

# Deep Learning Basics

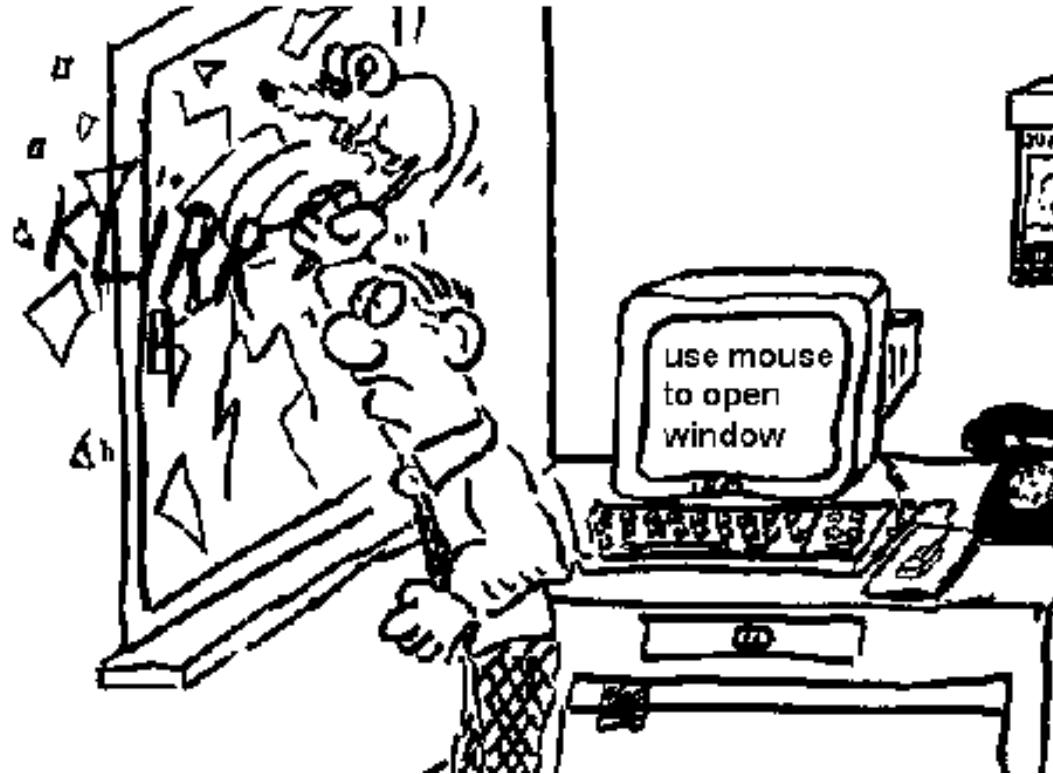
## Lecture 3: Optimization

**최성준** (고려대학교 인공지능학과)

**WARNING:** 본 교육 콘텐츠의 지식재산권은 재단법인 네이버커넥트에 귀속됩니다. **본 콘텐츠를 어떠한 경로로든 외부로 유출 및 수정하는 행위를 엄격히 금합니다.** 다만, 비영리적 교육 및 연구활동에 한정되어 사용할 수 있으나 재단의 허락을 받아야 합니다. 이를 위반하는 경우, 관련 법률에 따라 책임을 질 수 있습니다.

# Introduction

---



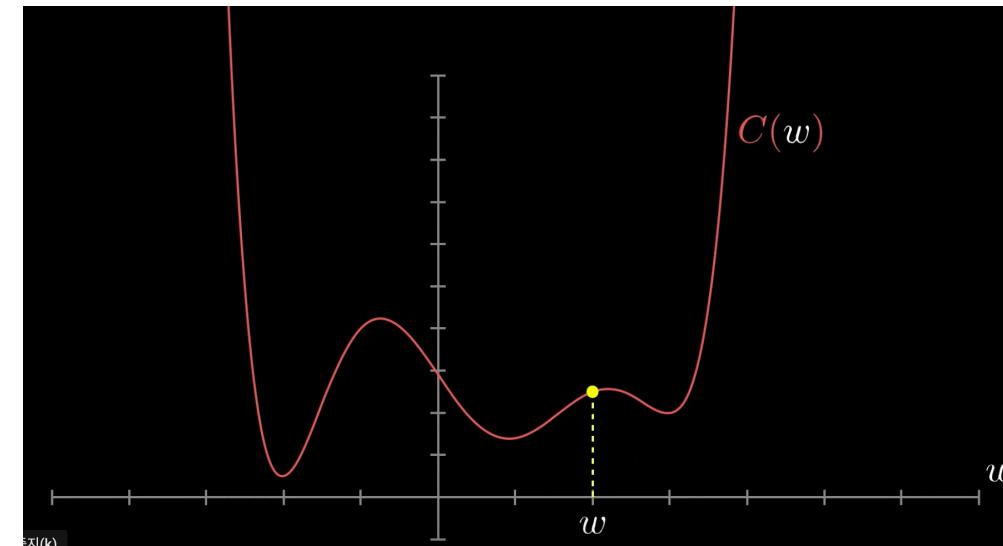
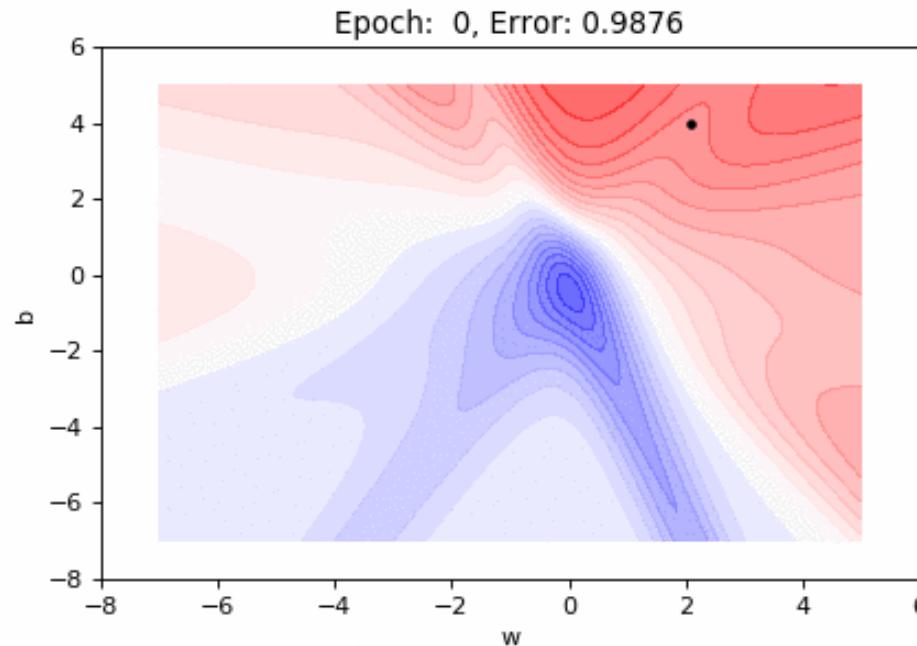
“Language is the source of misunderstandings”

Antoine de Saint-Exupéry (1900-1944)

# Introduction

---

- Gradient Descent
  - First-order iterative optimization algorithm for finding a local minimum of a differentiable function.



