

Logistic Regression(Logistic Model, Decision Boundary),

Classification

There are many examples for binary/multi-class classification tasks

- Email : Spam / Not Spam?
- Online Transactions : Fraudulent (Yes / No)?
- Tumor : Malignant / Benign?

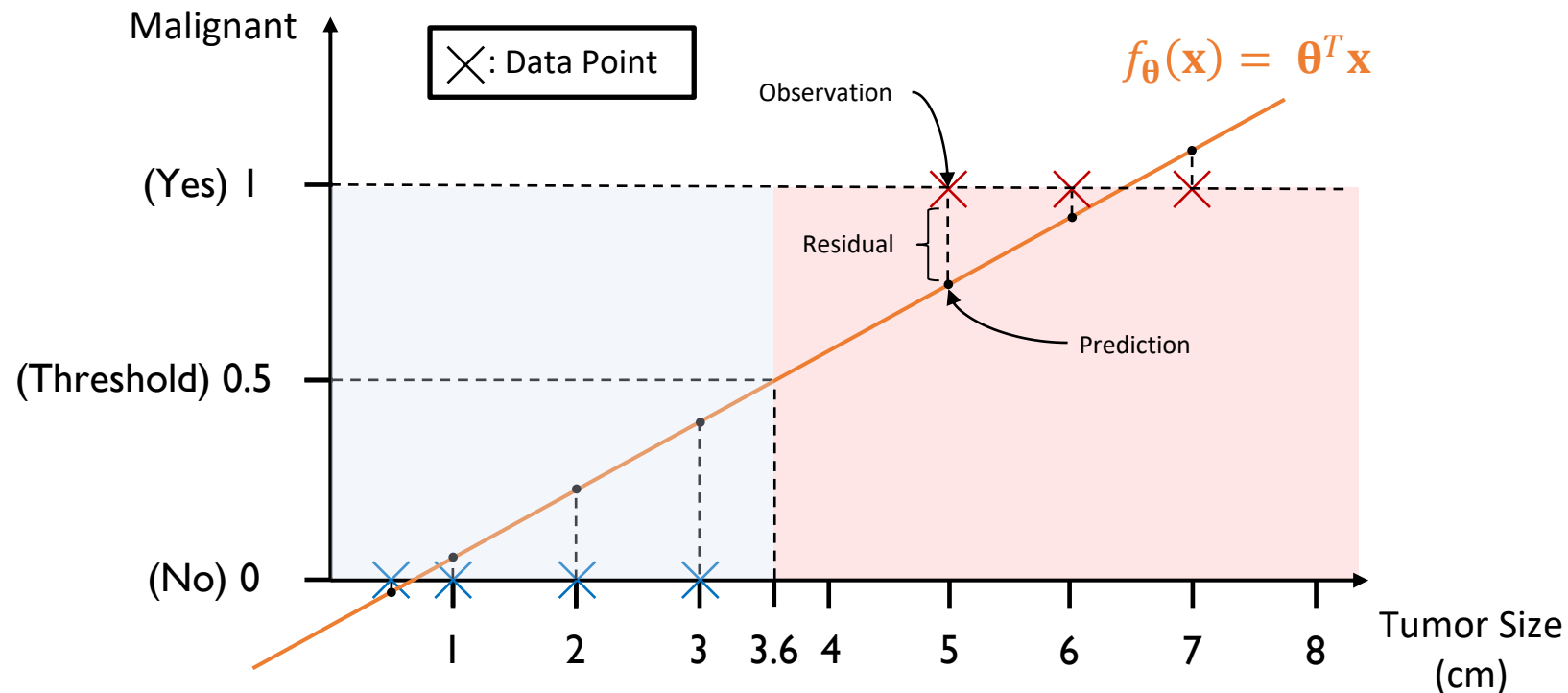
- Label for Binary Classification (benign tumor, malignant tumor) : $y \in \{0,1\}$

- Label for 4-class Classification (A, B, AB, O) : $y \in \{0,1,2,3\}$

Classification

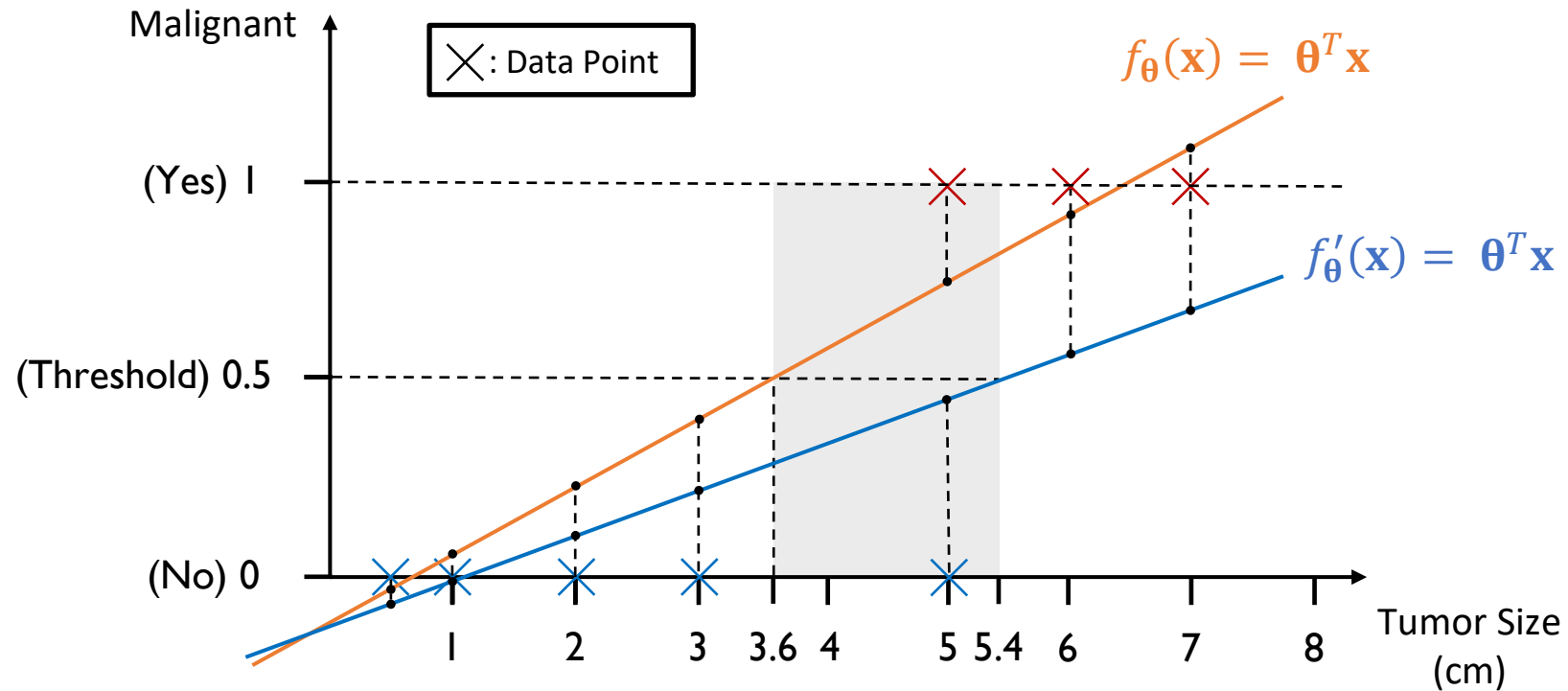
We can make a classification model that uses linear regression for classifying tumors

The model should be designed to minimize the difference between the predictions and the observations



Classification

The classification result of the linear regression model can vary depending on the model.



Classification vs Logistic Regression

There is a difference between the classification outputs and the logistic regression outputs

If a linear regression model is $f(\mathbf{x})$ and its output is y , $-\infty \leq y \leq \infty$

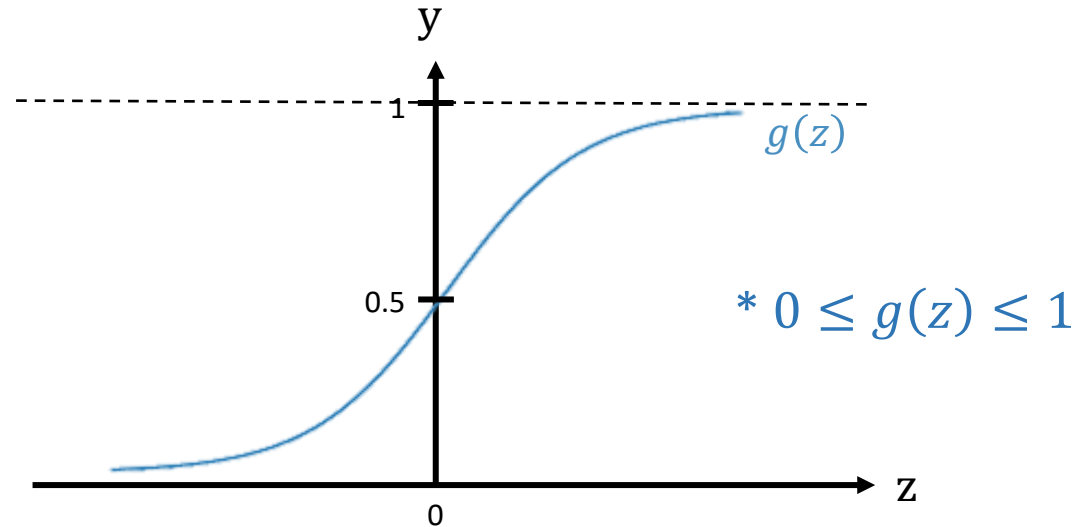
If a logistic regression model is $h(\mathbf{x})$ and its output is y , $0 \leq y \leq 1$

↑
How and why?

