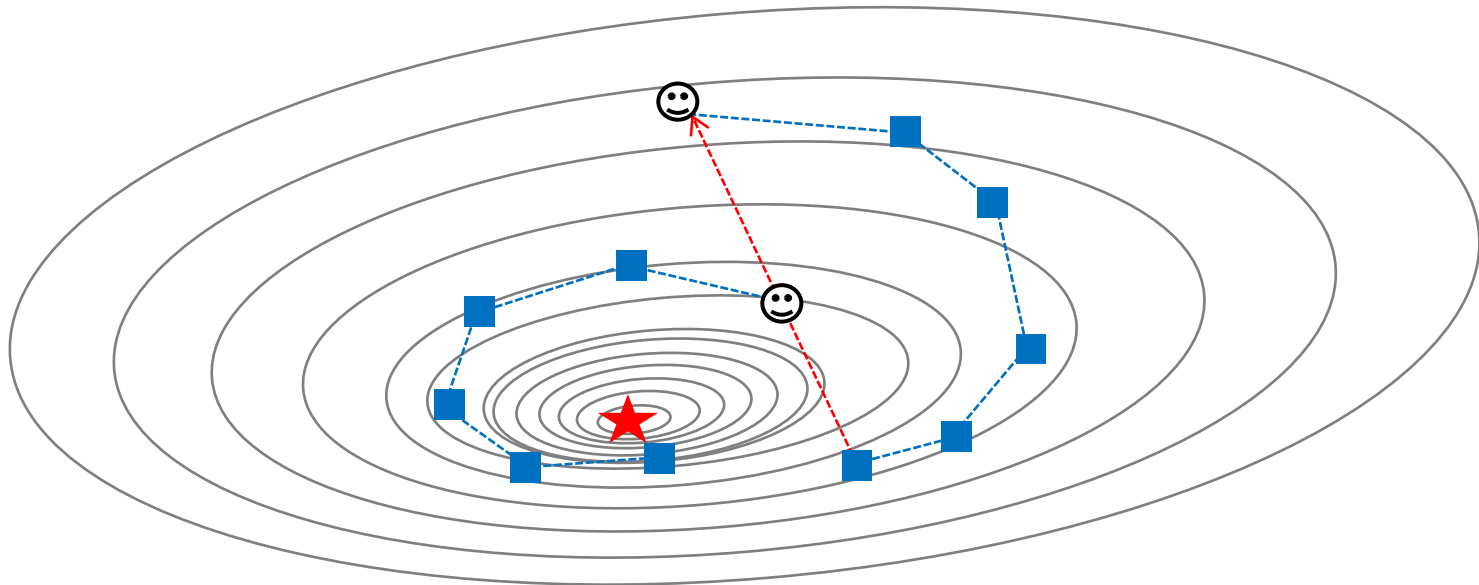
Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

(ex. $k = 5$)

Adam or SGD

interpolation



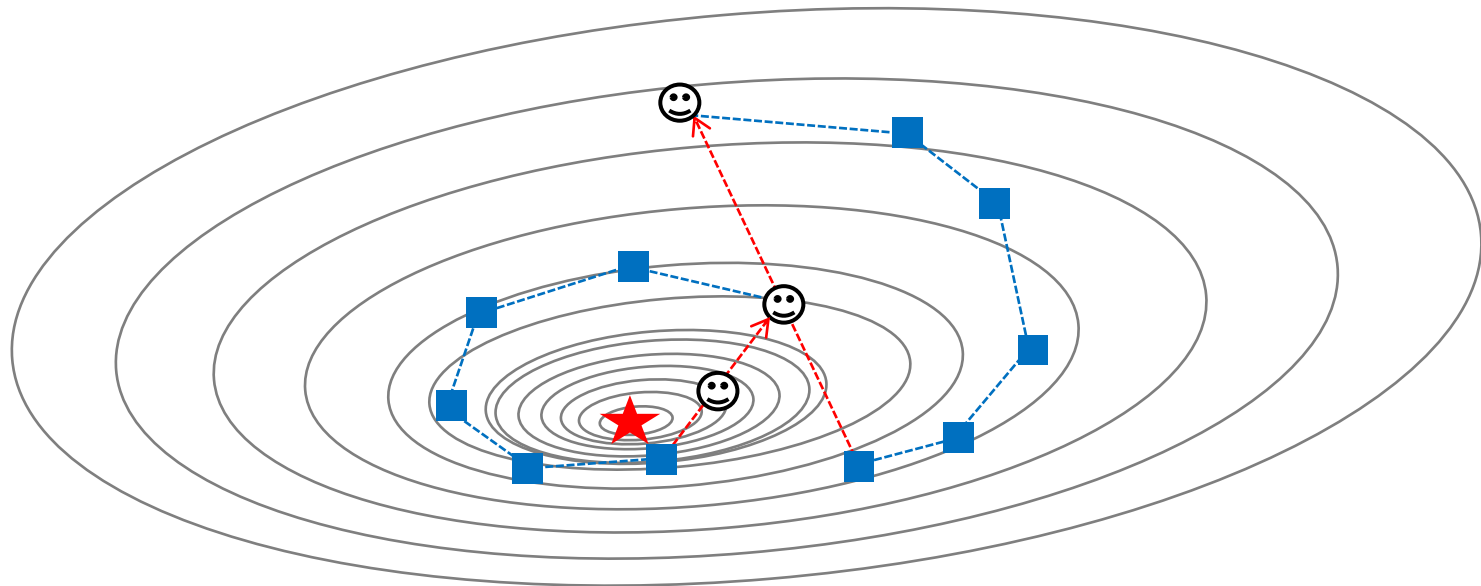
Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

