

# Deep Learning Seminar

Chapter 11 Practical Methodology

Chapter 12 Applications

Heechul Lim

Department of Information and Communication Engineering

DGIST

2017.11.29



# Contents

## ● Chapter 11 Practical Methodology

11.1 Performance Metrics

11.2 Default Baseline Models

11.3 Determining Whether to Gather More Data

11.4 Selecting Hyperparameters

11.5 Debugging Strategies

11.6 Example: Multi-Digit Number Recognition

## ● Chapter 12 Applications

12.1 LargeScale DeepLearning

12.2 Computer Vision

12.3 Speech Recognition

12.4 Natural Language Processing

12.5 Other Applications

# Contents

## ● Chapter 11 Practical Methodology

11.1 Performance Metrics

11.2 Default Baseline Models

11.3 Determining Whether to Gather More Data

11.4 Selecting Hyperparameters

11.5 Debugging Strategies

11.6 Example: Multi-Digit Number Recognition

## ● Chapter 12 Applications

12.1 LargeScale DeepLearning

12.2 Computer Vision

12.3 Speech Recognition

12.4 Natural Language Processing

12.5 Other Applications

# Contents

## ● Chapter 11 Practical Methodology

### 11.1 Performance Metrics

### 11.2 Default Baseline Models

### 11.3 Determining Whether to Gather More Data

### 11.4 Selecting Hyperparameters

### 11.5 Debugging Strategies

### 11.6 Example: Multi-Digit Number Recognition

## ● Chapter 12 Applications

### 12.1 LargeScale DeepLearning

### 12.2 Computer Vision

### 12.3 Speech Recognition

### 12.4 Natural Language Processing

### 12.5 Other Applications

# Introduction

- Applying deep learning techniques **require more**
- A good practitioner needs to know how to **monitor it**
- Practitioners need to
  - Decide whether to **gather more data**
  - Increase or decrease **model capacity**
  - Add or remove **regularizing features**
  - Improve the **optimization of a model**
  - Improve **approximate inference in a model**
  - **Debug** the software implementation of the model
- It is important to be able to **determine the right course**

# Introduction

- [Andrew Ng] Advice for applying Machine Learning
  - Determine your goals
  - Establish a working end-to-end pipeline as soon as possible
  - Diagnose which components are performing well
  - Repeatedly make incremental changes

# Performance Metrics

- Determine your **goals**
  - What **metric** to use
  - What **level** of performance you desire
- Guide all of your **future** actions
  - With **error** metric
- Keep in mind that **no app** achieve **zero error**
- Keep in mind that data can be **limited**

# Performance Metrics

- In the academic setting
  - **Attainable error rate** based on published benchmark results
- In the real-world setting
  - **Safe error rate**
  - **Cost-effective**
  - **Appealing to consumers**
- One kind of a mistake > another
- E-mail spam detection system
  - **incorrectly classifying a legitimate message as spam**
  - **incorrectly allowing a spam message to appear in the inbox**



# Performance Metrics

- Training a binary classifier to detect some rare event
- Medical test for a **rare disease**
- 1%
  - Probability of having disease
- 99%
  - Accuracy by simply hard-coding
- **Is that excellent?**

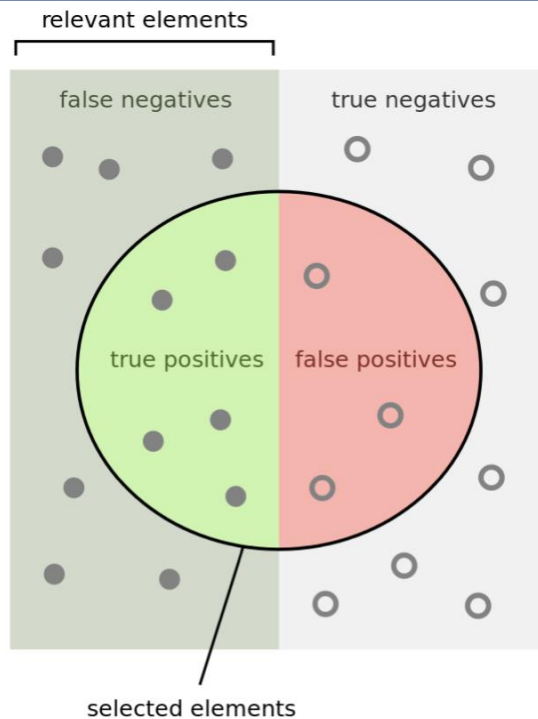
# Precision and recall

## ● Precision

- The detections reported by the model that were correct

## ● Recall

- The true events that were detected



$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

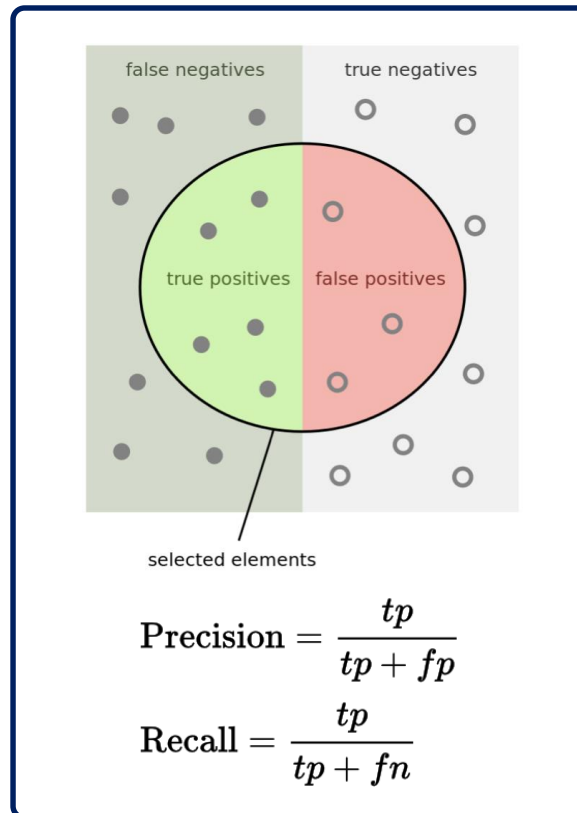
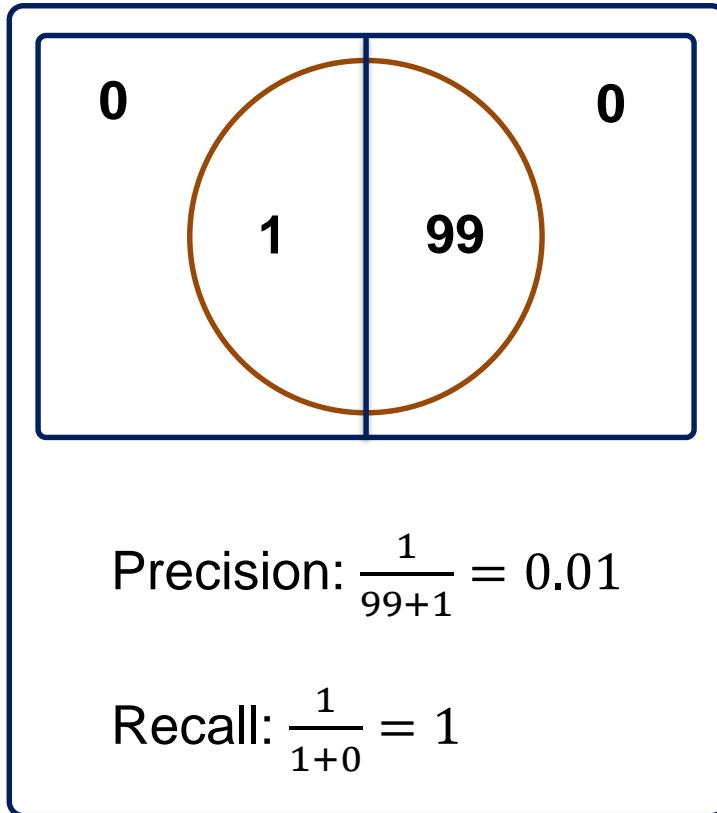
A Venn diagram illustrating Precision. It consists of two overlapping circles. The left circle is green and represents 'true positives'. The right circle is red and represents 'false positives'. The intersection of the two circles is shaded green, representing the ratio of true positives to the total number of detected elements (true positives plus false positives).

A Venn diagram illustrating Recall. It consists of two overlapping circles. The left circle is green and represents 'true positives'. The right circle is light gray and represents 'false negatives'. The intersection of the two circles is shaded green, representing the ratio of true positives to the total number of relevant elements (true positives plus false negatives).