Logistic Regression Model

If we want to limit the range of y between 0 and 1, we can use sigmoid(logistic) function to limit the range of y

- Sigmoid(logistic) function $h_{\theta}(\mathbf{x}) = g(\theta^T \mathbf{x}) = \frac{1}{1+e^{-\theta^T \mathbf{x}}}$
- If $\theta^T \mathbf{x} = \mathbf{z}$,



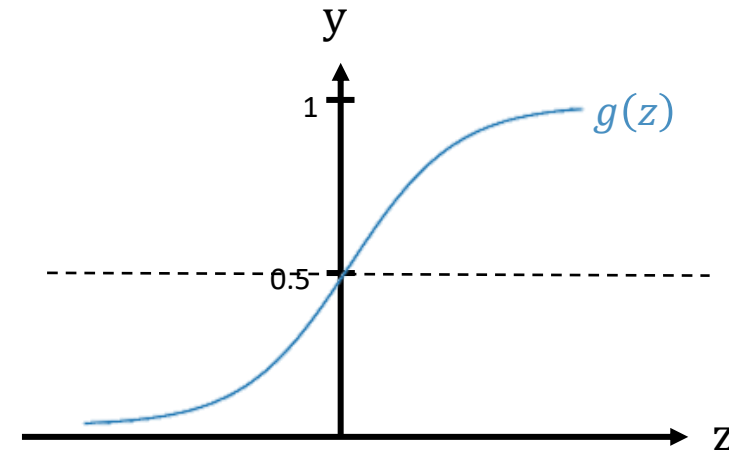
Logistic Regression

Logistic regression model $h_{\theta}(\mathbf{x})$ returns classification results based on the threshold

- $h_{\theta}(\mathbf{x}) = g(\theta^T \mathbf{x}) = \frac{1}{1+e^{-\theta^T \mathbf{x}}}$, $\theta^T \mathbf{x} = \mathbf{z}$, threshold = 0.5

- If $g(\mathbf{z}) \geq 0.5$, predict $y = 1$
 $= \mathbf{z} \geq 0$
 $= \theta^T \mathbf{x} \geq 0$

- If $g(\mathbf{z}) < 0.5$, predict $y = 0$
 $= \mathbf{z} < 0$
 $= \theta^T \mathbf{x} < 0$

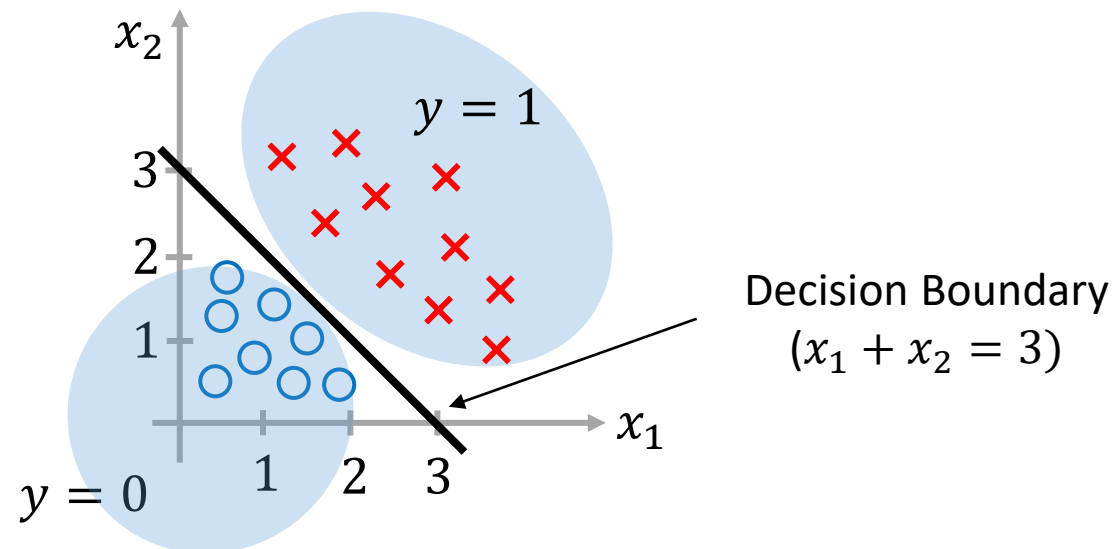


Decision Boundary

We can build a classification model $h_{\theta}(\mathbf{x})$ to predict house price

$$h_{\theta}(\mathbf{x}) = g(\theta_0 + \theta_1 \underset{\text{size}}{x_1} + \theta_2 \underset{\text{age}}{x_2}), \quad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix}$$

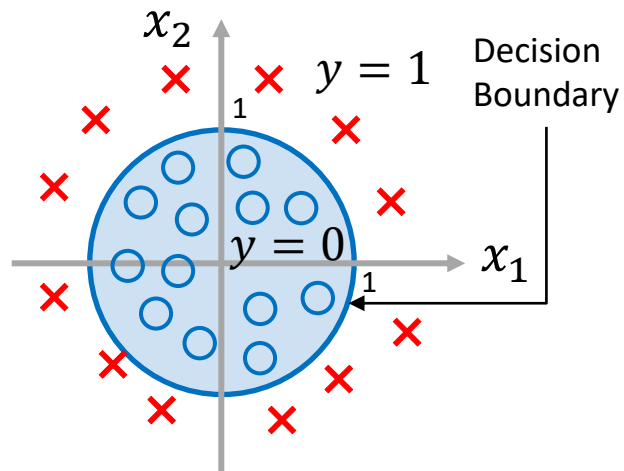
If the model predict “ $y = 1$ ” when $-3 + x_1 + x_2 \geq 0$, we can draw a decision boundary



Non-linear Decision Boundaries

- $h_{\theta}(\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2), \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$

If the model predict “ $y = 1$ ” when $-1 + x_1^2 + x_2^2 \geq 0$



How to Optimize Parameters θ ?

If we set a logistic regression model $h_{\theta}(\mathbf{x}) = \frac{1}{1+e^{-\theta^T \mathbf{x}}}$ for binary classification ($y \in \{0,1\}$) and train the model with a training dataset $\{(\mathbf{x}^{(1)}, y^{(1)}), (\mathbf{x}^{(2)}, y^{(2)}), \dots, (\mathbf{x}^{(m)}, y^{(m)})\}$ (m : number of examples),

How to optimize parameters θ ?

Cost Function of Logistic Regression

Cost function for a logistic regression model : $J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}, y^{(i)}))$

- If we use a mean-squared error as a cost function,

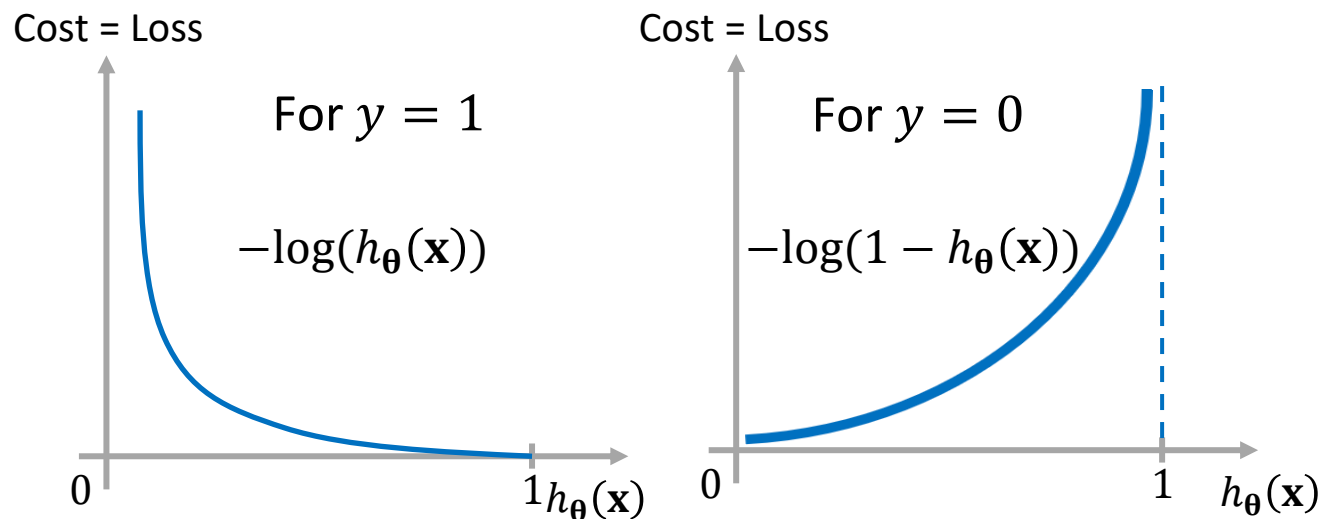
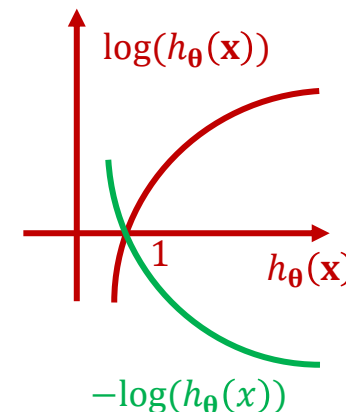
$$\text{Cost}(h_{\boldsymbol{\theta}}(\mathbf{x}, y)) = \frac{1}{2m} \sum_{i=1}^m (h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)})^2$$