# Important Concepts in Optimization

# Important Concepts in Optimization

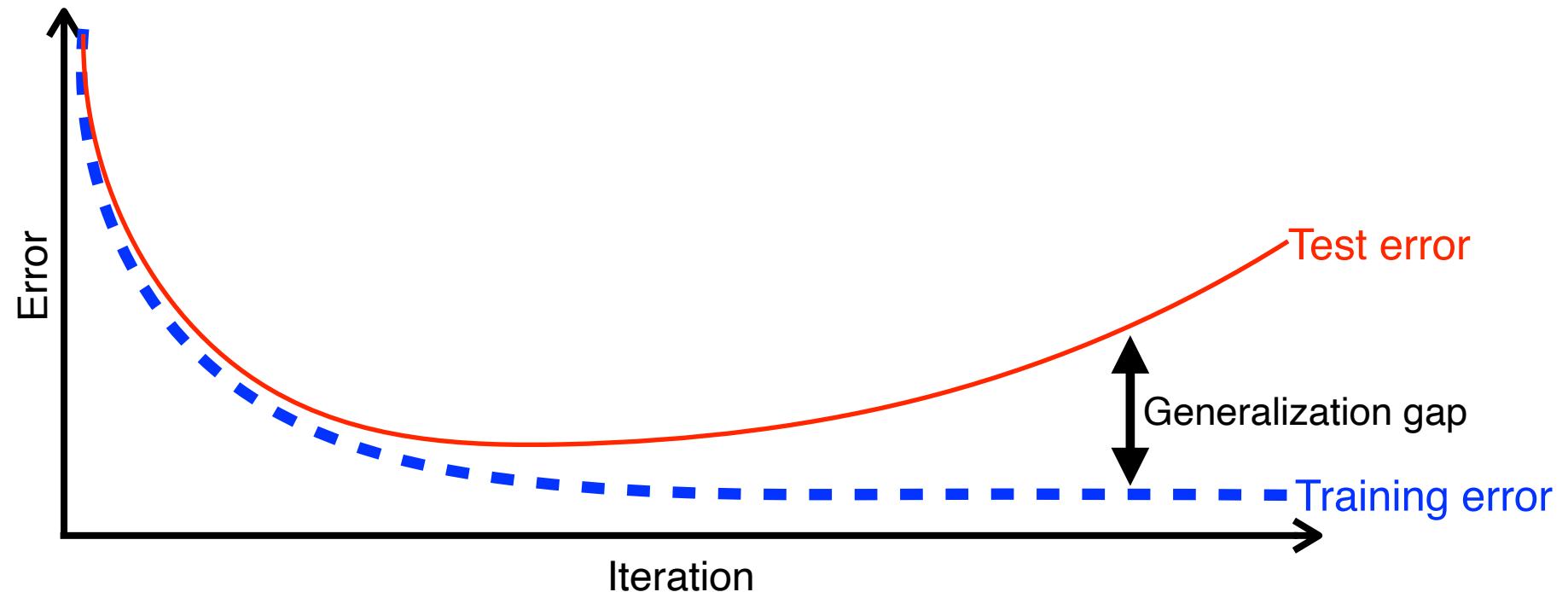
---

- Generalization
- Under-fitting vs. over-fitting
- Cross validation
- Bias-variance tradeoff
- Bootstrapping
- Bagging and boosting

# Generalization

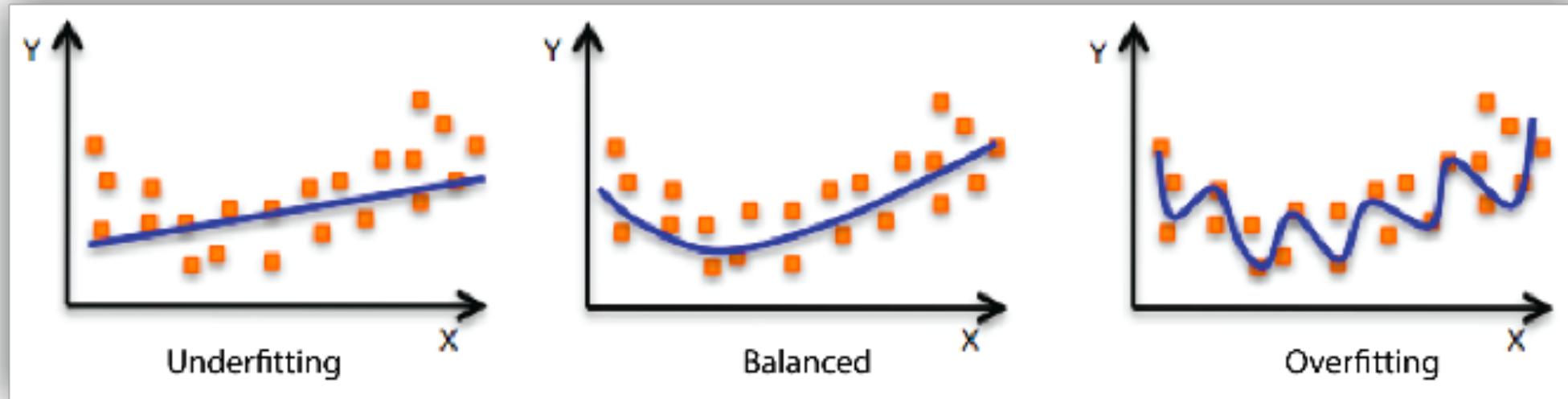
---

- How well the learned model will behave on unseen data.