[https://en.wikipedia.org/wiki/Precision\\_and\\_recall](https://en.wikipedia.org/wiki/Precision_and_recall)

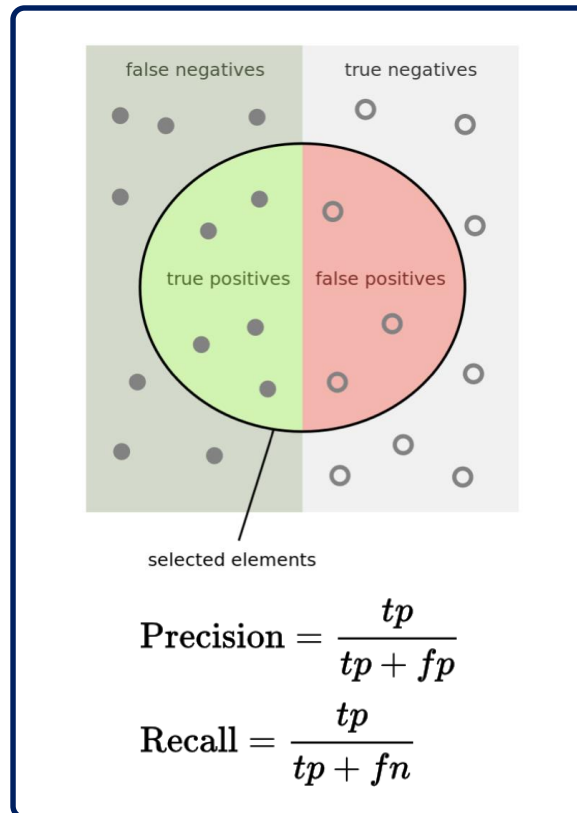
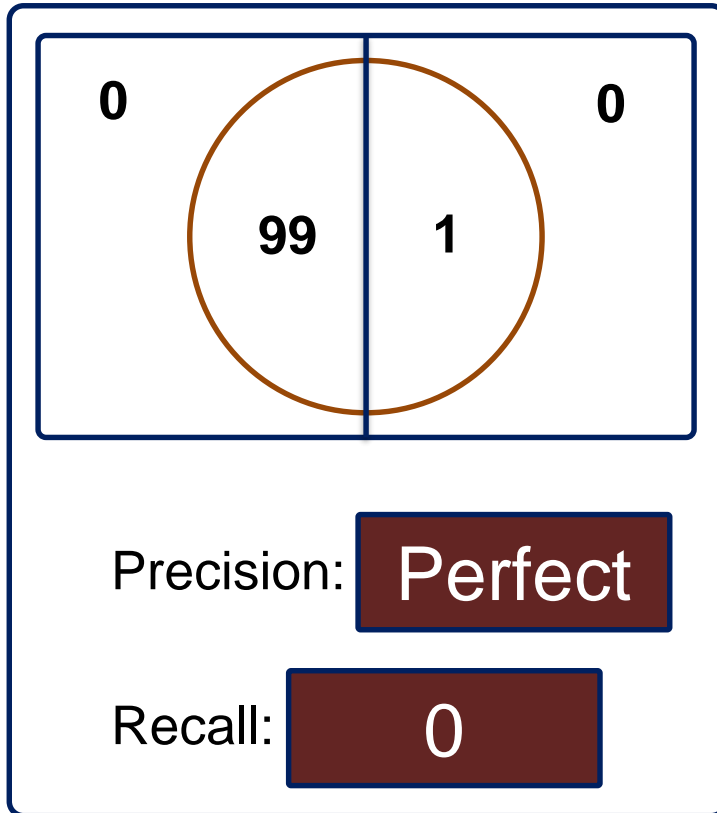
# Detecting disease

- A detector that says everyone has the disease



# Detecting disease

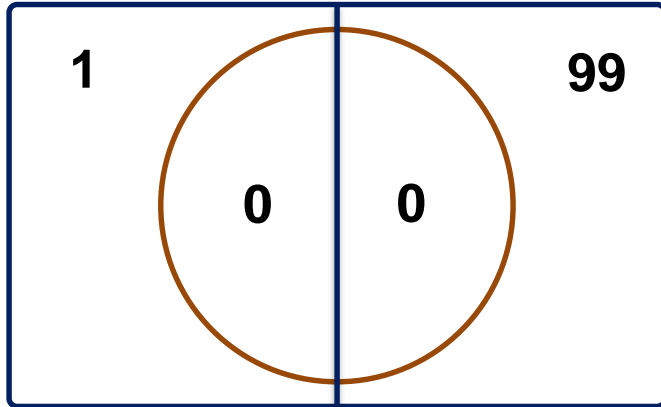
- A detector that says no one has the disease



**Wrong!**

# Detecting disease

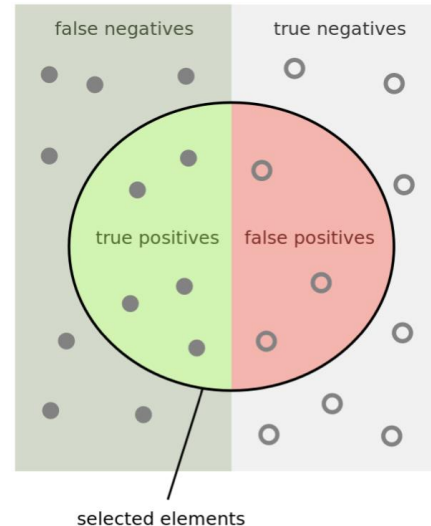
- A detector that says no one has the disease



$$\text{Precision: } \frac{0}{0+0}$$

$$\text{Recall: } \frac{0}{1+0} = 0$$

$$\text{Accuracy: } \frac{99}{1+0+0+99} = 0.99$$



$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

Correct!

# Precision and recall(PR)

- Summarizing PR with a single number

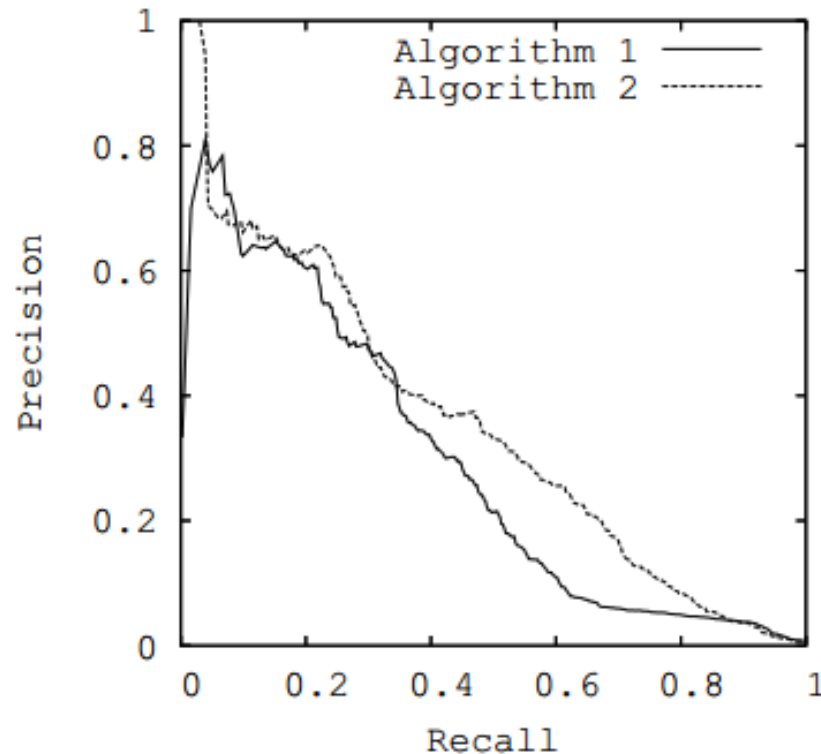
- F-score**

- $F = \frac{2pr}{p+r}$
- Harmonic mean

$$\bar{x} = n \times \left( \sum_{i=1}^n \frac{1}{x_i} \right)^{-1}$$

# Precision and recall

- Summarizing PR with graph
- PR curve



<https://www.quora.com/What-is-Precision-Recall-PR-curve>

# Performance Metrics

- Refusing to make a decision
  - Reducing the amount of job that the human must process
- Coverage
  - A response range of system
- Coverage V.S. Accuracy



# Contents

## ● Chapter 11 Practical Methodology

### 11.1 Performance Metrics

### **11.2 Default Baseline Models**

### 11.3 Determining Whether to Gather More Data

### 11.4 Selecting Hyperparameters

### 11.5 Debugging Strategies

### 11.6 Example: Multi-Digit Number Recognition

## ● Chapter 12 Applications

### 12.1 LargeScale DeepLearning

### 12.2 Computer Vision

### 12.3 Speech Recognition

### 12.4 Natural Language Processing

### 12.5 Other Applications

# Default Baseline Models

- **Recommendations** for which **algorithms** to use
- Depending on the **complexity** of your problem
  - Shallow learning model
  - Deep learning model
- Depending on **structure** of your data
  - Fully connected layer networks
  - Convolutional neural networks
  - Recurrent neural networks

# Default Baseline Models - Optimization

- **SGD with a decaying learning rate**
  - Decaying **linearly** until reaching minimum learning rate
  - Decaying **exponentially**
  - **Decreasing** the learning rate by a factor of 2-10 each time
  - Momentum, Nesterov, Adagrad, RMSprop, Adam
- **Batch normalization (BN)**
  - **Dramatic effect** on optimization performance
  - **Especially for convolutional networks and networks with sigmoidal nonlinearities**