When $y^{(i)} = 1$,

- If $h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) \geq 0.5$, Classification output is 1 and $\text{Cost}(h_{\boldsymbol{\theta}}(\mathbf{x}, y)) = 0$
- If $h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) < 0.5$, classification output is 0 and $\text{Cost}(h_{\boldsymbol{\theta}}(\mathbf{x}, y))$ can be 30?

Cost Function of Logistic Regression

$$\text{Cost}(h_{\theta}(\mathbf{x}, y)) = \begin{cases} -\log(h_{\theta}(\mathbf{x})) & \text{if } y = 1 \\ -\log(1 - h_{\theta}(\mathbf{x})) & \text{if } y = 0 \end{cases}$$



- If $h_{\theta}(\mathbf{x}) = 1$ and $y = 1$, $\text{Cost} = 0$
But as $h_{\theta}(\mathbf{x}) \rightarrow 0$, $\text{Cost} \rightarrow \infty$
- Captures intuition that if $h_{\theta}(\mathbf{x}) = 0$ (predict $P(y = 1|\mathbf{x}; \theta) = 0$) but $y = 1$, we will penalize the learning algorithm by a very large cost.

Cost Function of Logistic Regression

- $J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}), y^{(i)})$, $\text{Cost}(h_{\boldsymbol{\theta}}(\mathbf{x}), y) = \begin{cases} -\log(h_{\boldsymbol{\theta}}(\mathbf{x})) & \text{if } y = 1 \\ -\log(1 - h_{\boldsymbol{\theta}}(\mathbf{x})) & \text{if } y = 0 \end{cases}$

Note : Always $y = 0$ or 1

We can represent the loss function in both cases as an equation

- $\text{Cost}(h_{\boldsymbol{\theta}}(\mathbf{x}), y) = -y \log(h_{\boldsymbol{\theta}}(\mathbf{x})) - (1 - y) \log(1 - h_{\boldsymbol{\theta}}(\mathbf{x}))$
- To fit parameters $\boldsymbol{\theta} : \min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$
- To make a prediction given new \mathbf{x} :

Output:
$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

Gradient Descent

We can apply a gradient descent method to minimize $J(\boldsymbol{\theta})$

$$J(\boldsymbol{\theta}) = \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h_{\boldsymbol{\theta}}(\mathbf{x})) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(\mathbf{x})) \right]$$

If we want $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$:

Repeat $\boldsymbol{\theta}_{j+1} := \boldsymbol{\theta}_j - \alpha \frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta})$ (simultaneously update all $\boldsymbol{\theta}_j$)

$$\frac{\partial}{\partial \theta_j} J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^m (h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}) \mathbf{x}_j^{(i)}$$

Gradient Descent Example

$$h(ax + b) = \frac{1}{1 + e^{-(ax+b)}} = y, z = ax + b$$

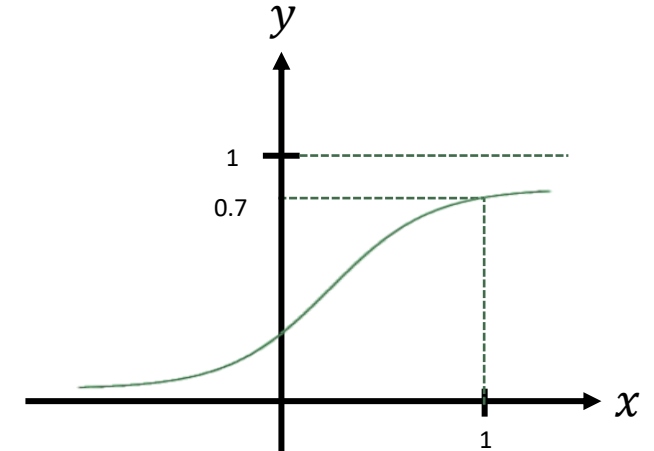
If we set an $a = 2$ and $b = -5$, $z = 2x - 5$

$$J(a, b) = -\log \frac{1}{1 + e^{-(ax+b)}}$$

$$\frac{\partial}{\partial a} J(a, b) = -\frac{1}{\frac{1}{1 + e^{-(ax+b)}}} \times \frac{-1}{(1 + e^{-(ax+b)})^2} \times (-e^{-(ax+b)}) \times x$$

$$= \frac{-e^{-\boxed{(ax+b)}}^{\boxed{z}}}{(1 + e^{-\boxed{(ax+b)}})} \times x = -\left(1 - \frac{1}{(1 + e^{-z})}\right) \times x$$

$$= \left(\frac{1}{(1 + e^{-z})} - 1\right) \times x = (h_{\theta}(\mathbf{x}) - 1) \times x$$



$$(\log_e x)' = \frac{1}{x}$$

$$\left(\frac{1}{x}\right)' = -\frac{1}{x^2}$$

$$(1 + e^{-x})' = -e^{-x}$$

Gradient Descent Example

- Positive class : $\begin{cases} e^{ax+b} \\ e^0 \end{cases} \rightarrow \begin{cases} e^1 \\ e^0 \end{cases} \rightarrow \begin{cases} 2.7 \\ 1 \end{cases} \rightarrow \text{sigmoid output for positive class : } \frac{2.7}{1+2.7}$
- $\frac{1}{1+e^{-z}} = \frac{e^z}{1+e^z} = \frac{e^z}{e^0+e^z}$
- Negative class : $\begin{cases} e^{ax+b} \\ e^0 \end{cases} \rightarrow \begin{cases} e^{-1} \\ e^0 \end{cases} \rightarrow \begin{cases} \frac{1}{2.7} \\ 1 \end{cases} \rightarrow \text{sigmoid output for negative class : } \frac{0.4}{1+0.4}$
- $1 - \frac{1}{1+e^{-z}} = \frac{1+e^{-z}-1}{1+e^{-z}} = \frac{e^{-z}}{1+e^{-z}}$

Gradient Descent

$$J(\boldsymbol{\theta}) = \frac{1}{m} \left[\sum_{i=1}^m y^{(i)} \log(h_{\boldsymbol{\theta}}(\mathbf{x})) + (1 - y^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(\mathbf{x})) \right]$$

If we want $\min_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$:

Repeat $\boldsymbol{\theta}_{j+1} := \boldsymbol{\theta}_j - \alpha \sum_{i=1}^m (h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)}) \mathbf{x}_j^{(i)}$

(simultaneously update all $\boldsymbol{\theta}_j$)

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \dots \\ \theta_n \end{bmatrix}$$
$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \boldsymbol{\theta}^T \mathbf{x}$$
$$h_{\boldsymbol{\theta}}(\mathbf{x}) = \frac{1}{1 + e^{-\boldsymbol{\theta}^T \mathbf{x}}}$$

Algorithm looks identical to linear regression!