(ex. $k = 5$)

Adam or SGD

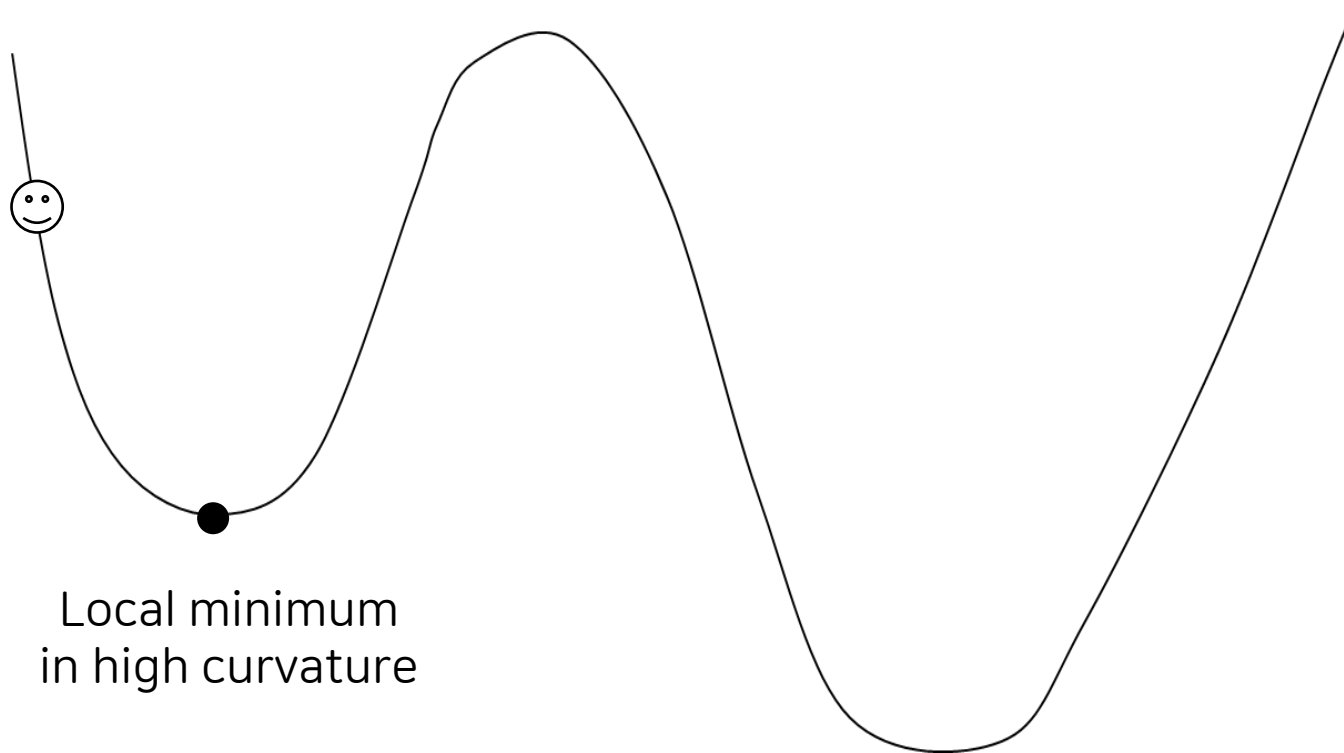
interpolation



Lookahead Optimizer: k steps forward, 1 step back