# Default Baseline Models - Optimization

- BN V.S. Whitening
- Definition of BN

**Input:** Values of  $x$  over a mini-batch:  $\mathcal{B} = \{x_1 \dots x_m\}$ ;

Parameters to be learned:  $\gamma, \beta$

**Output:**  $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{ mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{ mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{ normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{ scale and shift}$$

<https://kratzert.github.io/2016/02/12/understanding-the-gradient-flow-through-the-batch-normalization-layer.html>

# Default Baseline Models

- If your training set is limited
  - You should include **regularization** from the start
  - **Early stopping** should be used almost universally
  - **Dropout** is an excellent regularizer
  - BN sometimes **reduces generalization error**

# Contents

## ● Chapter 11 Practical Methodology

11.1 Performance Metrics

11.2 Default Baseline Models

**11.3 Determining Whether to Gather More Data**

11.4 Selecting Hyperparameters

11.5 Debugging Strategies

11.6 Example: Multi-Digit Number Recognition

## ● Chapter 12 Applications

12.1 LargeScale DeepLearning

12.2 Computer Vision

12.3 Speech Recognition

12.4 Natural Language Processing

12.5 Other Applications

# Determining Whether to Gather More Data

- How does one decide whether to gather more data?
  - Determine whether the performance on the training set is acceptable.
- If performance on the training set is poor
  - Try increasing the size of the model
  - Try improving the learning algorithm
- If large models and algorithms do not work well
  - The problem might be the of the training data.
  - Try collecting cleaner data or collecting a richer set of features.

# Determining Whether to Gather More Data

- If the **performance** on the **test set** is **acceptable**
  - There is nothing left to be done
- If test set **performance** is **worse** than training
  - Gathering more data is one of the most effective solutions
- The **key** considerations
  - The **cost** and **feasibility** of **gathering more data**
  - The **cost** and **feasibility** of **reducing test error** by **other means**
  - The **amount of data** that is **expected to be necessary**



# Contents

## ● Chapter 11 Practical Methodology

11.1 Performance Metrics

11.2 Default Baseline Models

11.3 Determining Whether to Gather More Data

**11.4 Selecting Hyperparameters**

11.5 Debugging Strategies

11.6 Example: Multi-Digit Number Recognition

## ● Chapter 12 Applications

12.1 LargeScale DeepLearning

12.2 Computer Vision

12.3 Speech Recognition

12.4 Natural Language Processing

12.5 Other Applications

# Selecting Hyperparameters

## ● Characters of hyperparameter

- Affecting the time and memory cost of running the algorithm
- Affecting the quality of the model recovered by the training
- Affecting its ability to infer correct results

## ● Various hyperparameters

- Convolution kernel width
- Implicit zero padding
- Dropout rate
- Learning rate
- Weight decay coefficient(Regularization parameter)
- Number of hidden units

# Selecting Hyperparameters - manual

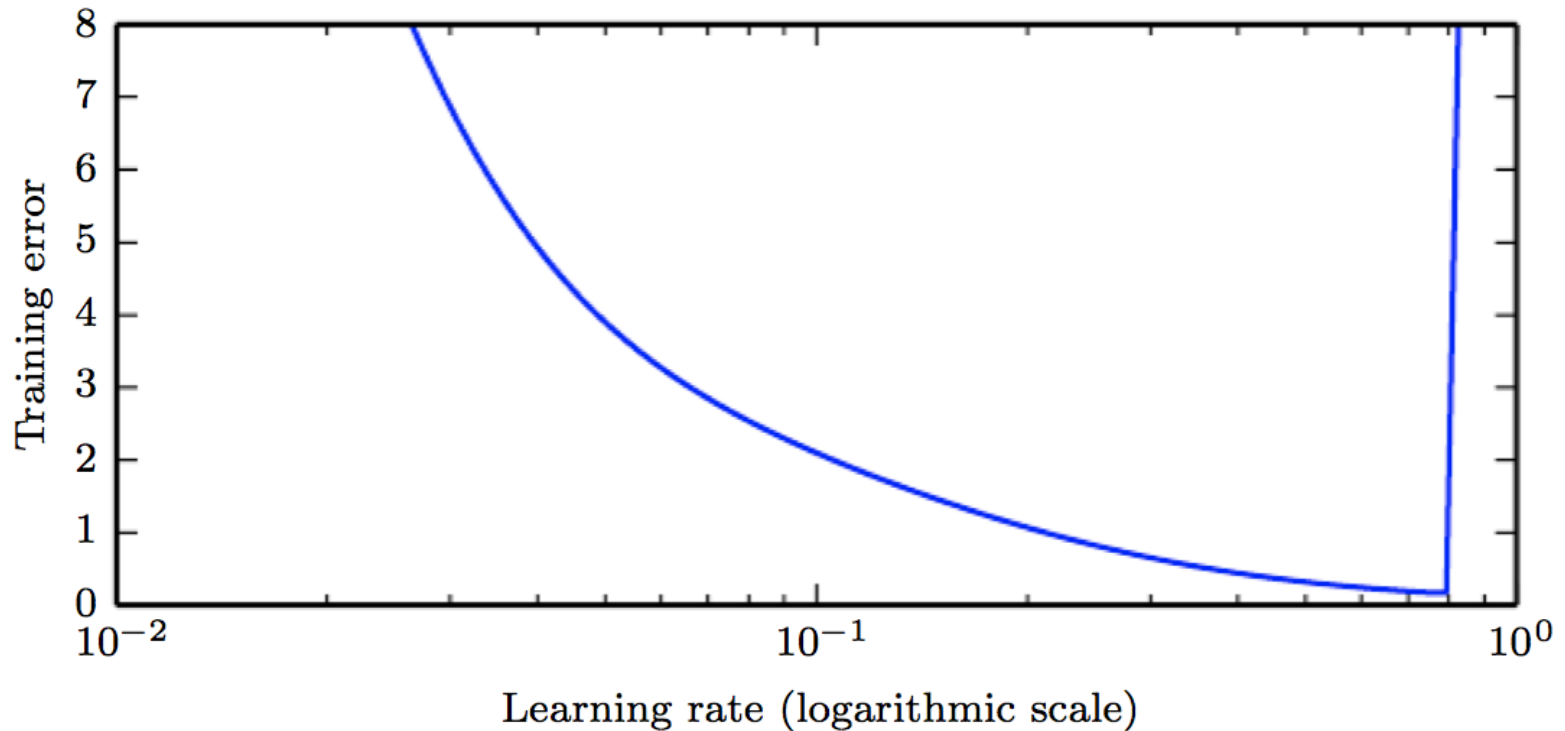
- Understanding the relationship between
  - Hyperparameters
  - Training error
  - Generalization error
  - Computational resources (memory and runtime)
- The goal of manual hyperparameter search
  - To find the **lowest generalization error**
- Three factors
  - The representational **capacity** of the model
  - The ability to minimize **the cost function** used to train the model
  - The degree to which training procedure **regularize** the model

# Selecting Hyperparameters - manual

- The hyperparameter value corresponds to **low** capacity
  - Generalization error is **high**, training error is **high**
  - **Underfitting** regime
- The hyperparameter value corresponds to **high** capacity
  - Generalization error is **high**, training error is **low**
  - **Overfitting** regime
- Somewhere in the **middle** lies the optimal capacity

# Selecting Hyperparameters - manual

- Learning rate



# Selecting Hyperparameters - manual

## ● The effect of various hyperparameters on capacity

Hyperparameter	Increase capacity when..
Number of hidden units	increased
Learning rate	tuned optimally
Convolution kernel width	increased
Implicit zero padding	increased
Weight decay coefficient	decreased
<b>Dropout rate</b>	<b>decreased</b>

# Selecting Hyperparameters – automatic

- Manual hyperparameter tuning can work very well
  - When the user has a good starting point
  - When the user has months or years of experience
- However, for many applications,
  - These starting points are not available
  - In these cases, automated algorithms can be useful
- Grid search, random search, model-based optimization

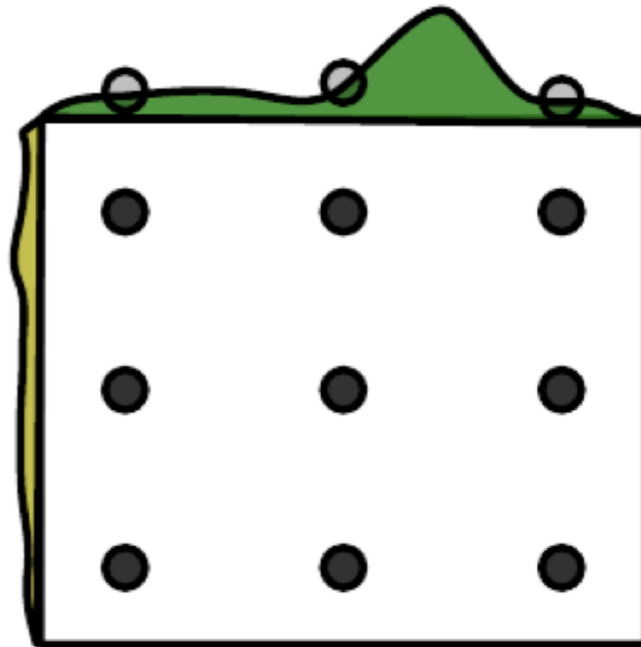
# Grid search(parameter sweep)