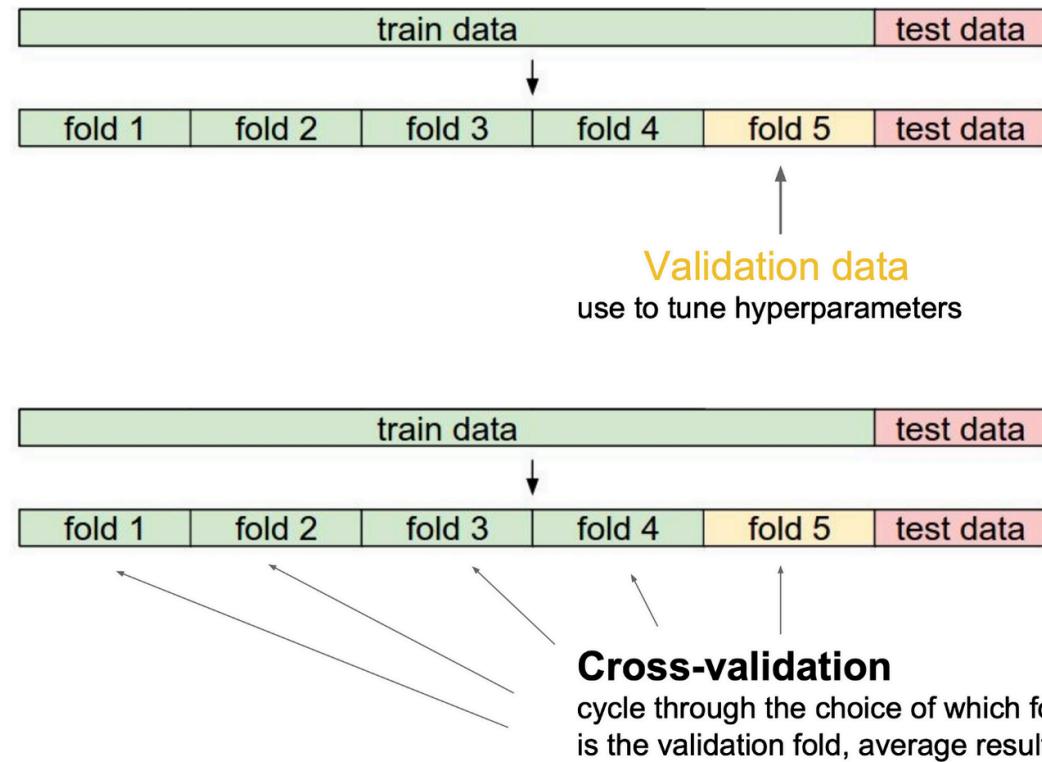
# Underfitting vs. Overfitting

---



# Cross-validation

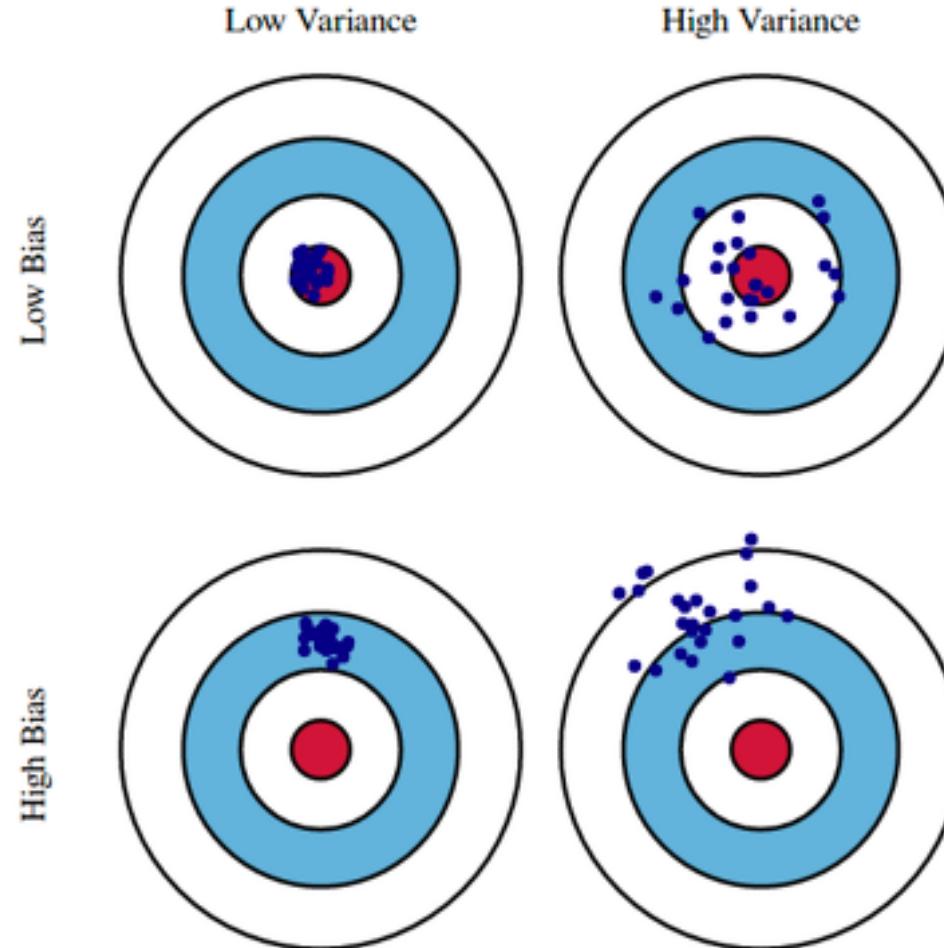
- Cross-validation is a model validation technique for assessing how the model will generalize to an independent (test) data set.



<https://blog.quantinsti.com/cross-validation-machine-learning-trading-models/>

# Bias and Variance

---



# Bias and Variance Tradeoff

---

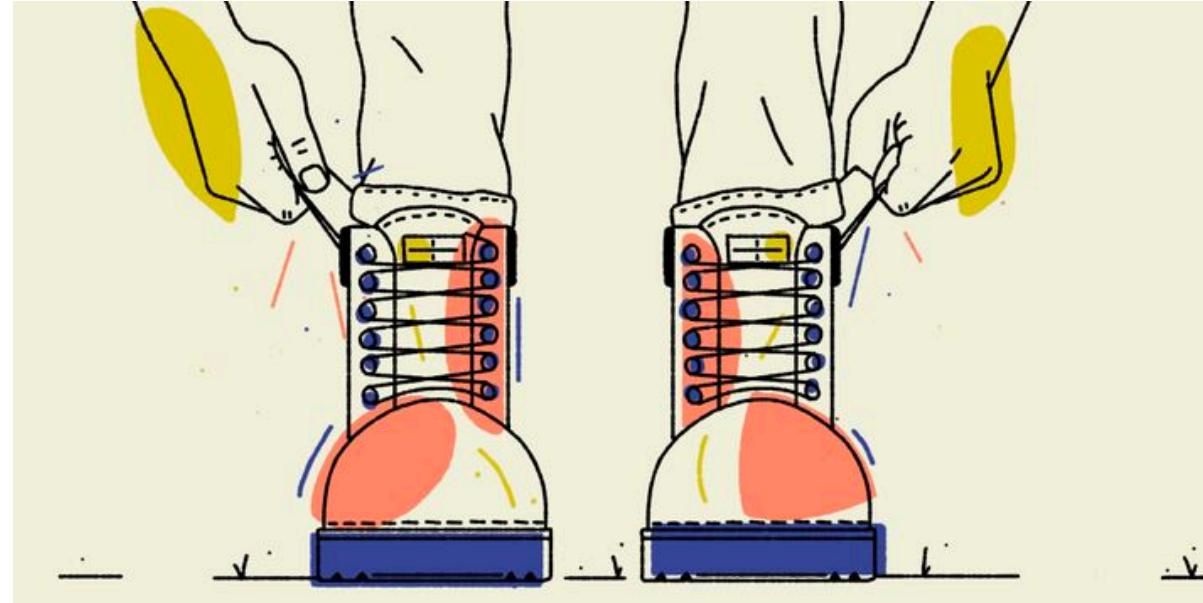
Given  $\mathcal{D} = \{(x_i, t_i)\}_{i=1}^N$ , where  $t = f(x) + \epsilon$  and  $\epsilon \sim \mathcal{N}(0, \sigma^2)$

We can derive that what we are minimizing (**cost**) can be decomposed into three different parts: **bias<sup>2</sup>**, **variance**, and **noise**.

$$\begin{aligned}\mathbb{E} [(t - \hat{f})^2] &= \mathbb{E} [(t - f + f - \hat{f})^2] \\ \text{cost} &= \dots \\ &= \mathbb{E} [(f - \mathbb{E}[\hat{f}])^2] + \mathbb{E} [(\mathbb{E}[\hat{f}] - \hat{f})^2] + \mathbb{E} [\epsilon]^2 \\ &\quad \text{bias}^2 \qquad \qquad \text{variance} \qquad \text{noise}\end{aligned}$$

# Bootstrapping

---



Pull yourself up by the **bootstraps**.