❖ SGD or Adam

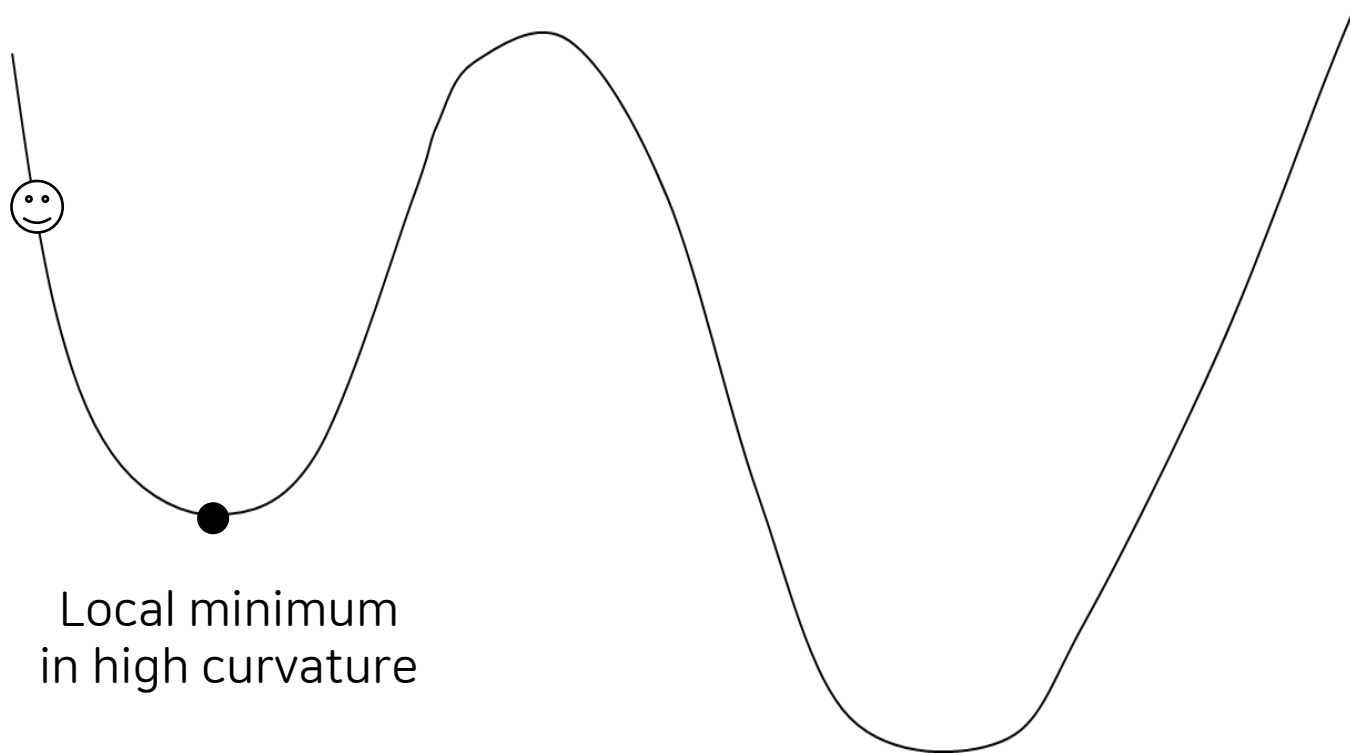
→ It is difficult to escape local minima in high curvature directions.