- Best value
- Shifting the grid
- Zooming in

Large computation cost  $O(n^m)$

$m$  : # of hyperparameters

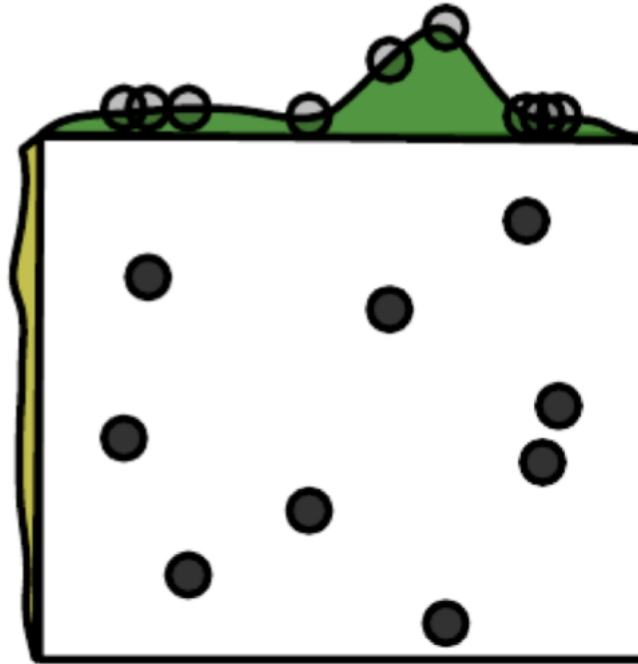
$n$  : # of values





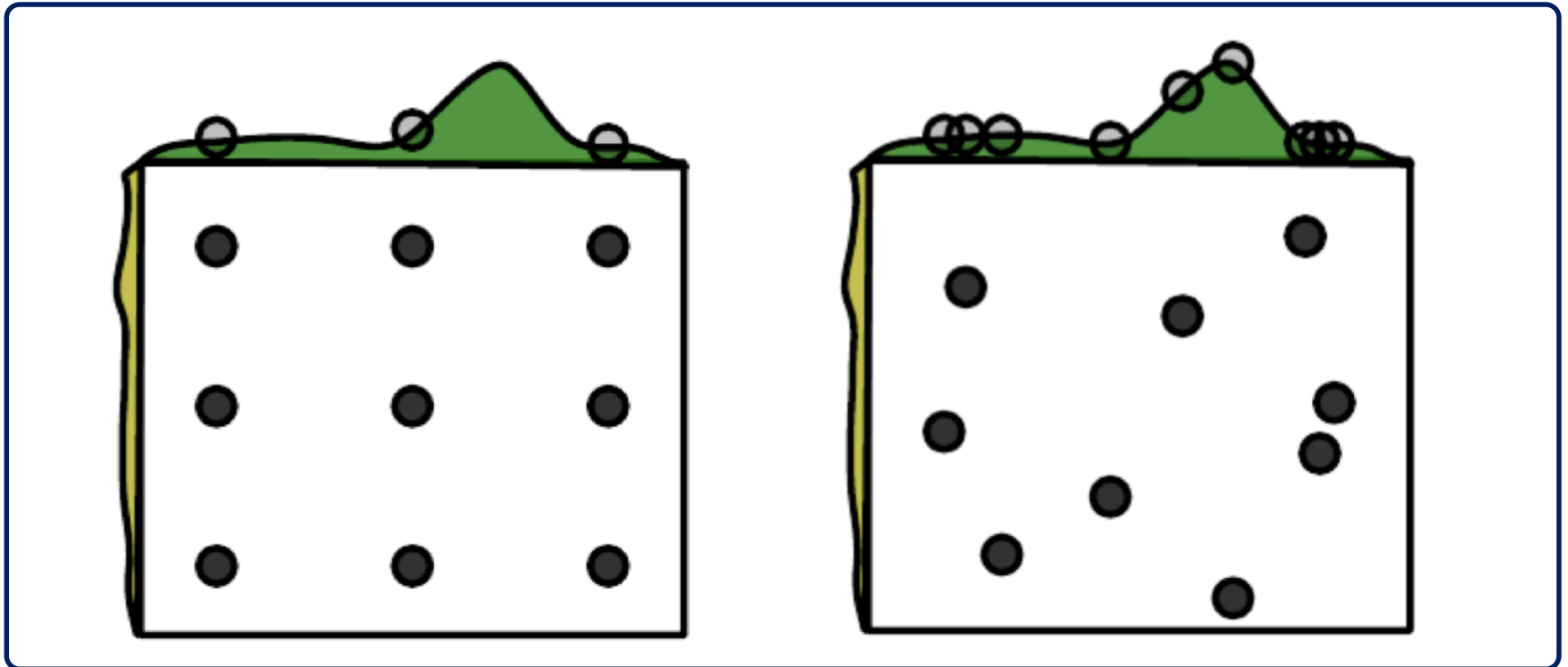
# Random search(parameter sweep)

- Probability distribution



# Grid and random search

- All the trials are independent
- Parallelization is pros



# Bayesian inference

- Set of data samples  $\{x^{(1)}, x^{(2)}, \dots, x^{(m)}\}$
- Parameter  $\theta$  is represented as random variable
- Combine the data likelihood with the prior via Bayes' rule:

$$\boxed{p(\theta | x^{(1)}, \dots, x^{(m)})} = \frac{\overset{\text{likelihood}}{p(x^{(1)}, \dots, x^{(m)} | \theta)} \overset{\text{prior}}{p(\theta)}}{p(x^{(1)}, \dots, x^{(m)})}$$

Bayesian inference

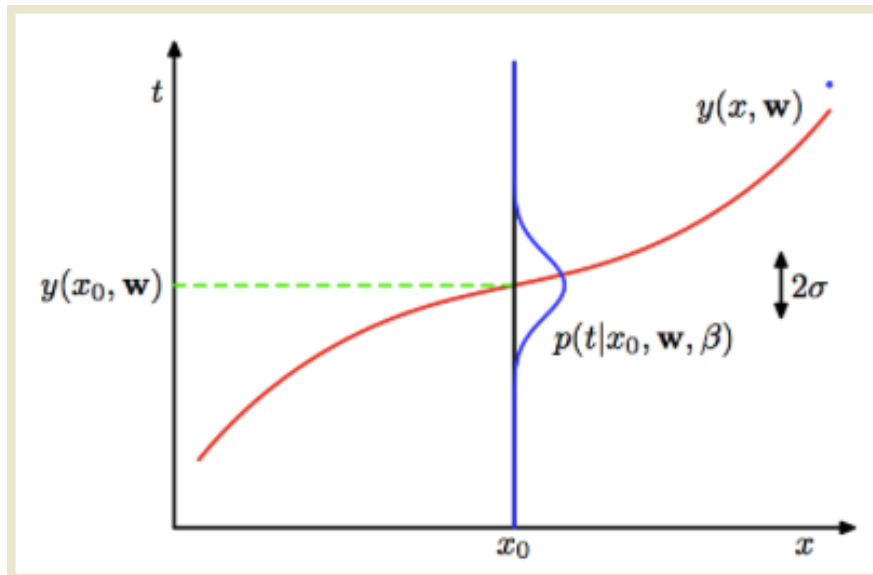
# Maximum A Posterior (MAP) Estimation

- Chose the point of maximal posterior probability

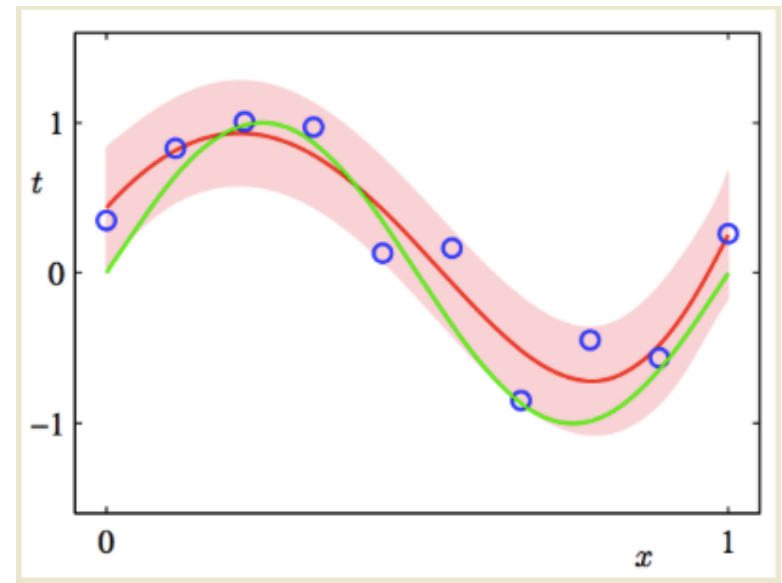
$$\theta_{MAP} = \arg \max_{\theta} \underbrace{p(\theta|x)}_{\text{posterior}} = \arg \max_{\theta} \underbrace{\log p(x|\theta)}_{\text{likelihood}} + \underbrace{\log p(\theta)}_{\text{prior}}$$