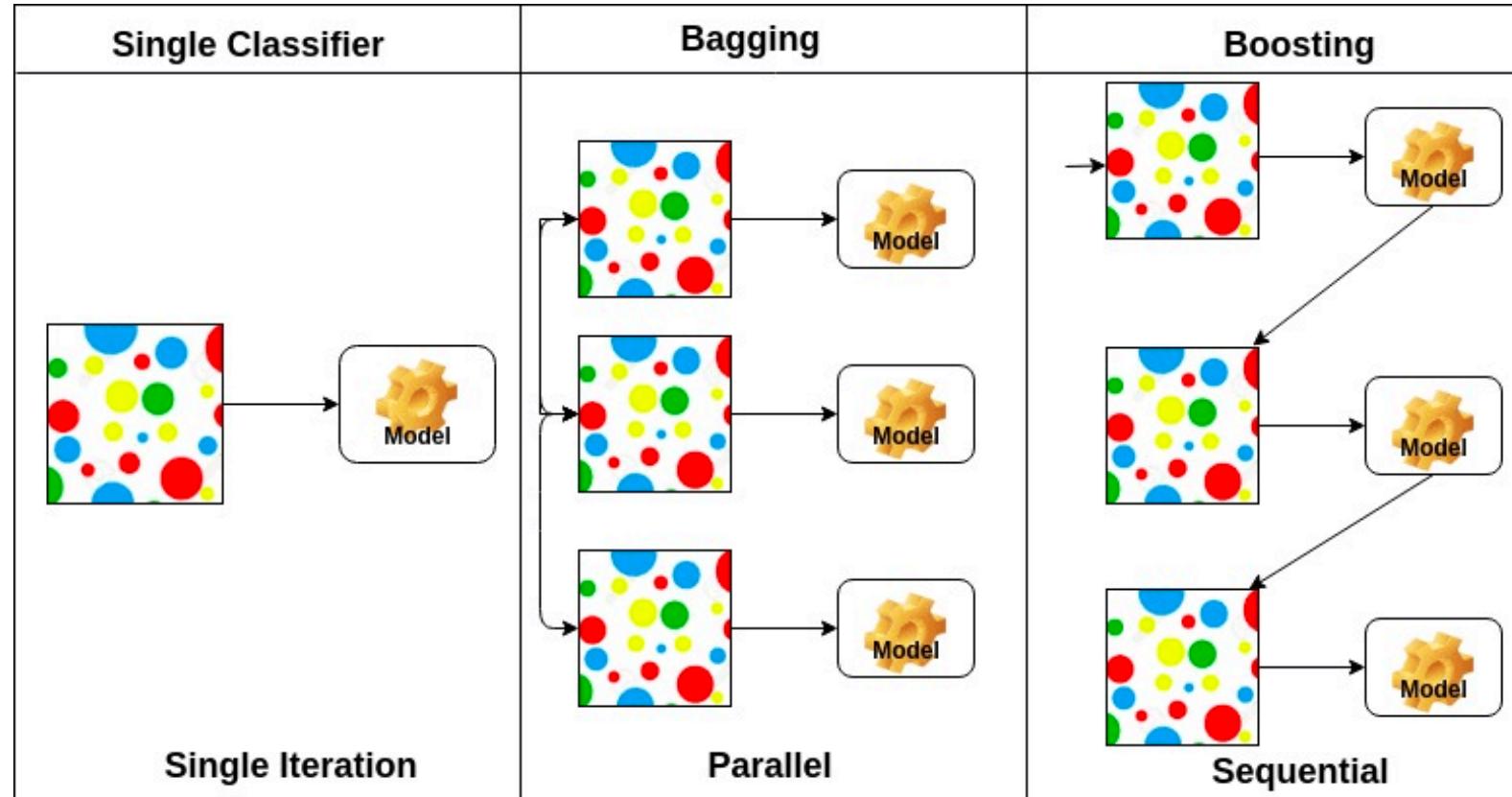
- Bootstrapping is any test or metric that uses random sampling with replacement.

# Bagging vs. Boosting

---

- Bagging (Bootstrapping aggregating)
  - Multiple models are being trained with bootstrapping.
  - ex) Base classifiers are fitted on random subset where individual predictions are aggregated (voting or averaging).
- Boosting
  - It focuses on those specific training samples that are hard to classify.
  - A strong model is built by combining weak learners in sequence where each learner learns from the mistakes of the previous weak learner.

# Bagging vs. Boosting



# Practical Gradient Descent Methods

# Gradient Descent Methods

---

- Stochastic gradient descent
  - Update with the gradient computed from a single sample.
- Mini-batch gradient descent
  - Update with the gradient computed from a subset of data.
- Batch gradient descent
  - Update with the gradient computed from the whole data.

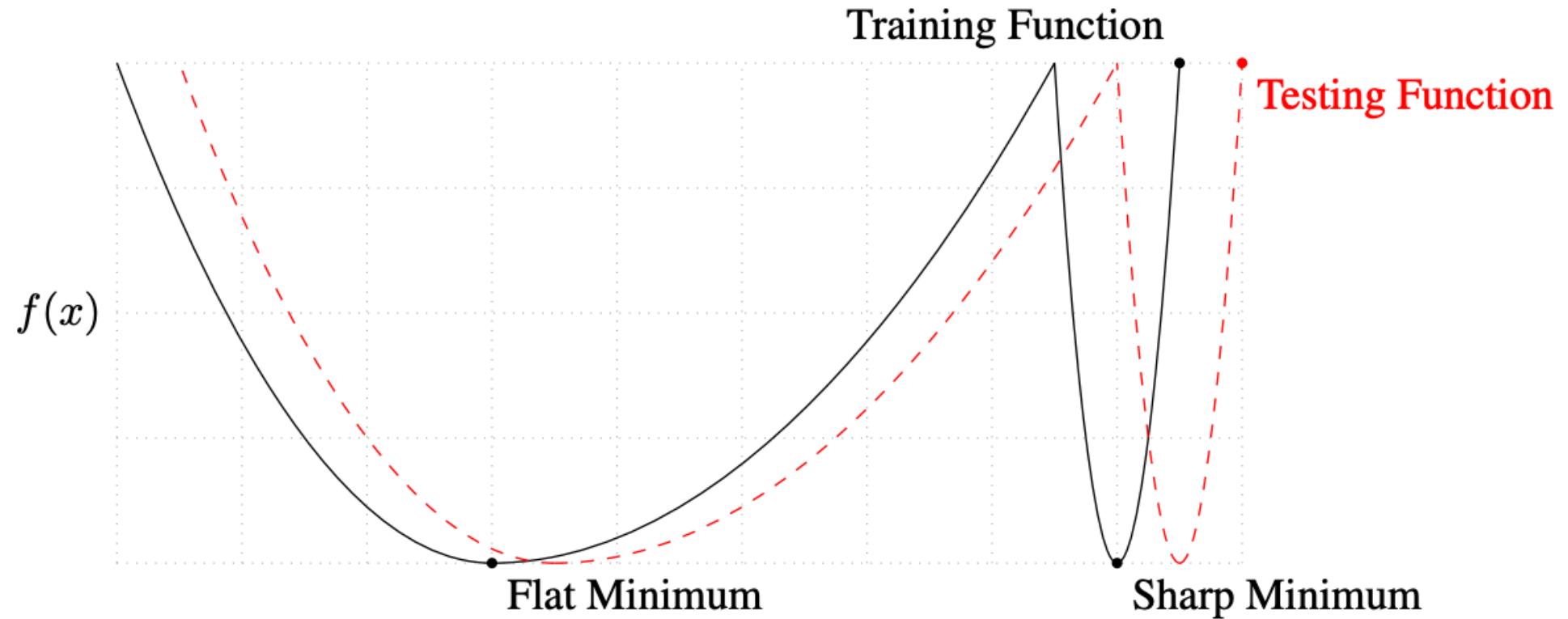
# Batch-size Matters

---

- "It has been observed in practice that when using a larger batch there is a degradation in the quality of the model, as measured by its ability to generalize."
- "We ... present numerical evidence that supports the view that large batch methods tend to converge to **sharp minimizers** of the training and testing functions. In contrast, small-batch methods consistently converge to **flat minimizers**... this is due to the inherent noise in the gradient estimation."