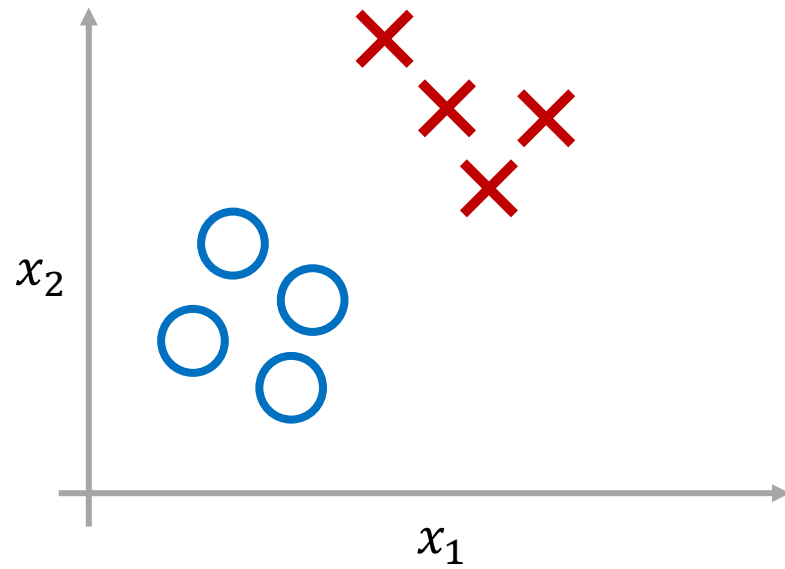
Multi-class Classification (1-vs-the-rest, 1-vs-1)

Multi-class Classification

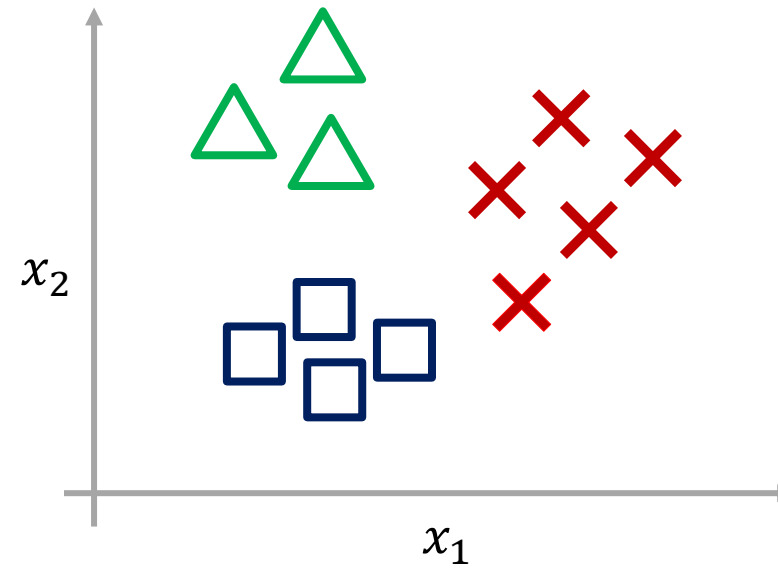
- Email foldering / tagging : Work , Friends , Family , Hobby
 $y = 0$ $y = 1$ $y = 2$ $y = 3$
- Medical diagrams : Not ill , Cold , Flu
 $y = 0$ $y = 1$ $y = 2$
- Weather : Sunny , Cloudy , Rain , Snow
 $y = 0$ $y = 1$ $y = 2$ $y = 3$

Multi-class Classification

Binary classification



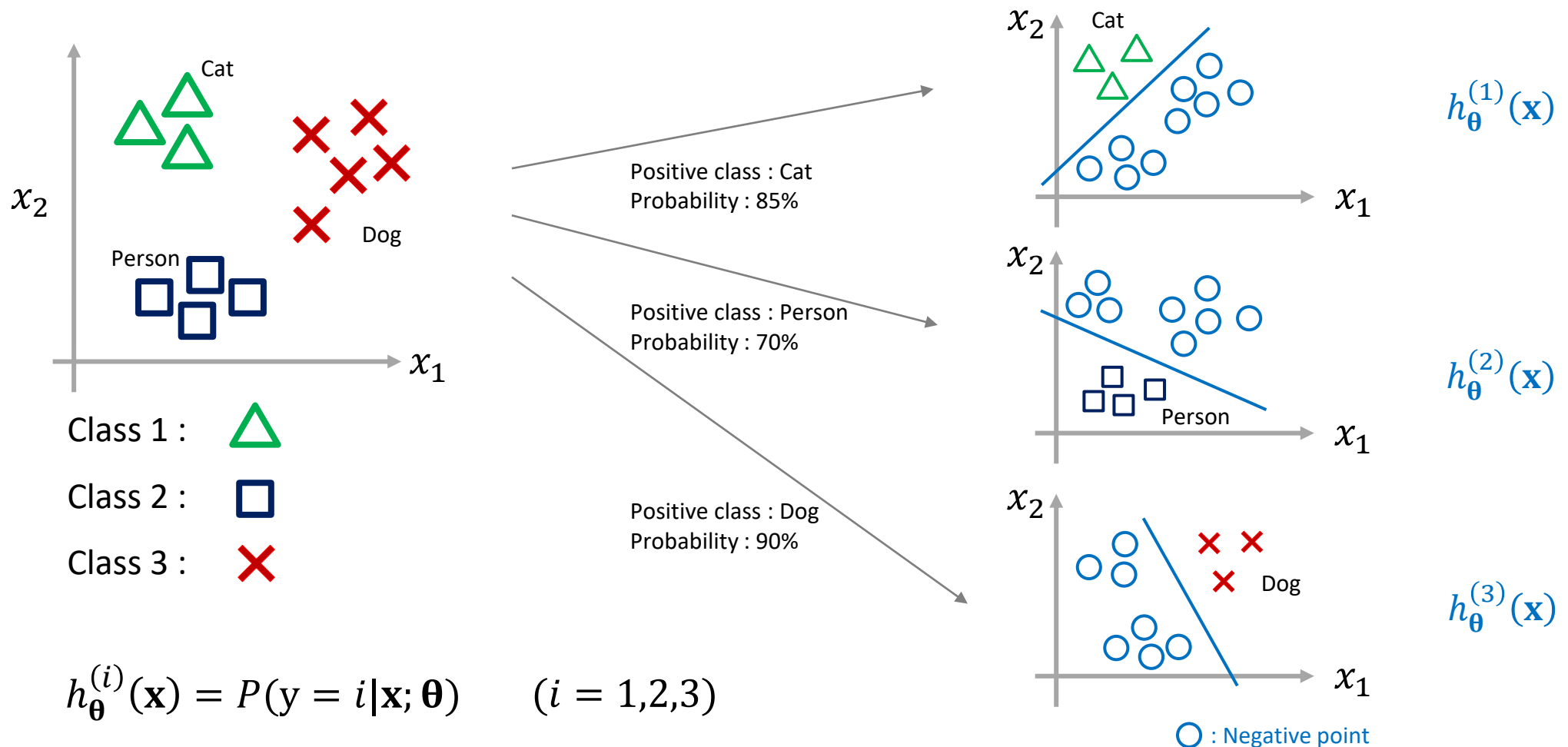
Multi-class classification



One-vs-all (one-vs-rest) Example

Train a logistic regression classifier $h_{\theta}^{(i)}(\mathbf{x})$ for each class i to predict the probability that $y = i$.

On a new input \mathbf{x} to make a prediction, pick the class i that maximizes $\max_i h_{\theta}^{(i)}(\mathbf{x})$



Multi-class Classification Example

- Binary-class classification using sigmoid

We have 3 binary-class classification models $\begin{cases} 2x_1 + 3x_2 + 4 \\ -x_1 + 5x_2 - 3 \\ -4x_1 + x_2 + 2 \end{cases}$

If $x_1 = 1$ and $x_2 = 2$,

$$\begin{cases} 2 \times 1 + 3 \times 2 + 4 = 12 \\ -1 \times 1 + 5 \times 2 - 3 = 6 \\ -4 \times 1 + 1 \times 2 + 2 = 0 \end{cases}$$

$$\text{Sigmoid output : } \begin{cases} 99.95\% \\ 98\% \\ 50\% \end{cases}$$

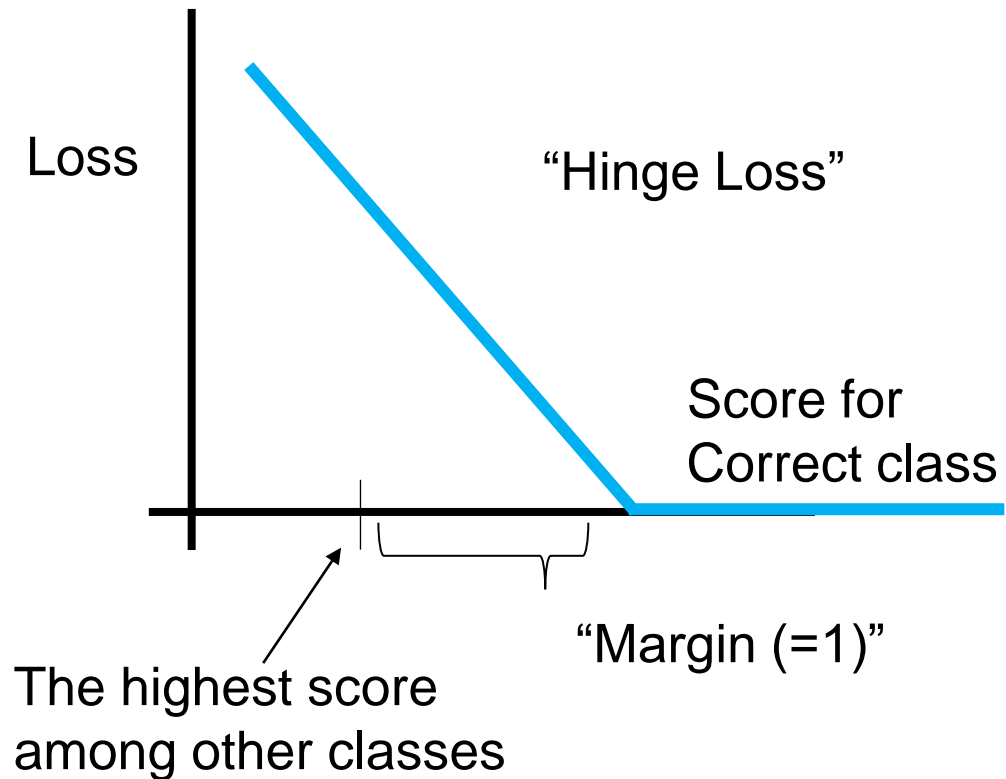
- Multi-class classification using softmax

$$\begin{cases} \text{cat} : e^{12} \\ \text{dog} : e^6 \\ \text{person} : e^0 \end{cases}$$

$$\text{Softmax output : } \begin{cases} \frac{e^{12}}{e^{12}+e^6+e^0} = 92\% \\ \frac{e^6}{e^{12}+e^6+e^0} = 7\% \\ \frac{e^0}{e^{12}+e^6+e^0} = 1\% \end{cases}$$

Hinge Loss

“The score of the correct class should be higher than all the other scores with at least a certain margin.”