Similar with weight decay term

# MLE vs MAP



<MLE>



<MAP>

# Model-based Hyperparameter Optimization

## ● Measure space

- $(U, B, \mu)$
- $U$ : data set
- $B$ : sub sets of  $U$
- $\mu$ : measure. function from  $B$  to real space

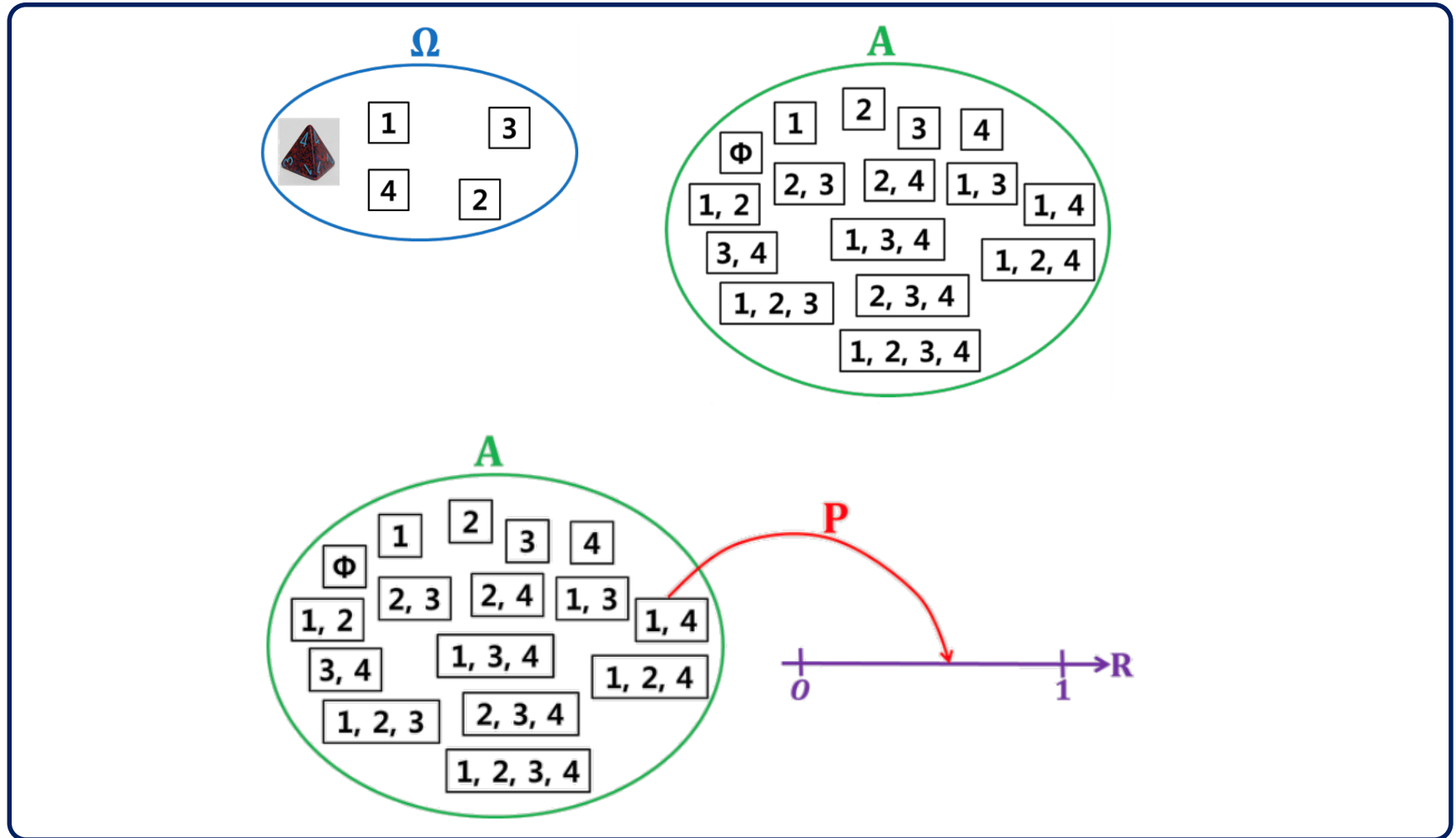
# Model-based Hyperparameter Optimization

## ● Probability space

- $(\Omega, A, P)$
- $\Omega$ : a sample space
- $A$ : a set of events
- $P$ : the assignment of probability to the events