# Batch-size Matters

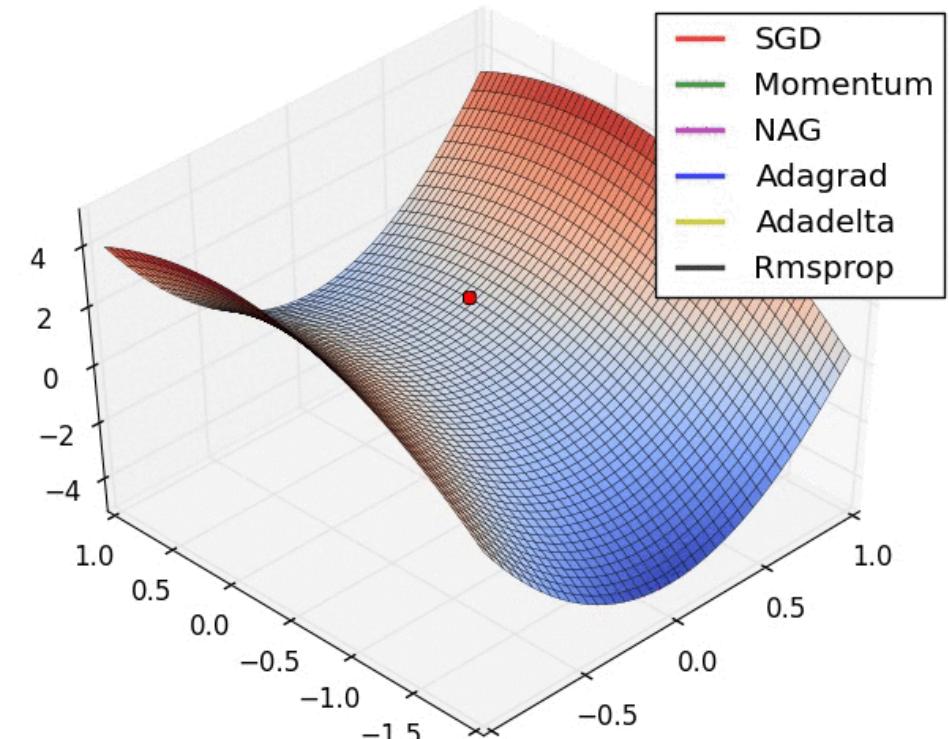
---



# Gradient Descent Methods

---

- Stochastic gradient descent
- Momentum
- Nesterov accelerated gradient
- Adagrad
- Adadelta
- RMSprop
- Adam



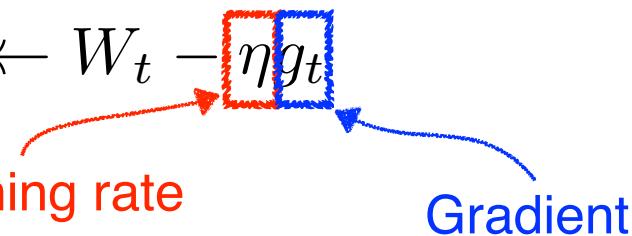
<https://imgur.com/a/Hqolp>

# Gradient Descent

---

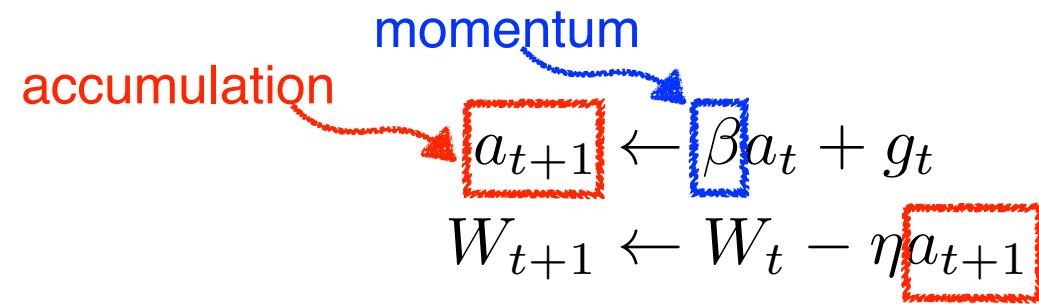
$$W_{t+1} \leftarrow W_t - \eta g_t$$

Learning rate      Gradient



# Momentum

---



# Nesterov Accelerated Gradient

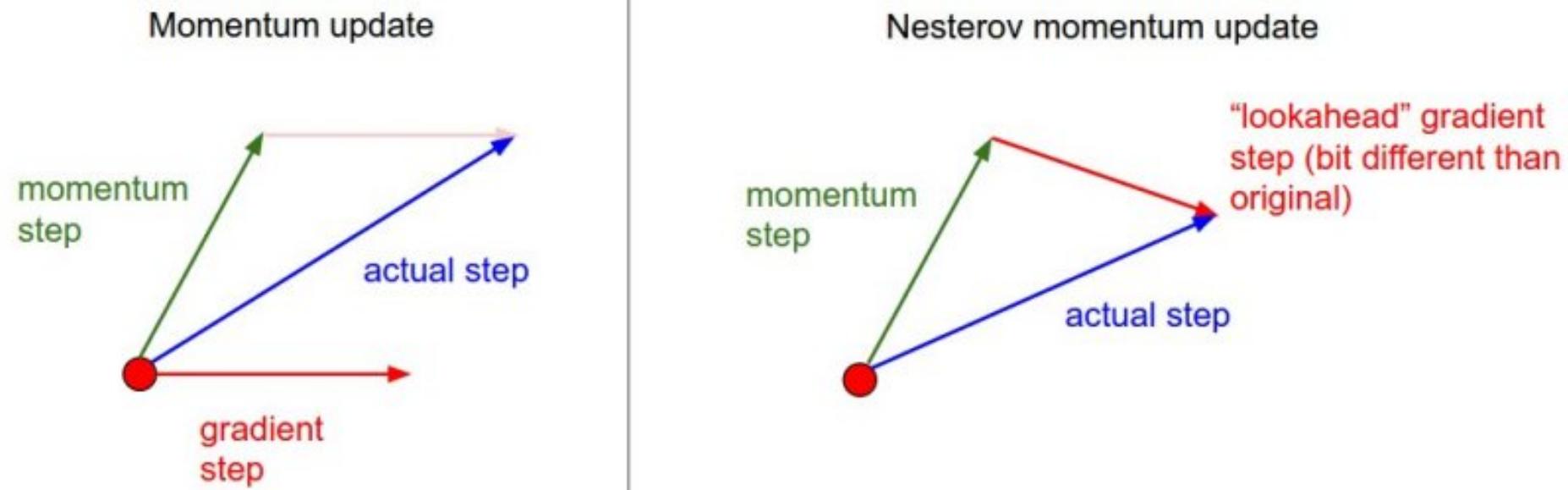
---

Lookahead gradient

$$a_{t+1} \leftarrow \beta a_t + \boxed{\nabla \mathcal{L}(W_t - \eta \beta a_t)}$$
$$W_{t+1} \leftarrow W_t - \eta a_{t+1}$$