Lookahead Optimizer: k steps forward, 1 step back

k steps forward, 1 step back

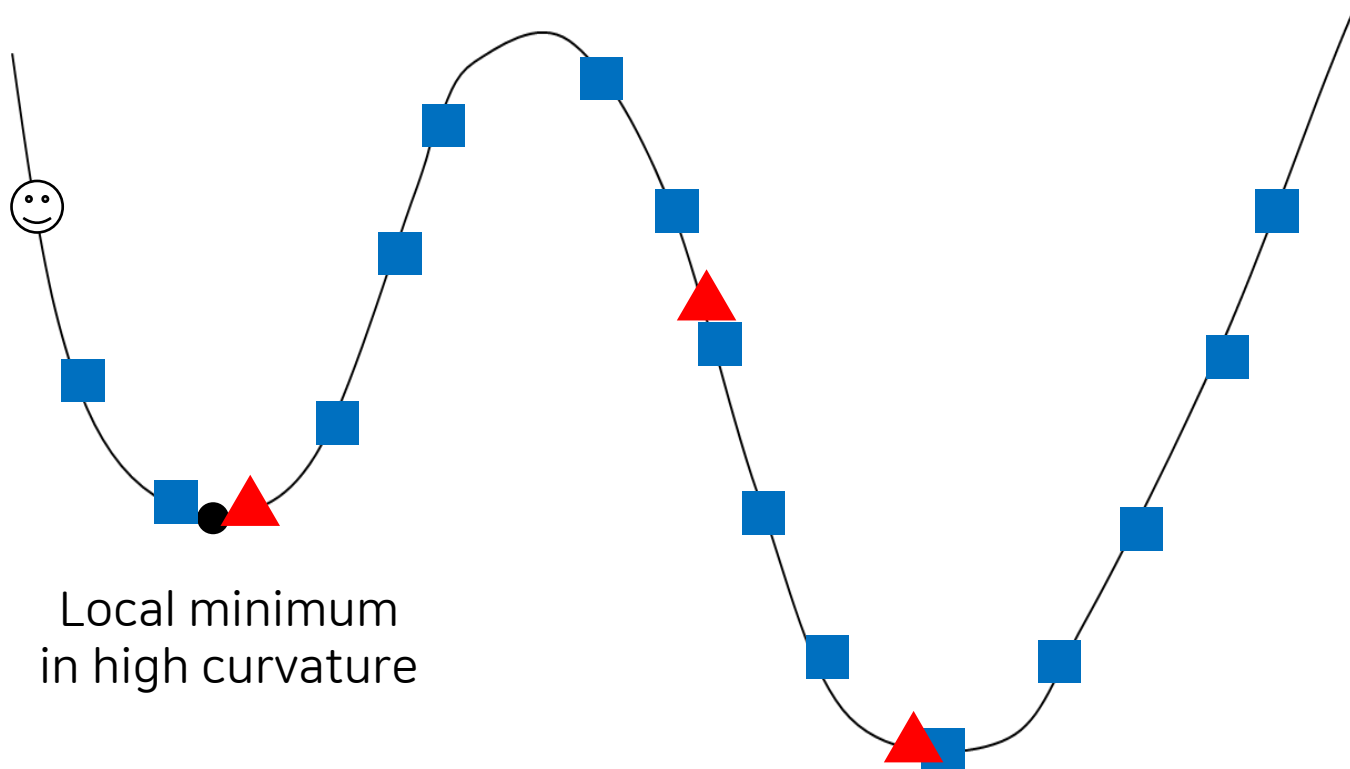
- SGD or Adam
- Large stepsize



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



Lookahead Optimizer: k steps forward, 1 step back

Require : Initial parameter ϕ_0 , Loss function L

Require : Synchronization period k ,

slow step size α , optimizer A

for $t = 1, 2, \dots$ do

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

for $i = 1, 2, \dots, k$ do

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

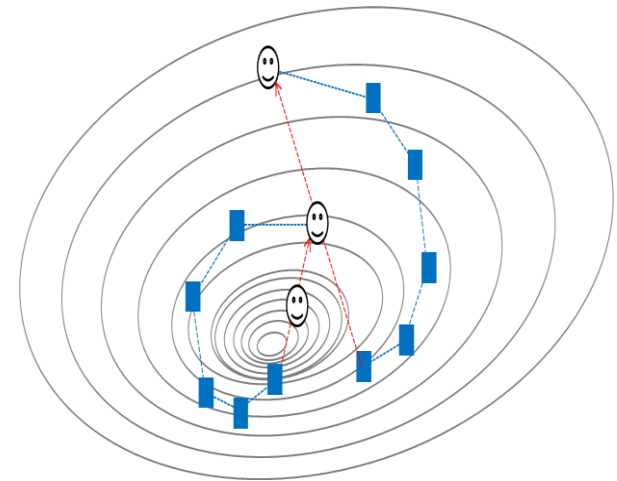
$\theta_{t,i} \leftarrow \theta_{t,i-1} + A(L, \theta_{t,i-1}, d)$