# Model-based Hyperparameter Optimization

- Probability space  $(\Omega, A, P)$

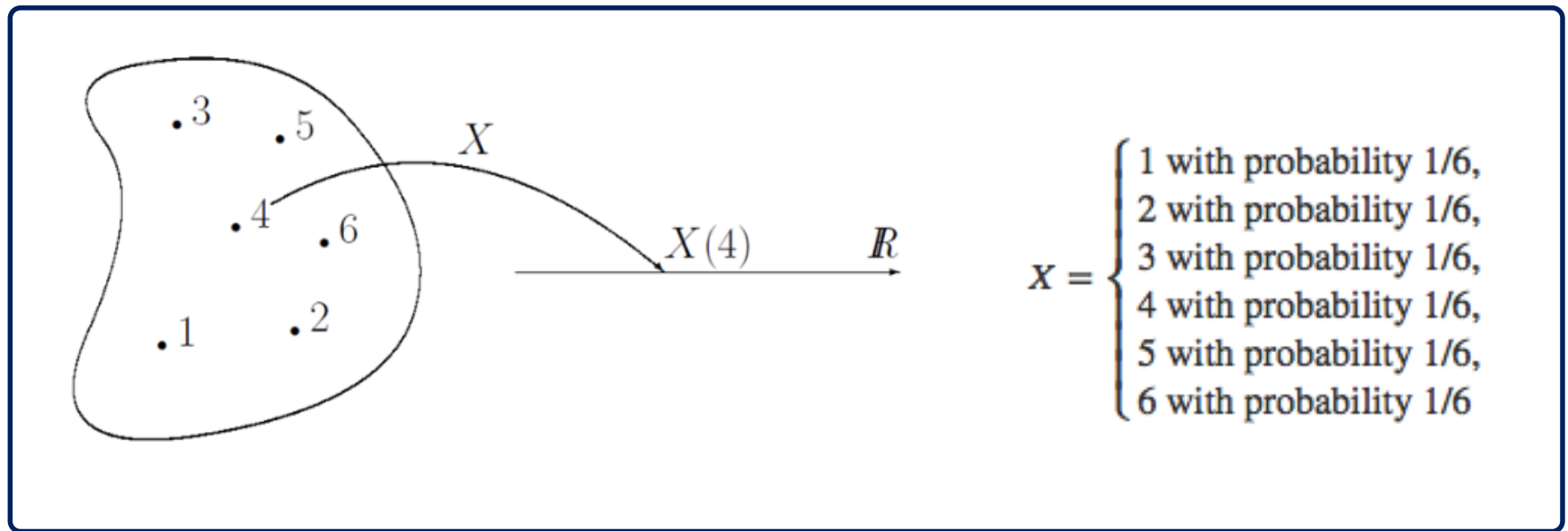




# Model-based Hyperparameter Optimization

- Random variable

- Function from probability space to real space

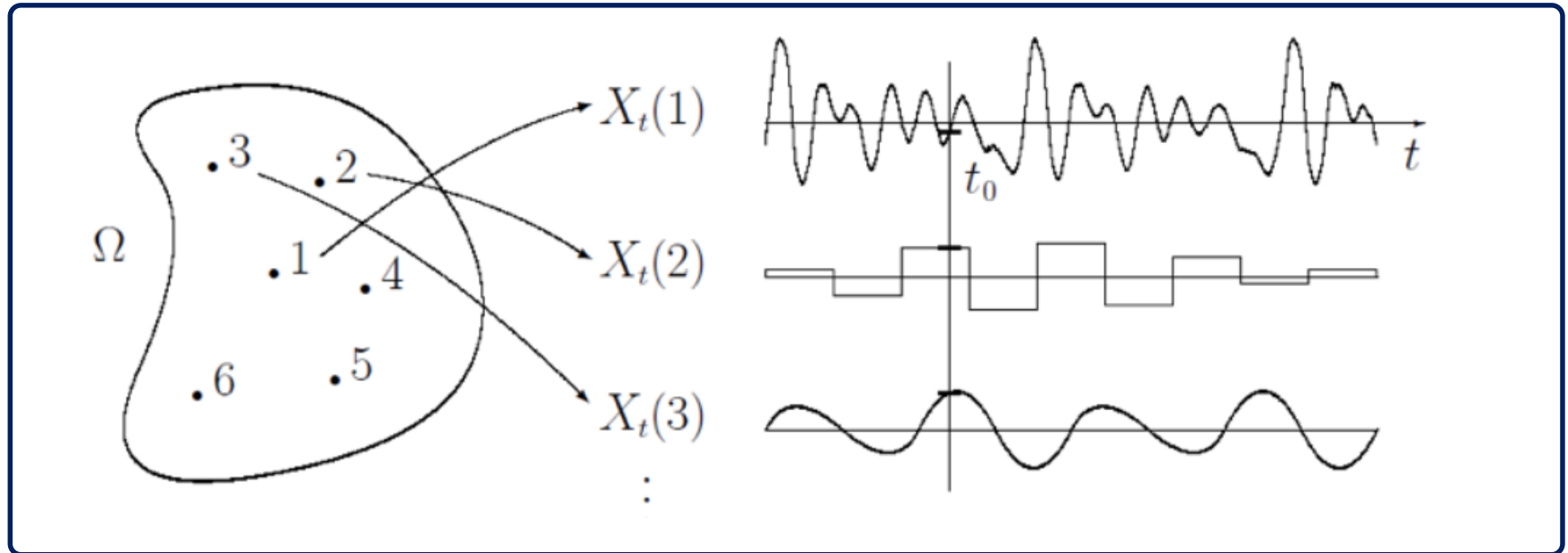


<http://sanghyukchun.github.io/99/>

# Model-based Hyperparameter Optimization

- **Stochastic process(random process)**

- Function from probability space to function space
- $X_t(w), t \in I$



<http://sanghyukchun.github.io/99/>

# Model-based Hyperparameter Optimization

## ● Gaussian process(GP)

- Random process  
finite sample  $\rightarrow$  multivariate normal distribution
- Mean function:  $m(x)$
- $\{x_1, \dots, x_n\}$ , random variable  $\{h(x_1), \dots, h(x_n)\}$

$$\begin{bmatrix} h(x_1) \\ \vdots \\ h(x_n) \end{bmatrix} \sim \mathcal{N} \left( \begin{bmatrix} m(x_1) \\ \vdots \\ m(x_n) \end{bmatrix}, \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix} \right)$$

$$f(x) \sim GP(m(x), k(x, x'))$$

$$f_i = f(x_i) \quad x_i: i\text{-th data of data set}$$

# Model-based Hyperparameter Optimization

## ● Covariance matrix

$$k_{sqe}(x, x') = \alpha \exp \left\{ -\frac{1}{2} \sum_{d=1}^D \left( \frac{x_d - x'_d}{\theta_d} \right)^2 \right\}$$

$$K = \begin{pmatrix} k_{1,1} & k_{1,2} & \cdots & k_{1,n} \\ k_{2,1} & k_{2,2} & \cdots & k_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ k_{n,1} & k_{n,2} & \cdots & k_{n,n} \end{pmatrix}$$

$k_{sqe}$ : squared-exponential kernel function  
(covariance function)

$x$ : one point

$x'$ : another point

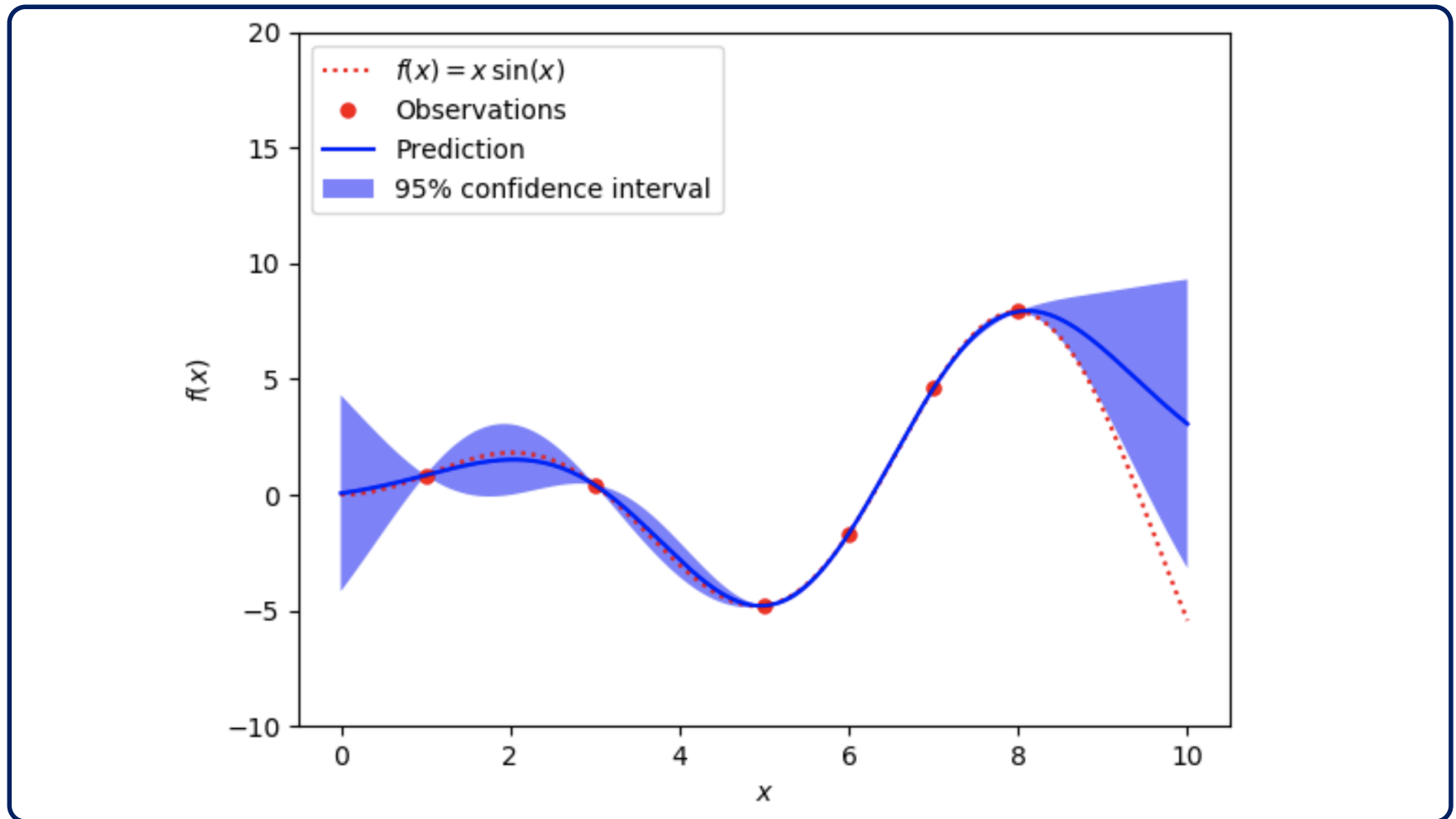
$x_d$ :  $d$  dimension value of  $x$

$\alpha, \theta_d$ : hyperparameter

$K$ : covariance matrix

# Model-based Hyperparameter Optimization

## ● Gaussian Process Regression(GPR)



<http://sanghyukchun.github.io/99/>

# Model-based Hyperparameter Optimization

## ● Bayesian Optimization for “Black-box” function

$$x^* = \arg \min_{x \in X} f(x).$$

- Estimate  $f(x)$  by using data
- Choice point by using decision rule
- Adding the point to data and repeat this until achieve criteria

## ● Acquisition function $EI(x)$

- Balancing between explore and exploit

# Model-based Hyperparameter Optimization

## ● Acquisition function(AF) - Probability of Improvement

$$\gamma(\mathbf{x}) = \frac{f(\mathbf{x}_{\text{best}}) - \mu(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)}{\sigma(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)}$$

$$a_{\text{PI}}(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta) = \Phi(\gamma(\mathbf{x}))$$

$a(x; \{x_n, y_n\}, \theta)$ : dependence(AF and previous observations)