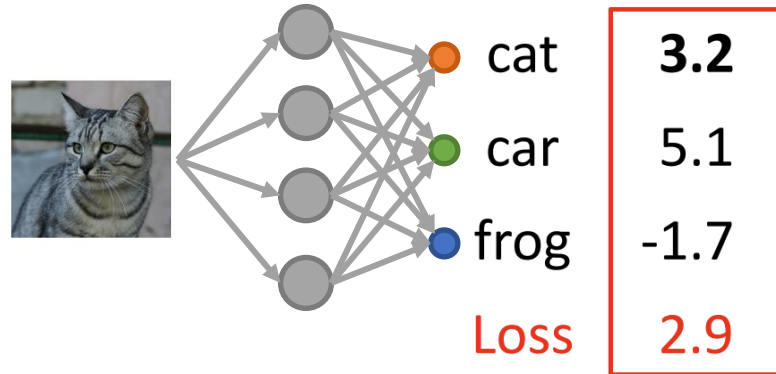
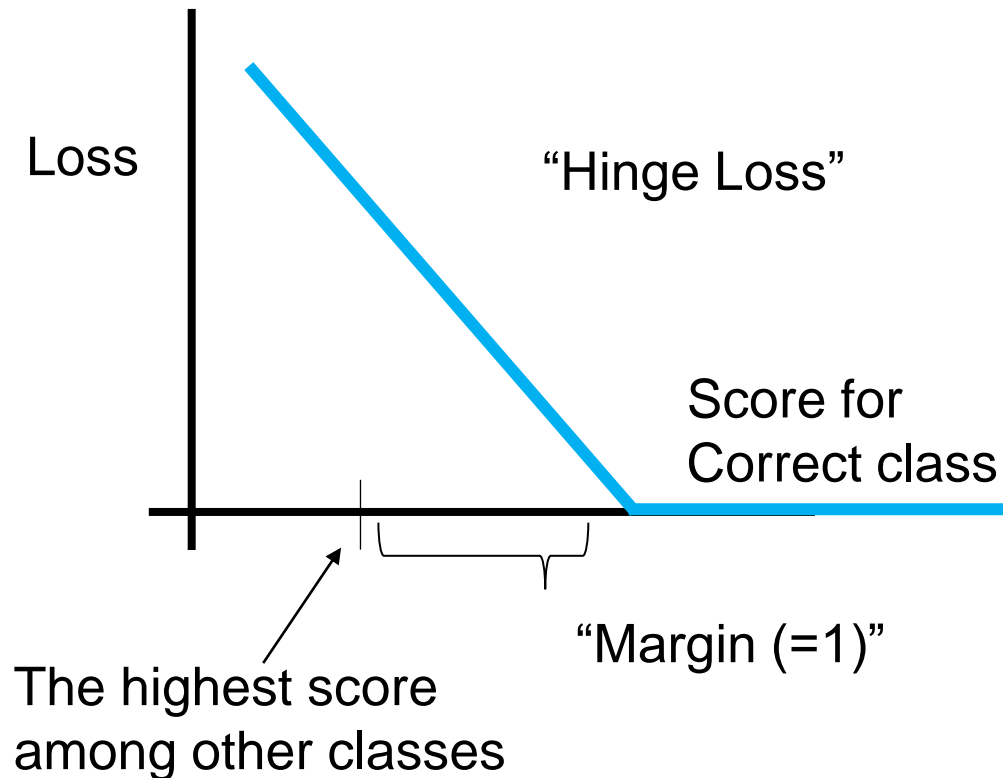
Hinge Loss

“The score of the correct class should be higher than all the other scores with at least a certain margin.”

- Given an example (x_i, y_i) , let $s = f(x_i, W)$ be class scores.

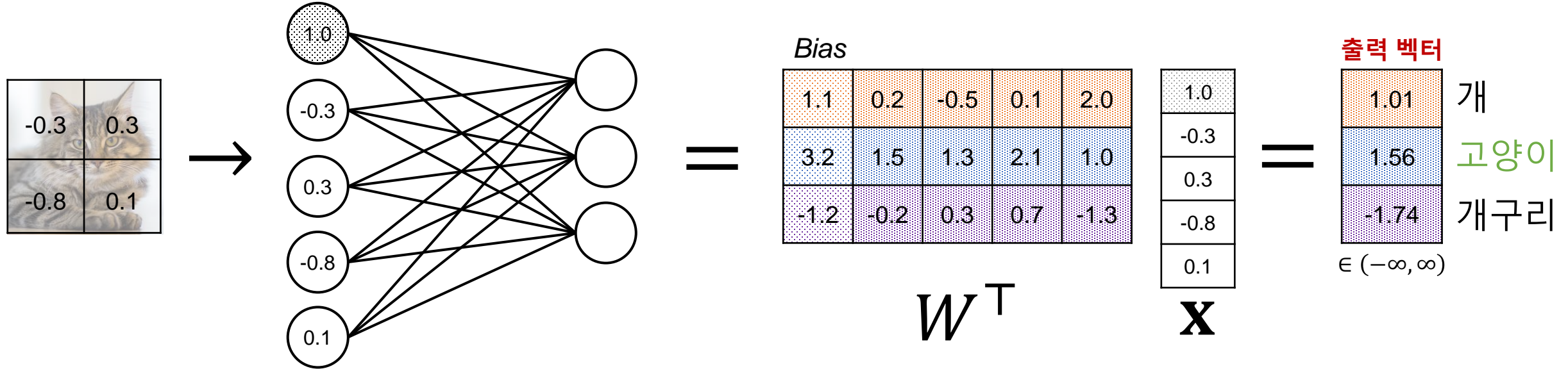
- The SVM loss has the form:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$



$$\text{Loss } 2.9 = \max(0, 5.1 - 3.2 + 1) + \max(0, -1.7 - 3.2 + 1)$$

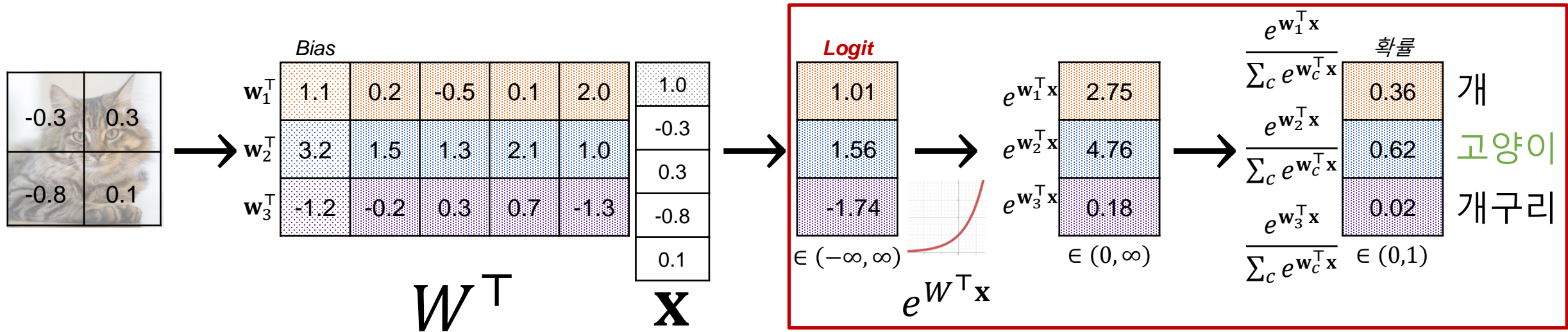
Softmax Layer for Multi-Class Classification



- 먼저 Linear layer를 두고, 출력 벡터의 Dimension을 Class 개수와 동일하게 설정
- 특정 Dimension의 값이 클 수록, 해당 Class에 부여되는 확률 값이 커지도록 함
- 출력 벡터를 Normalize하여 합이 1인 형태의 상대적인 확률 분포로 변환