# Nesterov Accelerated Gradient

---



# Adagrad

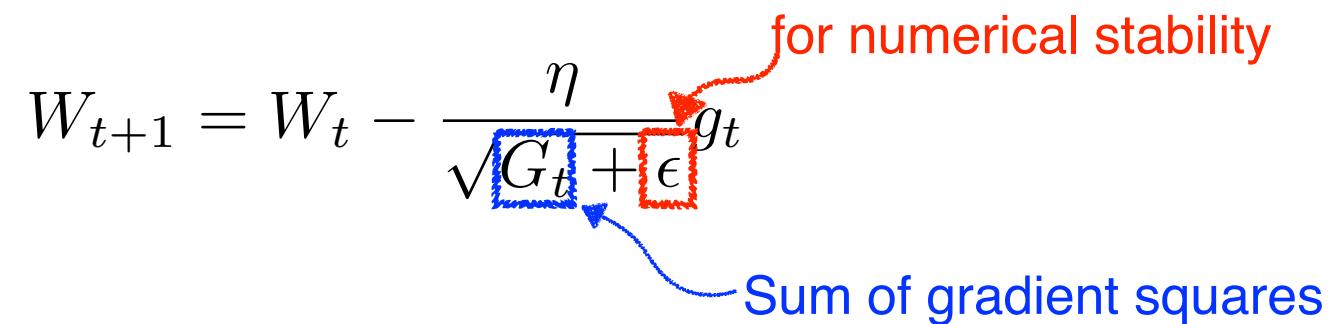
---

- **Adagrad** adapts the learning rate, performing larger updates for infrequent and smaller updates for frequent parameters.

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{G_t} + \epsilon} g_t$$

for numerical stability

Sum of gradient squares



What will happen if the training occurs for a long period?

# Adadelta

---

- **Adadelta** extends **Adagrad** to reduce its monotonically decreasing the learning rate by restricting the accumulation window.

EMA of gradient squares

$$G_t = \gamma G_{t-1} + (1 - \gamma) g_t^2$$

EMA of difference squares

$$W_{t+1} = W_t - \frac{\sqrt{H_{t-1} + \epsilon}}{\sqrt{G_t + \epsilon}} g_t$$

$$H_t = \gamma H_{t-1} + (1 - \gamma)(\Delta W_t)^2$$

There is **no learning rate** in Adadelta.

# RMSprop

- **RMSprop** is an unpublished, adaptive learning rate method proposed by Geoff Hinton in his lecture.

EMA of gradient squares

$$G_t = \gamma G_{t-1} + (1 - \gamma) g_t^2$$
$$W_{t+1} = W_t - \frac{\eta}{\sqrt{G_t + \epsilon}} g_t$$


# Adam

---

- Adaptive Moment Estimation (Adam) leverages both past gradients and squared gradients.

The diagram illustrates the Adam optimization algorithm. It starts with two equations:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

These are labeled "Momentum" and "EMA of gradient squares" respectively. Arrows point from these labels to the corresponding terms in the equations. Below these, the final weight update equation is shown:

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{v_t + \epsilon}} \frac{\sqrt{1 - \beta_2^t}}{1 - \beta_1^t} m_t$$

An arrow points from the "Stepsize" label to the learning rate term  $\eta$ . Another arrow points from the "Stepsize" label to the momentum term  $m_t$ .

Adam effectively combines momentum with adaptive learning rate approach.

# Regularization

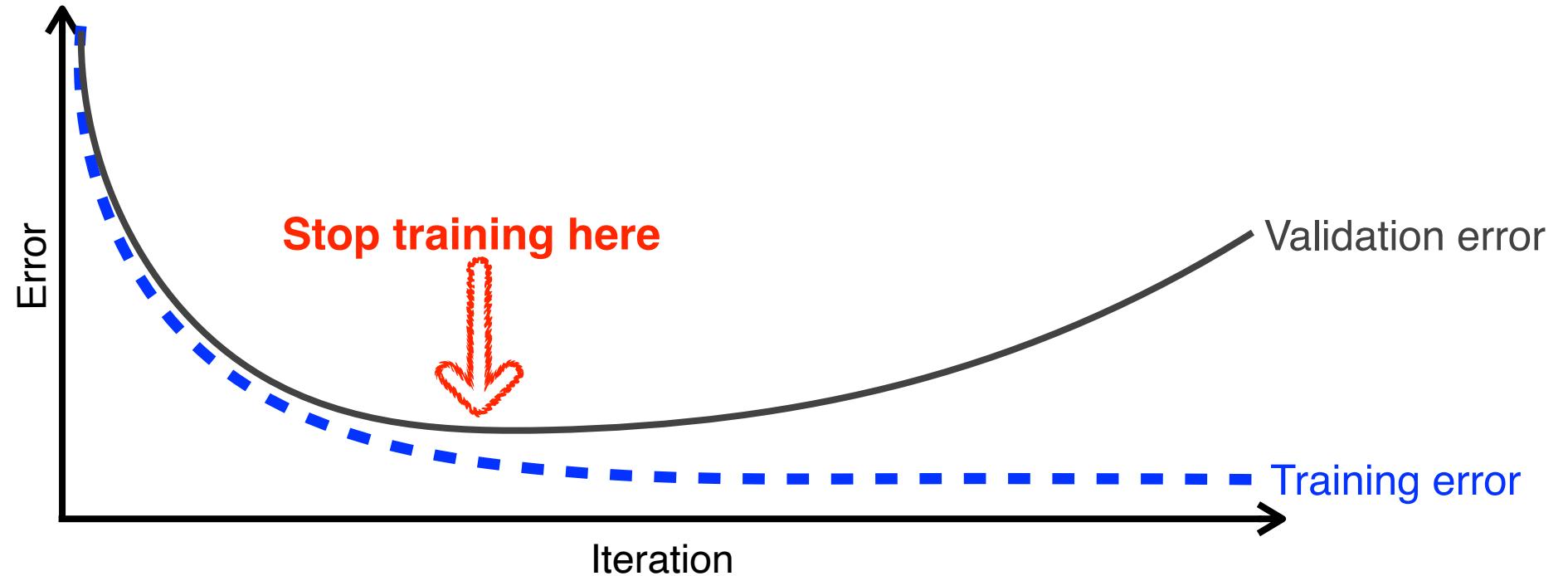
# Regularization

---

- ➊ Early stopping
- ➋ Parameter norm penalty
- ➌ Data augmentation
- ➍ Noise robustness
- ➎ Label smoothing
- ➏ Dropout
- ➐ Batch normalization

# Early Stopping

---



Note that we need **additional validation data** to do early stopping.