end for

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

end for

return parameters ϕ



Lookahead Optimizer: k steps forward, 1 step back

Require : Initial parameter ϕ_0 , Loss function L

Require : Synchronization period k ,

slow step size α , optimizer A

for $t = 1, 2, \dots$ do

Slow weight

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

for $i = 1, 2, \dots, k$ do Fast weight

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

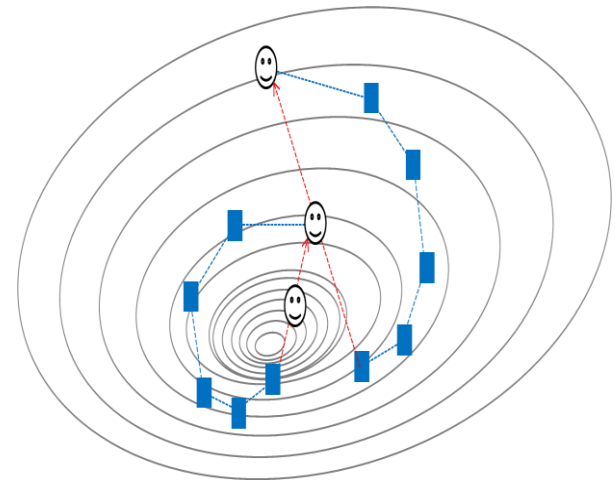
$\theta_{t,i} \leftarrow \theta_{t,i-1} + A(L, \theta_{t,i-1}, d)$

end for

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

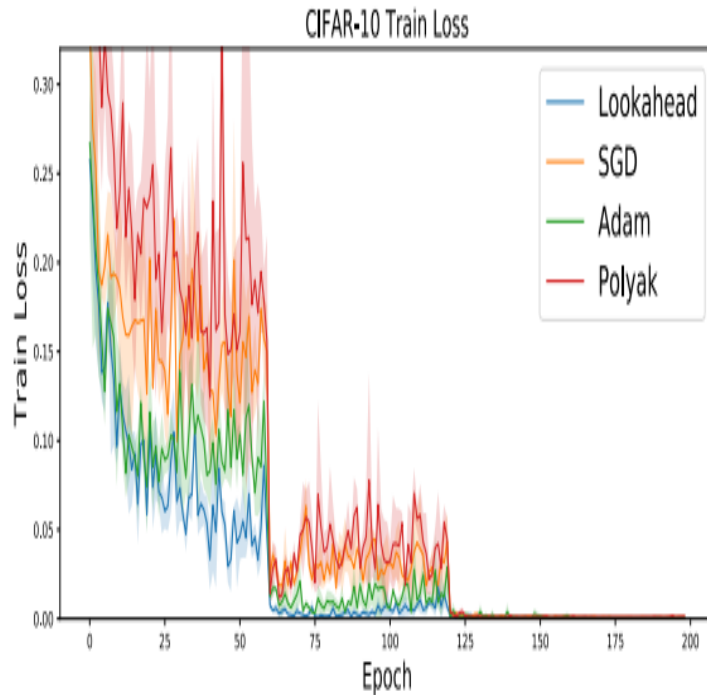
end for

return parameters ϕ



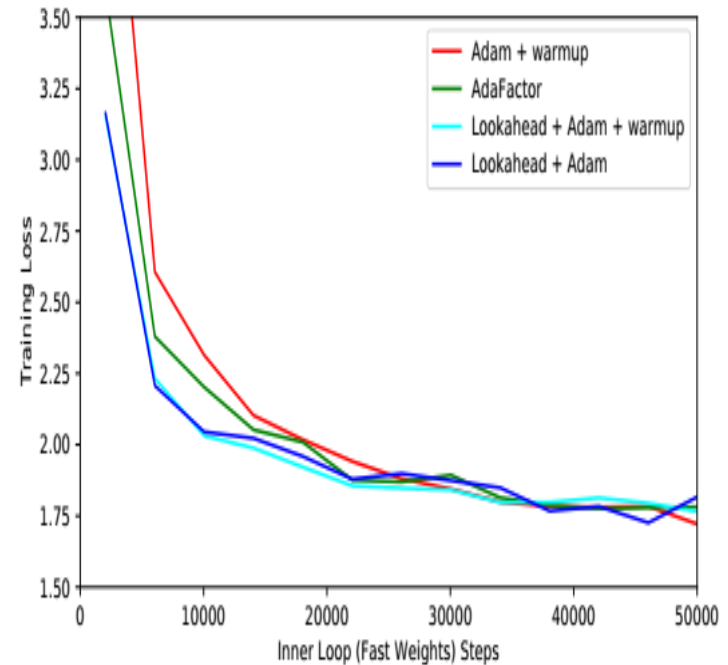
Lookahead Optimizer: k steps forward, 1 step back