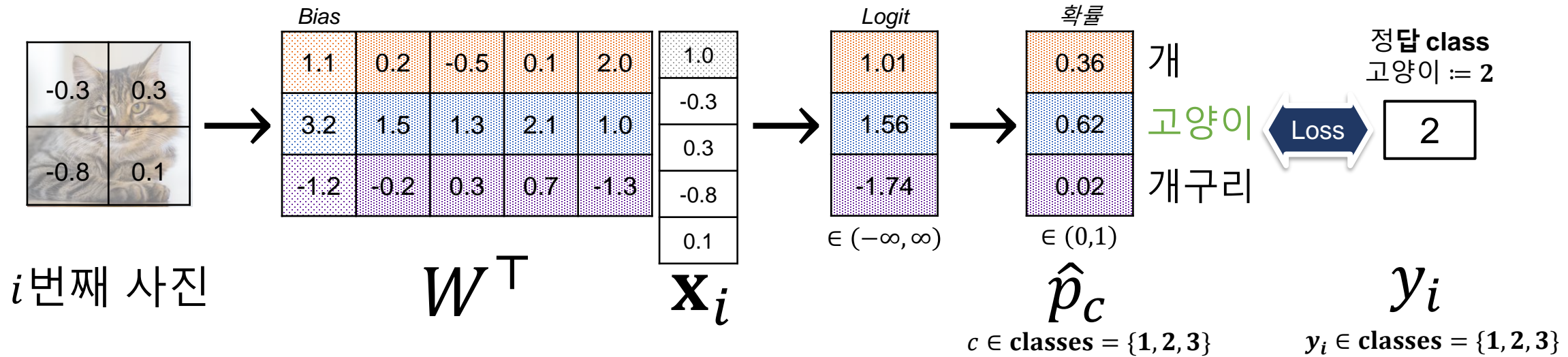
→ 이러한 layer를 **Softmax layer**라고 부름

Softmax Layer

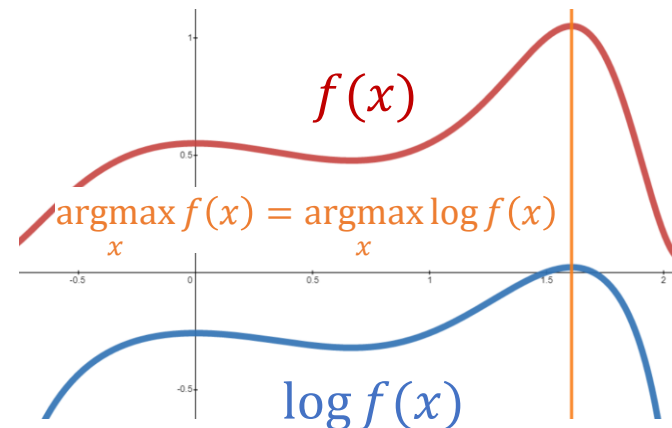
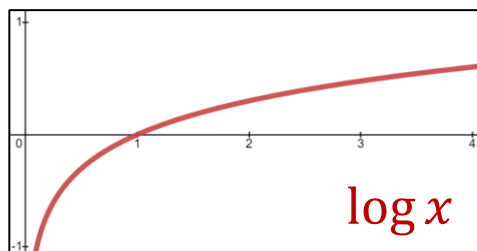


- **Logit vector:** Softmax의 입력 벡터이자, 직전 Linear layer의 출력 벡터
- **Logit vector**의 각각의 값에 단조 증가함수인 지수 함수를 적용:
 - $(-\infty, \infty)$ 사이의 Logit을 $(0, \infty)$ 사이의 양수 값으로 변환
 - 변환 후에도 크기 순서가 유지
- 해당 양수 값들의 합에 대한 각 값의 상대적인 크기를 계산
 - 각 Class에 대한 결과 값의 합이 1 \rightarrow 확률 분포로 해석할 수 있음

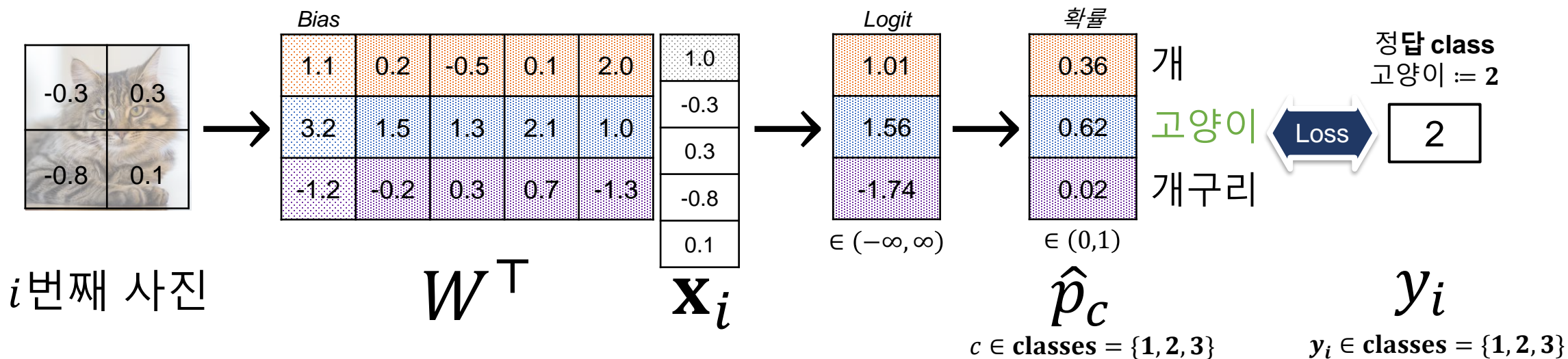
Softmax Classifier의 Loss Function



- 정답 Class에 대한 Likelihood \hat{p}_{y_i} 를 최대화
- 이는 Log-likelihood 최대화와 동등: $\log \hat{p}_{y_i}$
 - (0, 1) 사이의 값을 $(-\infty, 0)$ 사이로 변환
 - 로그 함수는 단조 증가 → 크기 순서 유지



Softmax Loss = Negative Log-Likelihood Loss



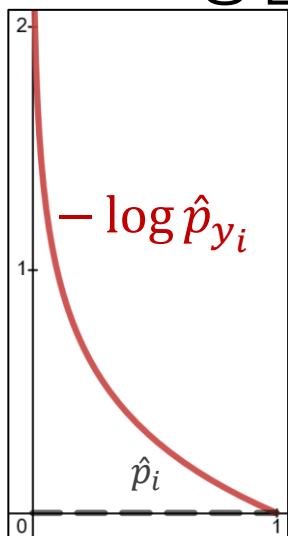
- 정답 Class에 대한 Likelihood \hat{p}_{y_i} 를 최대화하고자 하나,

우리는 일반적으로 최소화하고자 하는 “Loss”의 형태로 정의함

→ **Negative Log-Likelihood (NLL) loss**, 즉 $-\log \hat{p}_{y_i}$ 를 최소화

- (0, 1) 사이의 값을 (0, ∞) 사이로 변환
- 마이너스 로그 함수는 단조 감소 함수

→ 그리고, 이 Loss를 **Softmax loss**라고도 부름



Softmax Loss = Cross Entropy Loss

이산 확률 분포 P 와 Q 에 대해서, **Cross entropy**는 다음과 같음:

$$H(P, Q) = - \sum_{x \in X} P(x) \log Q(x)$$

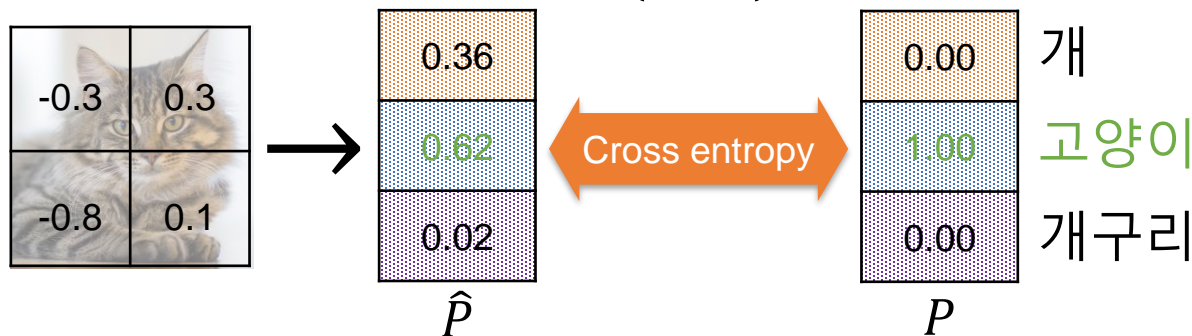
- 이 값은 두 분포 P 와 Q 가 얼마나 다른지를 측정
- **Cross entropy**가 Loss로 사용될 시, Ground-truth 확률 분포와 예측 확률 분포간 차이를 측정
- 일반적으로 Ground-truth 확률 분포를 P 로 두고, 예측 확률 분포 \hat{P} 를 Q 로 둠

Cross entropy에서 Ground-truth 분포를 One-hot vector로 계산 → **Softmax loss**와 동일

- One-hot vector: $P = [p_1, \dots, p_C]$ 의 정답 위치 p_{y_i} 에만 1을 넣고 다른 위치에는 0을 할당,