$\{x_n, y_n\}_{n=1}^N$ : observations form

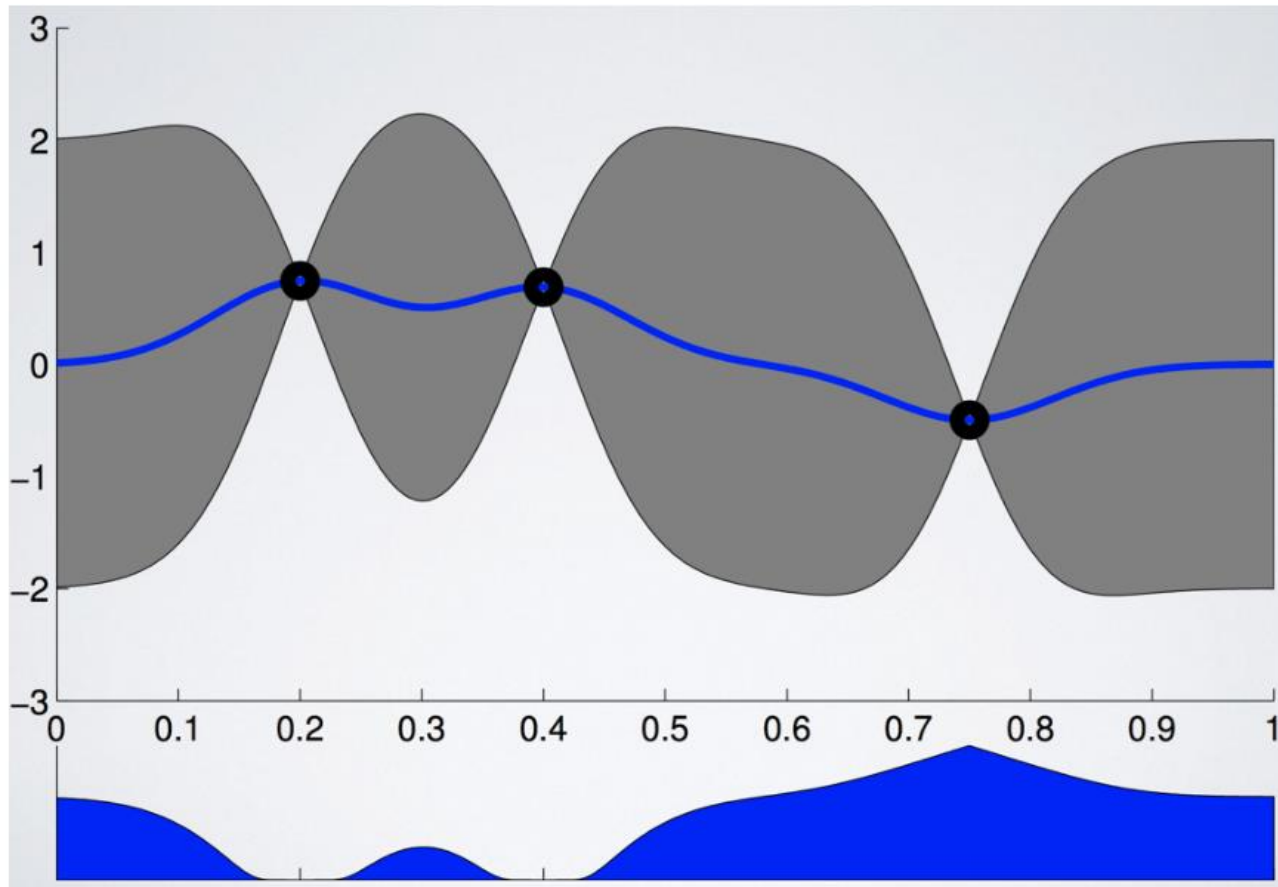
$x_{\text{best}}$ : best current value as  $x_{\text{best}} = \operatorname{argmin}_{x_n} f(x_n)$

$\phi(\cdot)$ : cumulative distribution function

[NIPS 12] Practical Bayesian Optimization of Machine Learning Algorithms

# Model-based Hyperparameter Optimization

- Acquisition function(AF) - Probability of Improvement



<http://sanghyukchun.github.io/99/>



# Model-based Hyperparameter Optimization

## ● Acquisition function - Expected Improvement

$$\gamma(\mathbf{x}) = \frac{f(\mathbf{x}_{\text{best}}) - \mu(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)}{\sigma(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta)}$$

$$a_{\text{EI}}(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta) = \sigma(\mathbf{x}; \{\mathbf{x}_n, y_n\}, \theta) (\gamma(\mathbf{x}) \Phi(\gamma(\mathbf{x})) + \mathcal{N}(\gamma(\mathbf{x}); 0, 1))$$

$a(x; \{x_n, y_n\}, \theta)$ : dependence(AF and previous observations)

$\{x_n, y_n\}_{n=1}^N$ : observations form

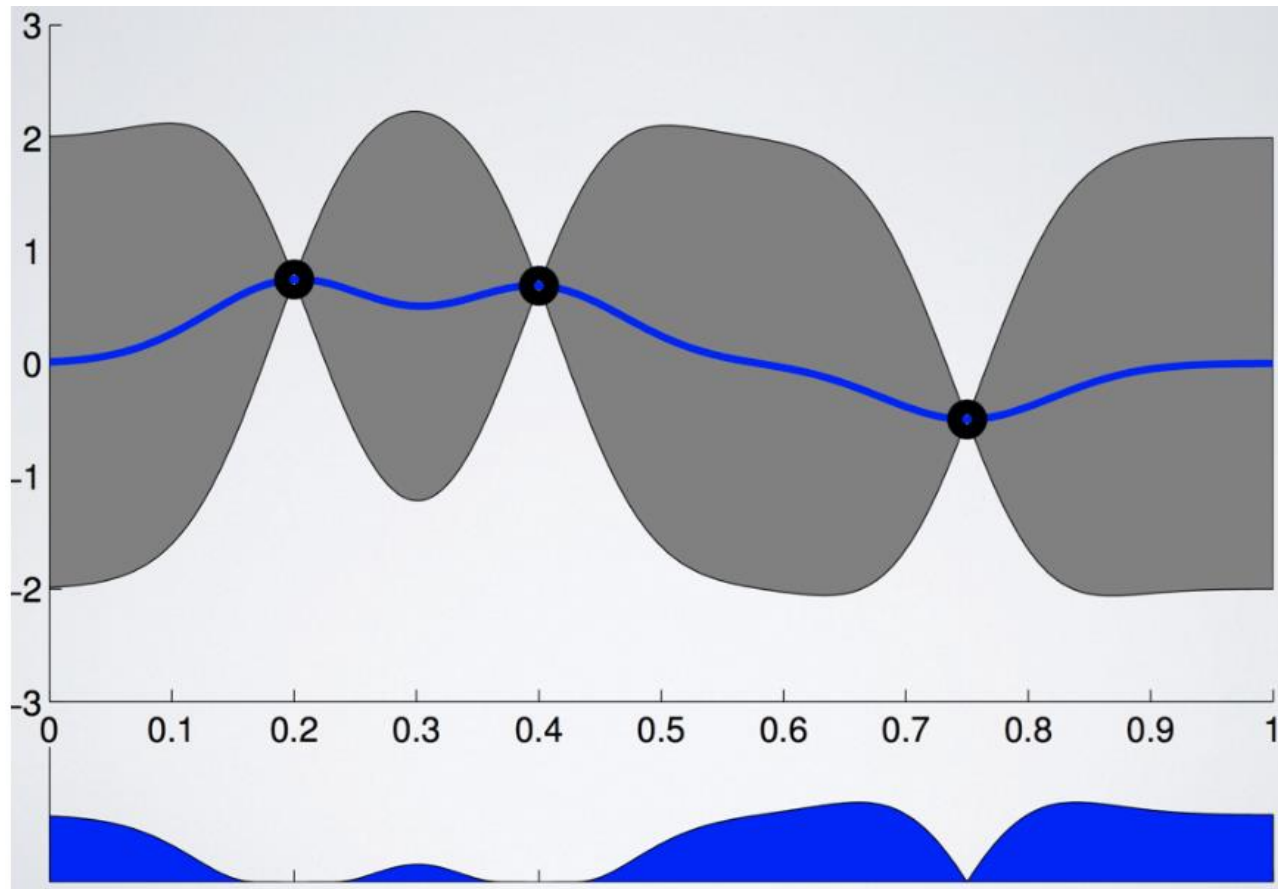
$x_{\text{best}}$ : best current value as  $x_{\text{best}} = \operatorname{argmin}_{x_n} f(x_n)$

$\phi(\cdot)$ : cumulative distribution function

[NIPS 12] Practical Bayesian Optimization of Machine Learning Algorithms

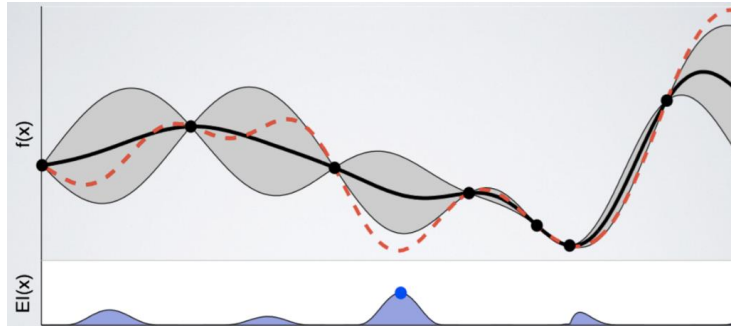
# Model-based Hyperparameter Optimization