# Parameter Norm Penalty

---

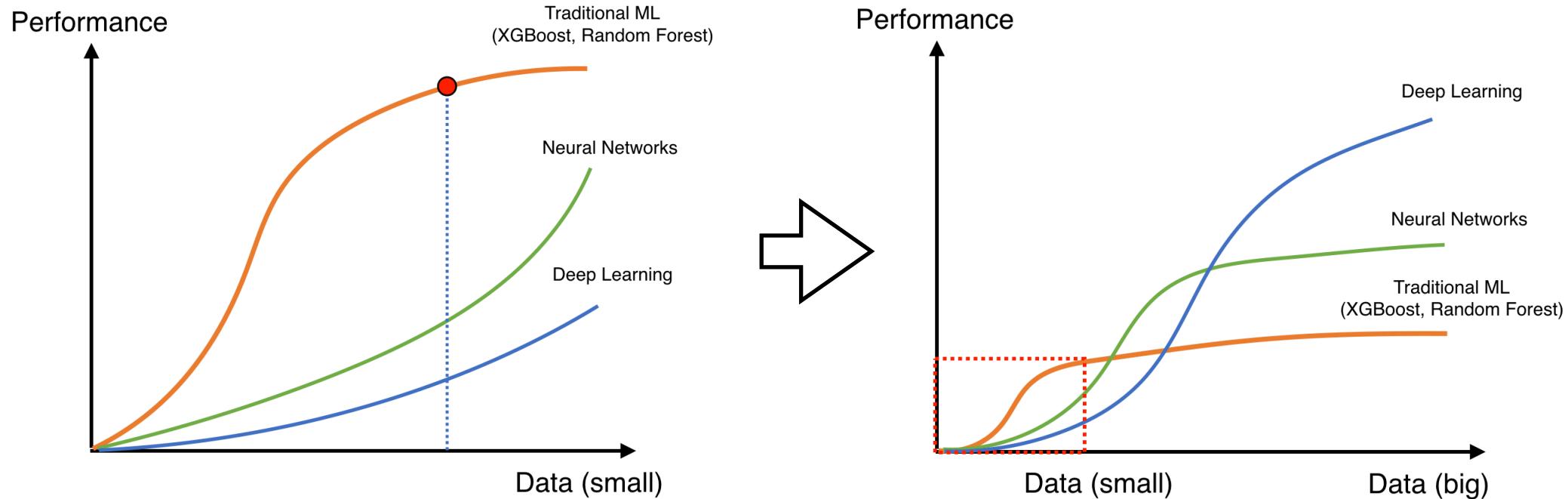
- It adds smoothness to the function space.

Parameter Norm Penalty

$$\text{total cost} = \text{loss}(\mathcal{D}; W) + \frac{\alpha}{2} \|W\|_2^2$$

# Data Augmentation

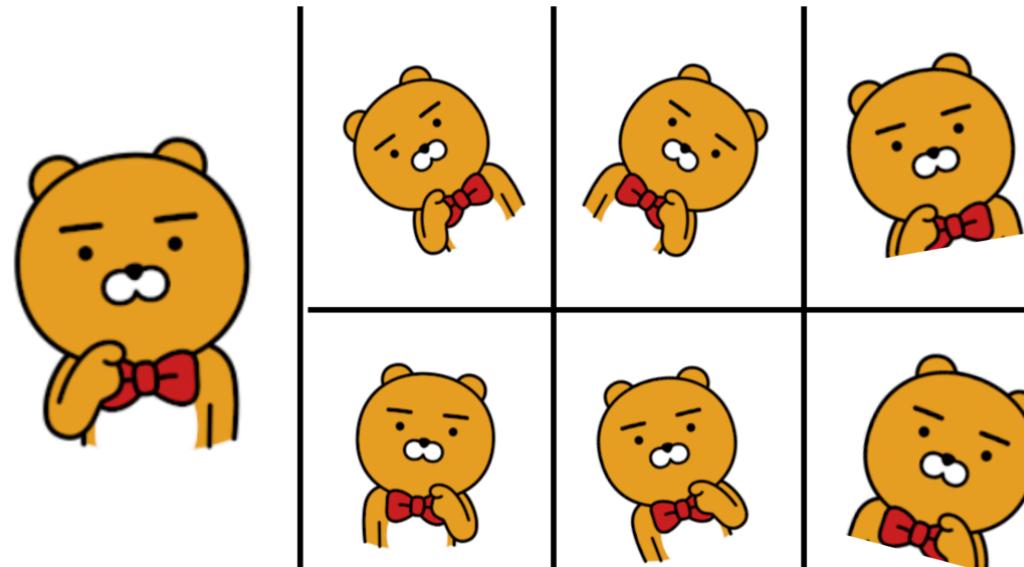
- More data are always welcomed.



# Data Augmentation

---

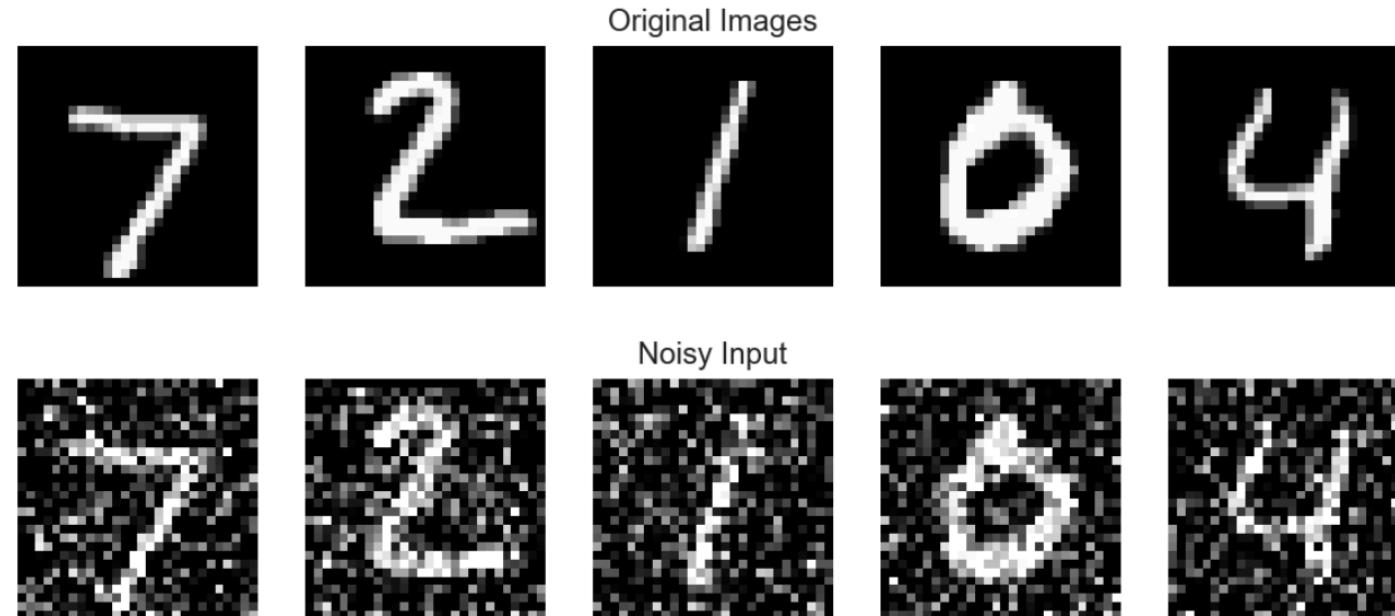
- More data are always welcomed.
- However, in most cases, training data are given in advance.
- In such cases, we need data augmentation.