ResNet-18
(CIFAR-10)



$\alpha = 0.8, k = 5$

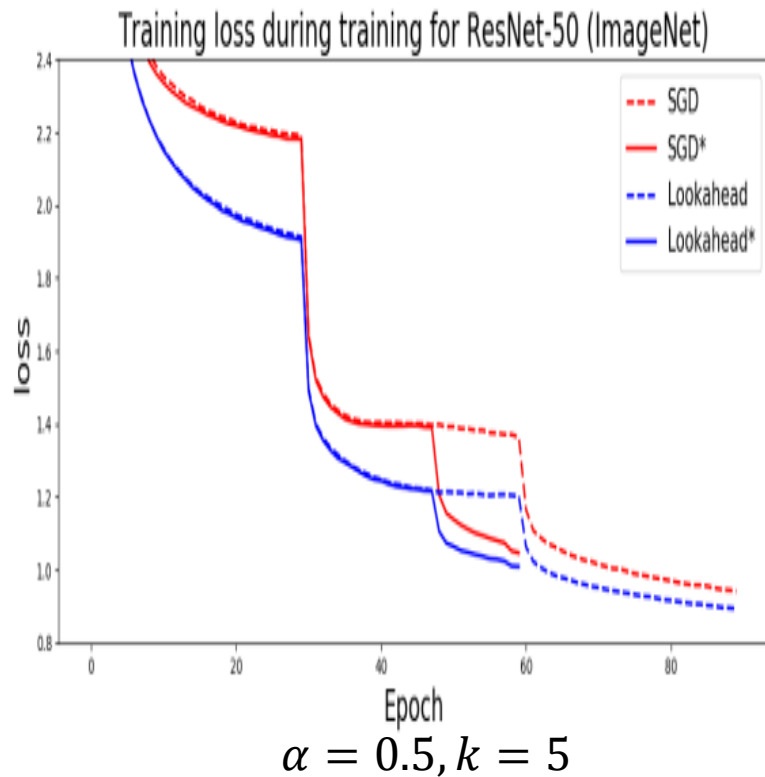
Transformer
(English to German)



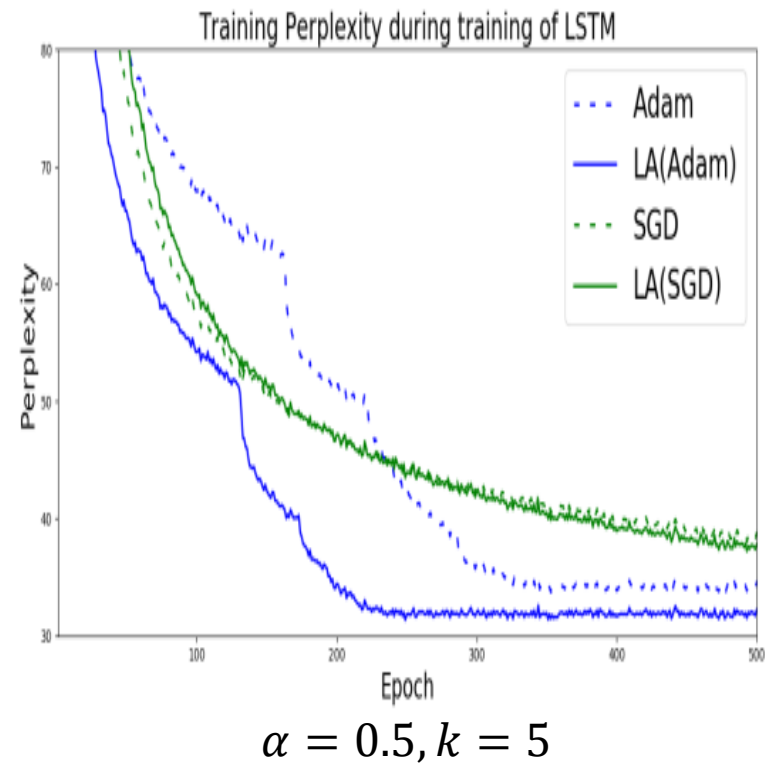
$\alpha = 0.5, k = 10$

Lookahead Optimizer: k steps forward, 1 step back

ResNet-50
(ImageNet)