- Acquisition function - Expected Improvement



<http://sanghyukchun.github.io/99/>

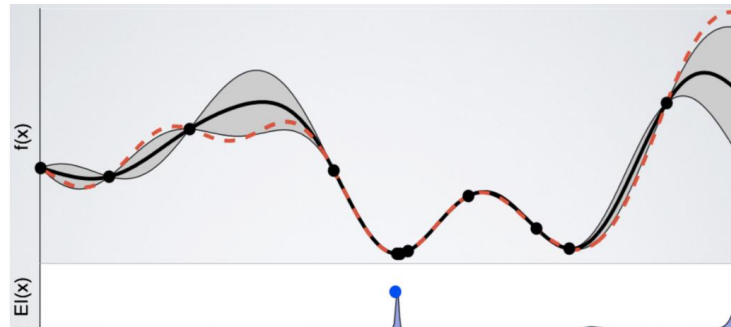
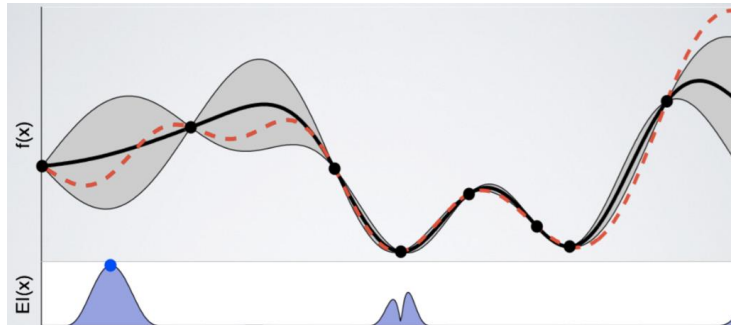
# Practical Bayesian Optimization of Machine Learning Algorithms



Red line: unknown black box function  $f(x)$

Black line: estimation

$EI(x)$ : acquisition function



<http://sanghyukchun.github.io/99/>

# Model-based Hyperparameter Optimization

- Limitation of Bayesian Optimization

- Tuning the second hyper parameters(e.g. kernel function)
- No guide about stochastic assumption
- Impossible parallelization
- Difficult to implement software

- To solve this, refer to this paper

- [NIPS 12] Practical Bayesian Optimization of Machine Learning Algorithms
- Jasper Snoek, Hugo Larochelle, Ryan P. Adams
- Citation: 1031

# Contents

## ● Chapter 11 Practical Methodology

11.1 Performance Metrics

11.2 Default Baseline Models

11.3 Determining Whether to Gather More Data

11.4 Selecting Hyperparameters

**11.5 Debugging Strategies**

11.6 Example: Multi-Digit Number Recognition

## ● Chapter 12 Applications

12.1 LargeScale DeepLearning

12.2 Computer Vision

12.3 Speech Recognition

12.4 Natural Language Processing

12.5 Other Applications

# Debugging

- Deep learning systems are difficult to debug
- Bias update error example
  - $b \leftarrow b - \alpha$   $b$ : biases,  $\alpha$ : learning rate
  - It is so difficult to find this error
- Debugging tests
  - Visualize the model in action
  - Reasoning about software using train and test error
  - Fit a tiny dataset
  - Compare back-propagated derivatives to numerical derivatives
  - Monitor histograms of activations and gradient

# Contents

## ● Chapter 11 Practical Methodology

11.1 Performance Metrics

11.2 Default Baseline Models

11.3 Determining Whether to Gather More Data

11.4 Selecting Hyperparameters

11.5 Debugging Strategies

11.6 Example: Multi-Digit Number Recognition

## ● Chapter 12 Applications

12.1 LargeScale DeepLearning

**12.2 Computer Vision**

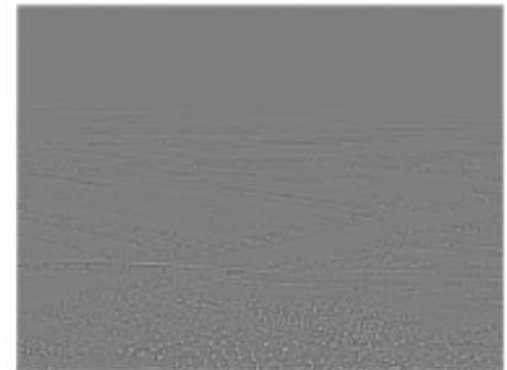
12.3 Speech Recognition

12.4 Natural Language Processing

12.5 Other Applications

# Computer Vision – Contrast Normalization

- A comparison of global and local contrast normalization



Input image

GCN

LCN



# Computer Vision – Contrast Normalization

- Contrast Normalization

- Contrast formula

- One of the sources of variation that can be safely removed

$$\bar{\mathbf{X}} = \frac{1}{3rc} \sum_{i=1}^r \sum_{j=1}^c \sum_{k=1}^3 X_{i,j,k}$$

$$\sqrt{\frac{1}{3rc} \sum_{i=1}^r \sum_{j=1}^c \sum_{k=1}^3 (X_{i,j,k} - \bar{\mathbf{X}})^2}$$

$X \in \mathbb{R}^{r \times c \times 3}$ : image tensor

$X_{i,j,1}$ : red intensity

$X_{i,j,2}$ : green intensity

$X_{i,j,3}$ : blue intensity

# Computer Vision – Contrast Normalization

- **Global contrast normalization(GCN)**

- Preventing images from having varying amounts of contrast

$$X'_{i,j,k} = s \frac{X_{i,j,k} - \bar{X}}{\max \left\{ \epsilon, \sqrt{\lambda + \frac{1}{3rc} \sum_{i=1}^r \sum_{j=1}^c \sum_{k=1}^3 (X_{i,j,k} - \bar{X})^2} \right\}}$$

$X_{i,j,1}$ : red intensity

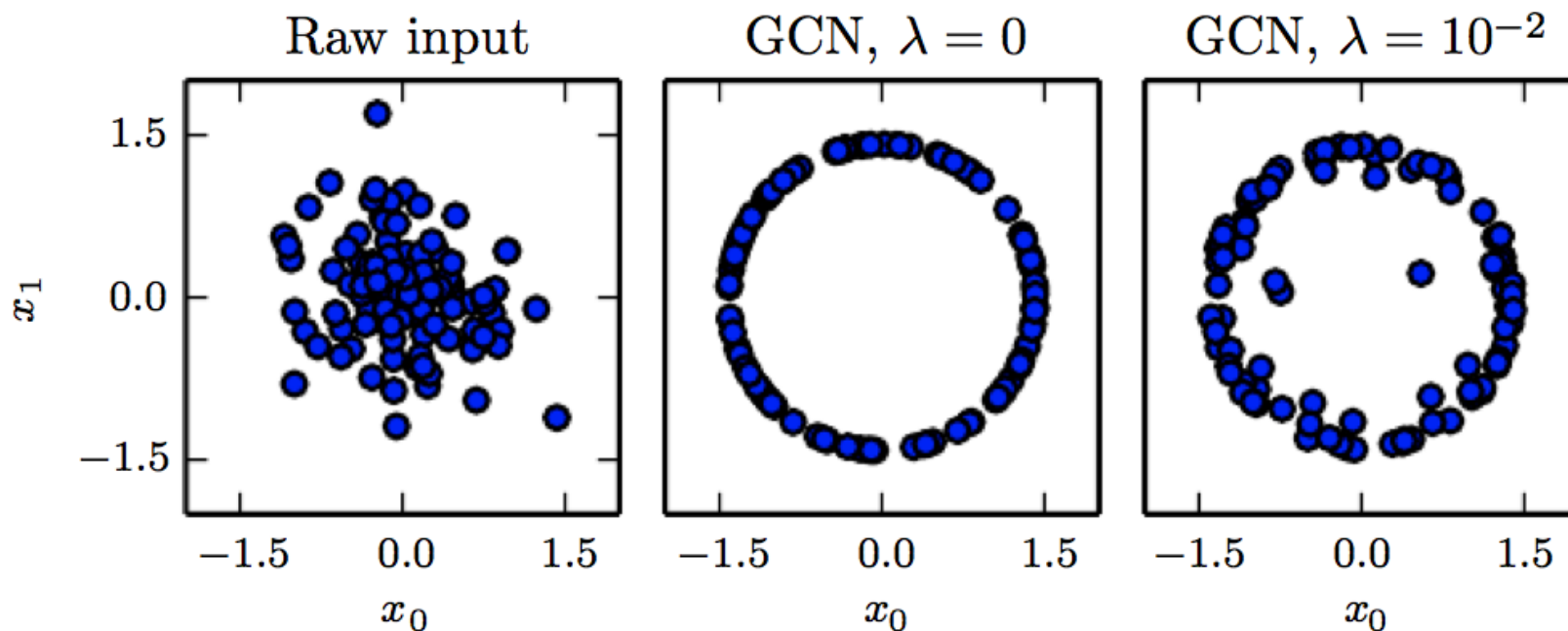
$X_{i,j,2}$ : green intensity

$X_{i,j,3}$ : blue intensity

$s, \epsilon, \lambda$ : hyper parameters

# Computer Vision – Contrast Normalization

- GCN maps examples onto a sphere



**Thank you**