# Noise Robustness

---

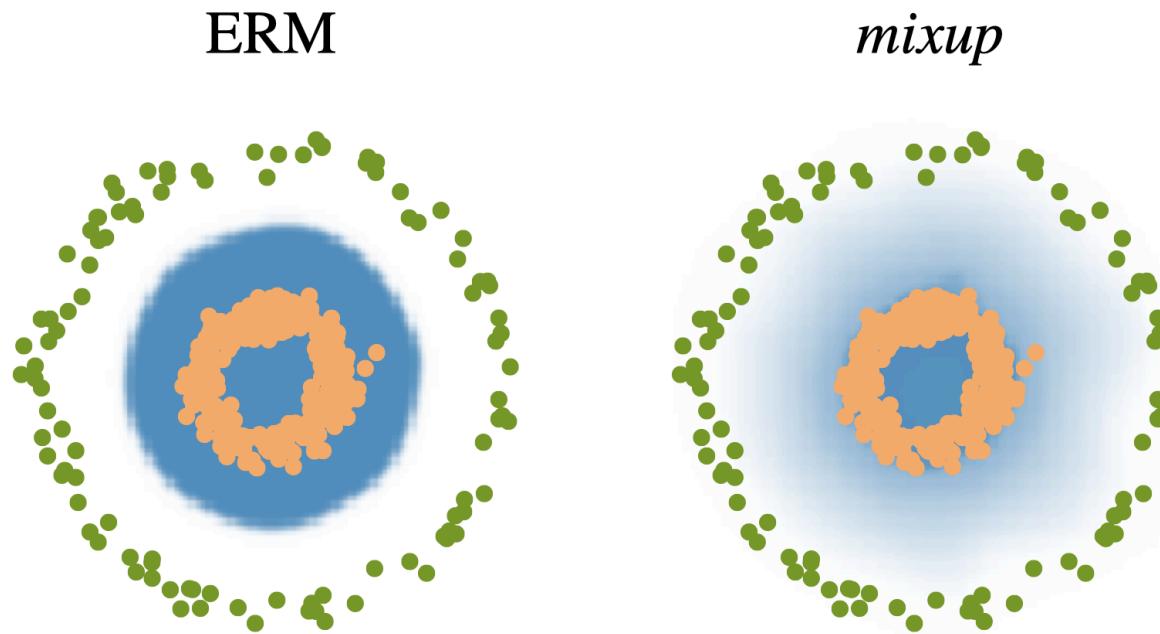
- Add random noises inputs or weights.



# Label Smoothing

---

- **Mix-up** constructs augmented training examples by mixing both input and output of two randomly selected training data.



mixup: Beyond Empirical Risk Minimization, 2018

# Label Smoothing

- **CutMix** constructs augmented training examples by mixing inputs with cut and paste and outputs with soft labels of two randomly selected training data.

	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4

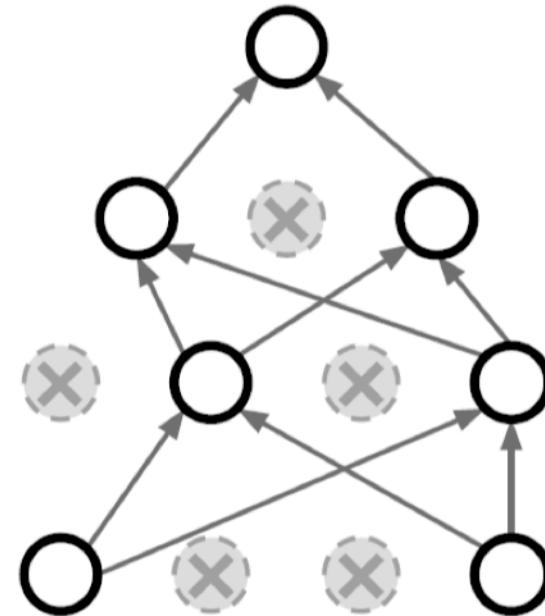
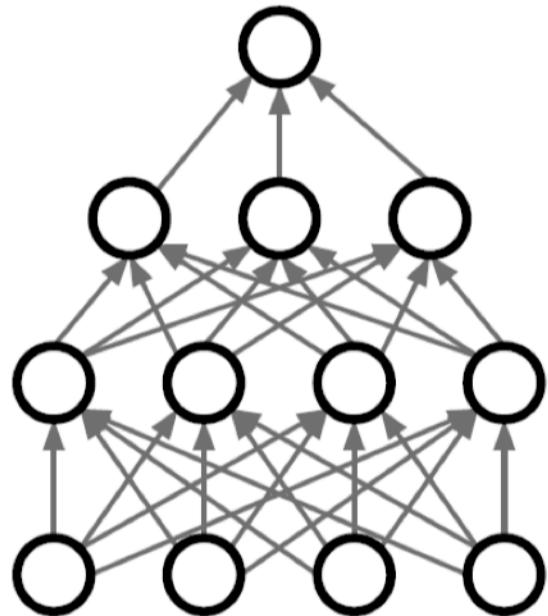
CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features, 2019

© NAVER Connect Foundation

# Dropout

---

- In each forward pass, randomly set some neurons to zero.



# Batch Normalization

---

- Batch normalization compute the empirical mean and variance independently for each dimension (layers) and normalize.

$$\mu_B = \frac{1}{m} \sum_{i=1}^m x_i$$

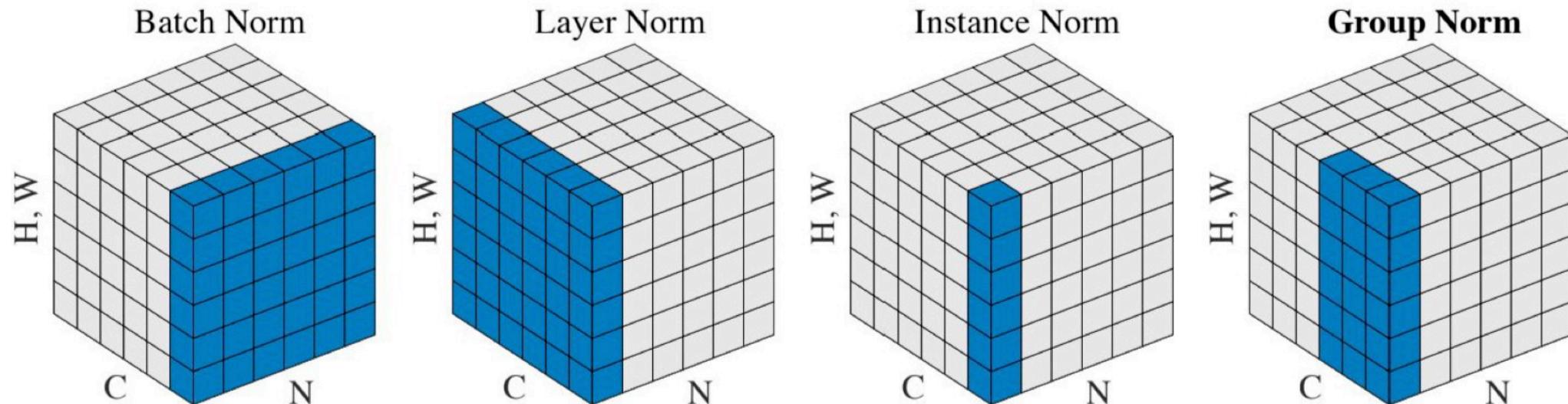
$$\sigma_B^2 = \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2$$

$$\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$$

# Batch Normalization

---

- There are different variances of normalizations.



# Thank you for listening

---