(예시: $[0, 1, 0]$)

$$H(P, Q) = H(P, \hat{P}) = - \sum_{c=1}^C p_c \log \hat{p}_c = -\log(\hat{p}_{y_i})$$



KL Divergence vs. Cross Entropy

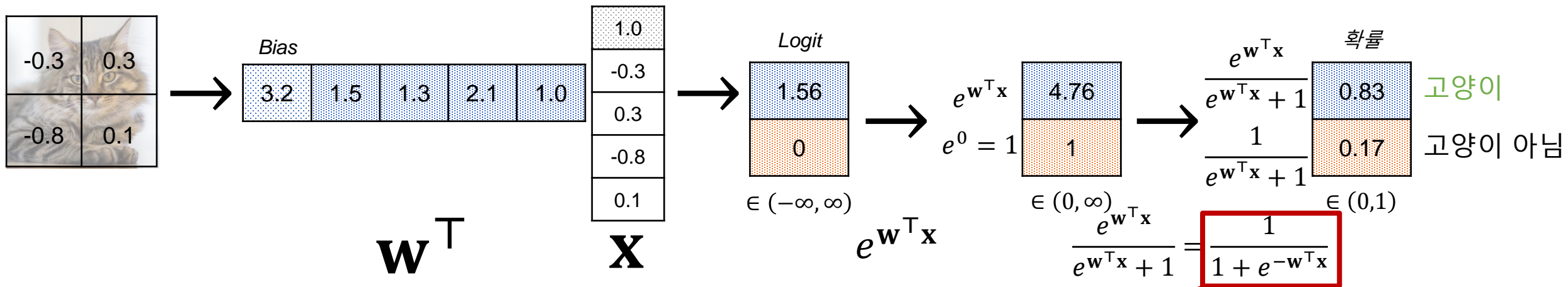
Cross entropy는 KL divergence로도 표현 가능

$$D_{KL}(P \parallel Q) = - \sum_{x \in X} P(x) \log \left(\frac{Q(x)}{P(x)} \right) = - \sum_{x \in X} P(x) \log Q(x) + \sum_{x \in X} P(x) \log P(x) = H(P, Q) - H(P)$$

Cross entropy와 KL divergence의 차이는 추가적인 scalar 값 $-H(P)$ 존재 유무

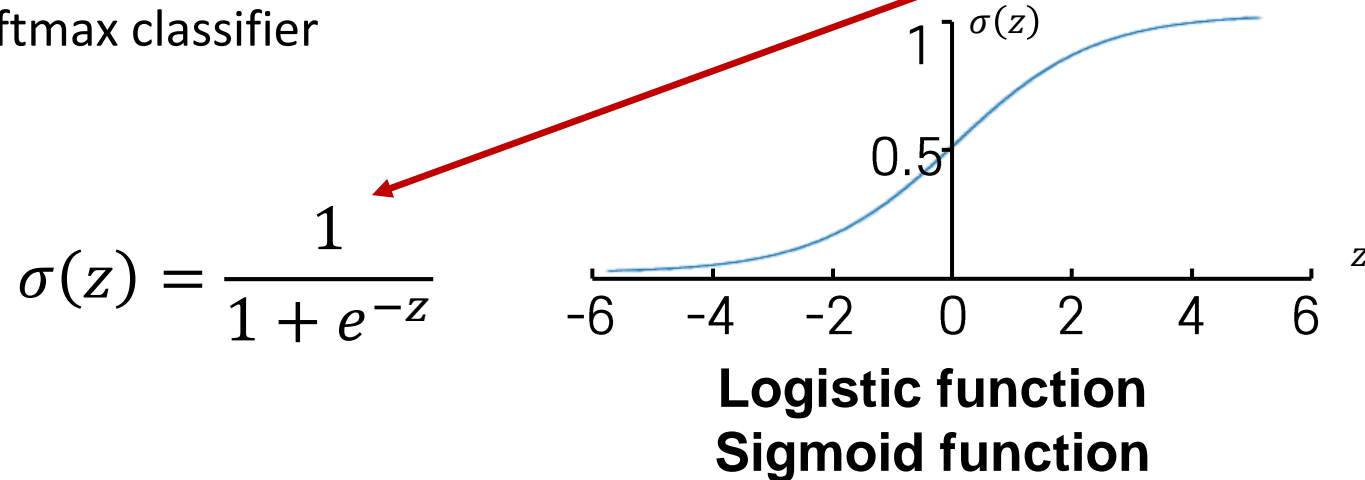
- Ground-truth 확률 분포 P 의 Entropy인 $H(P)$ 는 상수 값
→ 최적화 과정에서는 영향을 주지 않음
- 또한, P 가 One-hot vector인 경우, $H(P) = 0$
따라서, Cross entropy 최소화 = KL divergence 최소화
→ Cross entropy 대신 KL divergence도 Loss로 사용 가능
- Ground-truth 확률 분포 P 가 Dense vector인 경우도 존재
 - e.g., Knowledge distillation
 - Dense vector 예시: [0.2, 0.4, 0.1, 0.3] (vs. one-hot vector [0, 1, 0, 0])

Logistic Regression은 Softmax Classifier의 Special Case

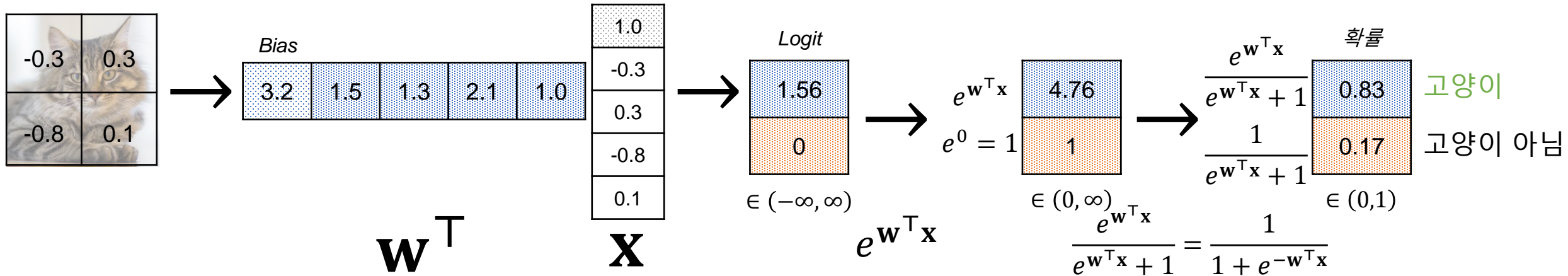


Logistic regression

→ Binary classification 문제에서 Positive class가 아닌 나머지 Class의 Logit을 상수 0으로 고정 한 형태의 Softmax classifier



Sigmoid Function



Logistic regression

→ Binary classification 문제에서 Positive class가 아닌 나머지 Class의 Logit을 상수 0으로 고정한 형태의 Softmax classifier

- 2열 크기 행렬 W 로 두 Class에 대한 Softmax classification 으로도 가능
 - 이 경우 Logistic regression보다 두 배 많은 수의 Parameter를 가짐

Binary Cross Entropy for Logistic Regression

Logistic Regression의 Binary cross entropy loss

➔ Softmax classifier의 Cross entropy에서 Class가 두 개일 때와 동일함

- Cross Entropy

$$\mathcal{L} = - \sum_{c=1}^C y_c \log(p_c)$$

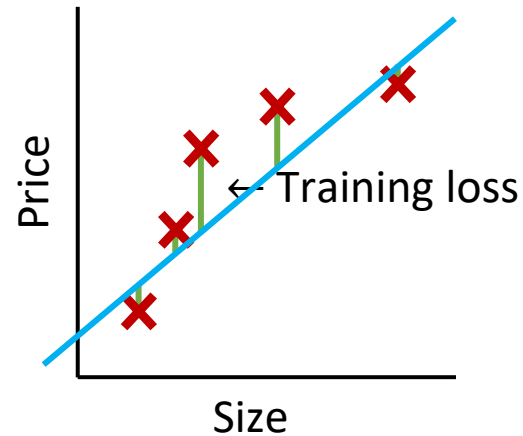
- Binary Cross Entropy (BCE)

$$\mathcal{L} = - \sum_{c=1}^2 y_c \log(p_c) = -y_1 \log(p_1) - y_2 \log(p_2) = -y_1 \log(p_1) - (1 - y_1) \log(1 - p_1)$$
$$\because y_2 = 1 - y_1, p_2 = 1 - p_1$$