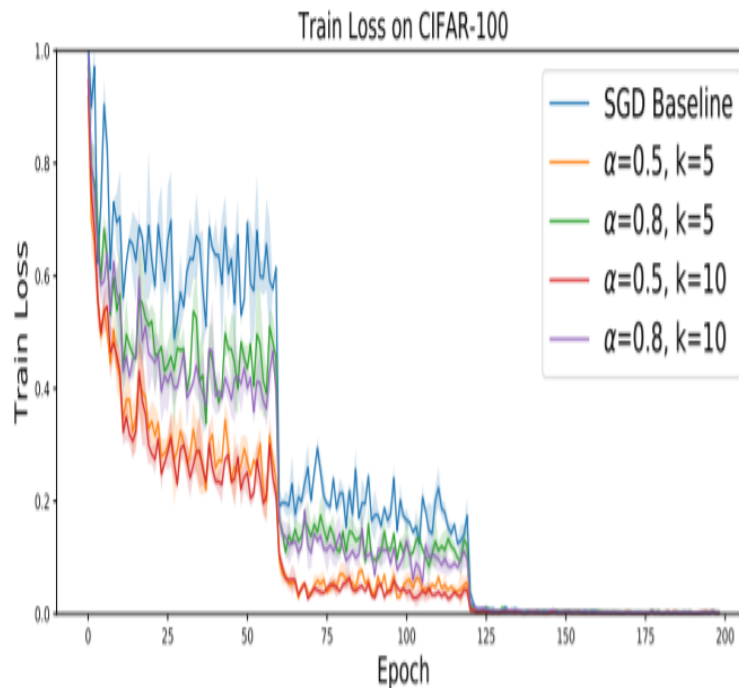


LSTM
(Penn Treebank)



Lookahead Optimizer: k steps forward, 1 step back

ResNet-18
(CIFAR-10)

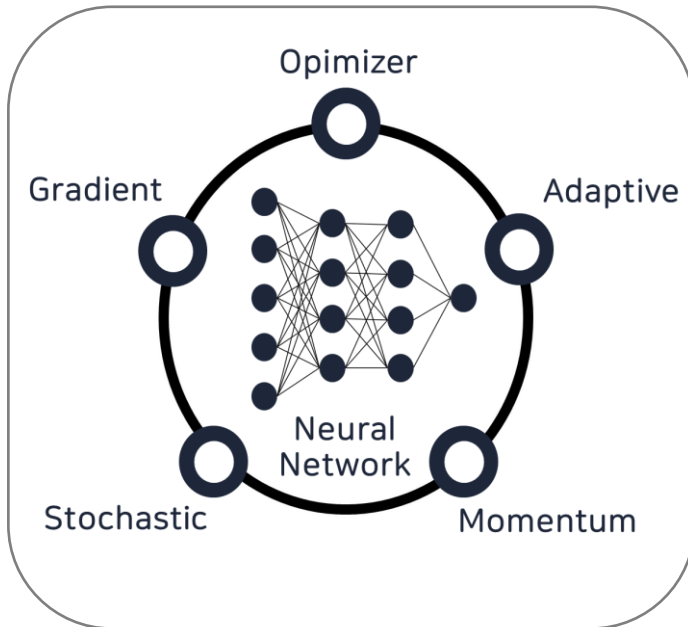


α K	0.5	0.8
5	$78.24 \pm .02$	$78.27 \pm .04$
10	$78.19 \pm .22$	$77.94 \pm .22$

- Test Accuracy
- α, K Robust

Conclusion

~ Current



Current ~

A circular diagram containing several grey and black dots, representing a partial optimization process. Below it is the word "Partial".

- Noise reduction
- Variance reduction

Second-order Optimization

$$W_{C+1} = W_C - \frac{H_L \nabla L}{\text{Hessian}}$$

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>