Overfitting / Underfitting, Regularization

An Example of Linear Regression

- An example of predicting housing prices

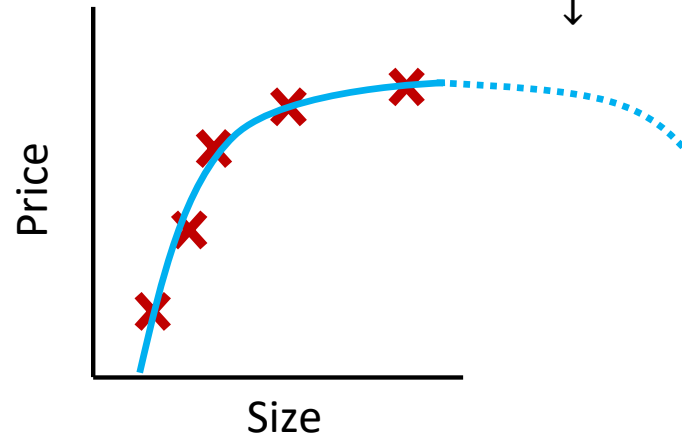


Function

$$\theta_0 + \theta_1 x$$

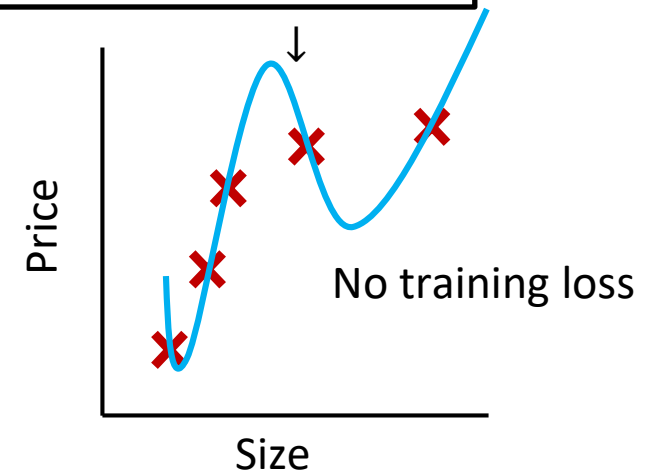
Model
Evaluation

Underfitting,
High bias



$$\theta_0 + \theta_1 x + \theta_2 x^2$$

Just right



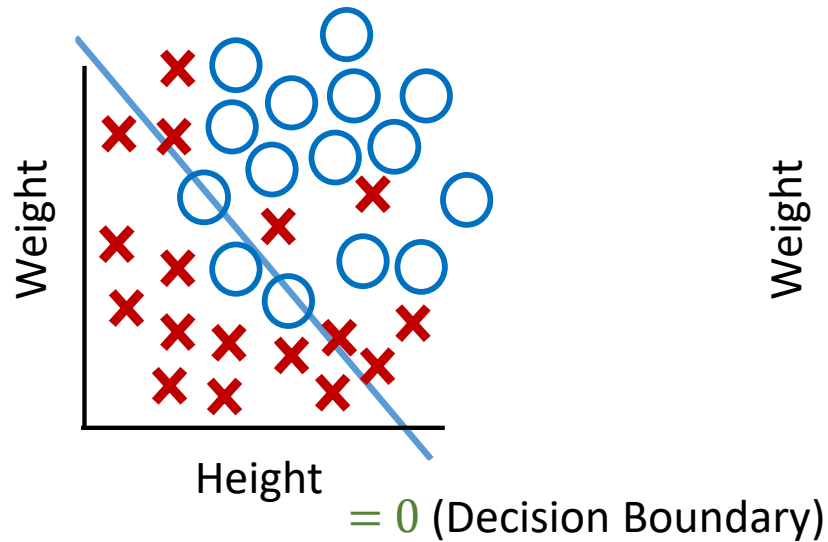
$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

Overfitting,
High variance

It is incorrectly predicted that the housing price will go down as the size increases.

An Example of Logistic Regression

- An example of classifying people with diabetes

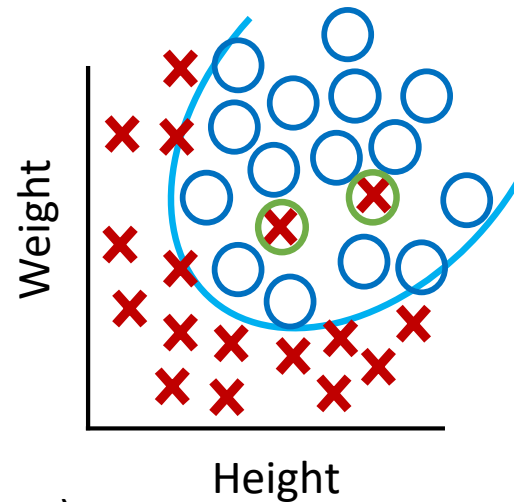


Function

$$h_{\theta}(\mathbf{x}) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

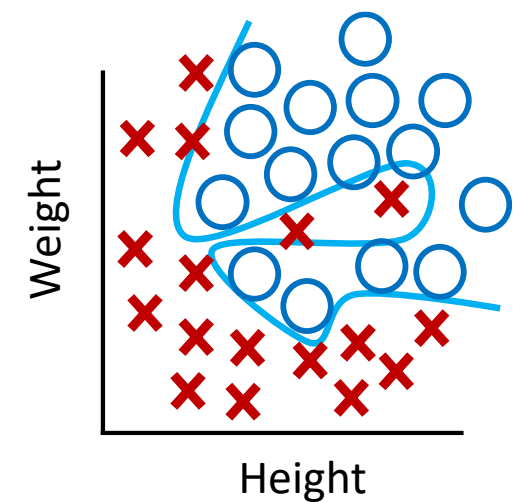
(g = sigmoid function)

Underfitting



$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_2^2 + \theta_5 x_1 x_2)$$

Just Right



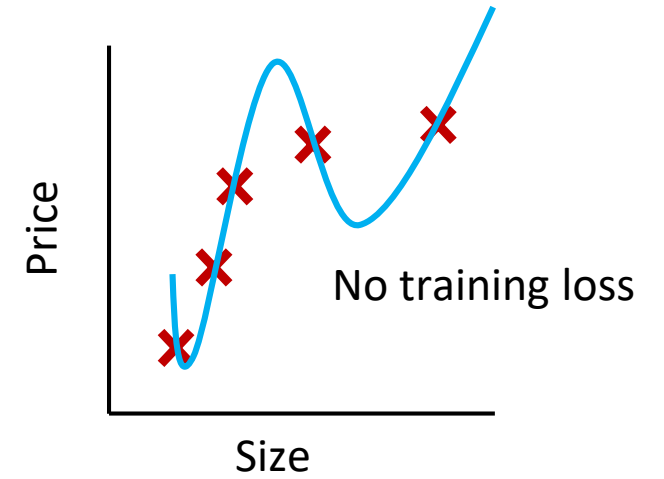
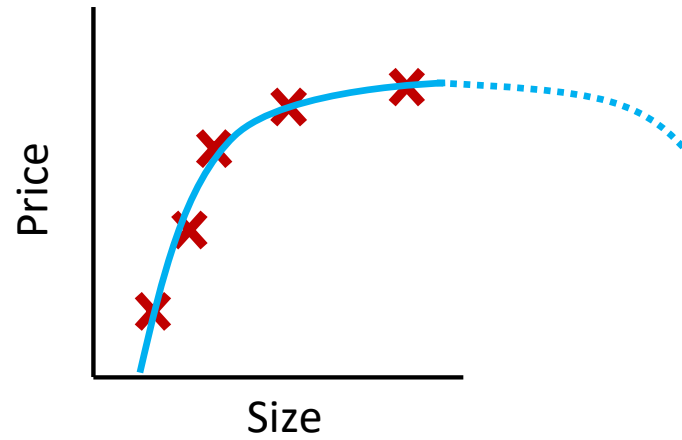
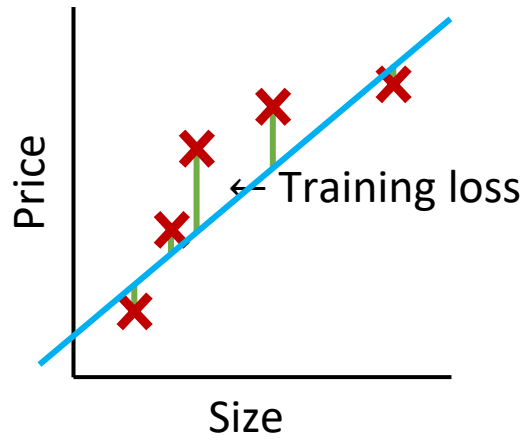
$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \theta_6 x_1^3 x_2 + \dots)$$

Overfitting

Model
Evaluation

Bias Variance Trade Off

- An example of predicting housing prices



Function

$$\theta_0 + \theta_1 x$$

$$\theta_0 + \theta_1 x + \theta_2 x^2$$

$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

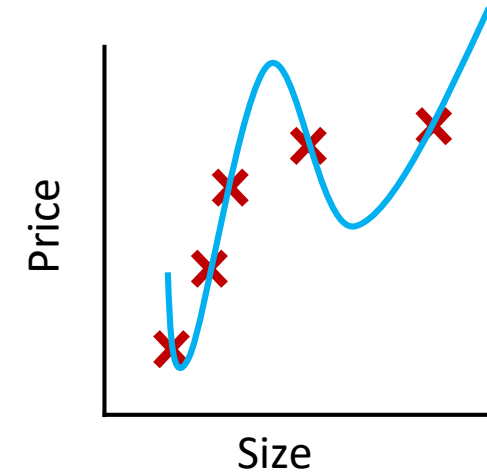
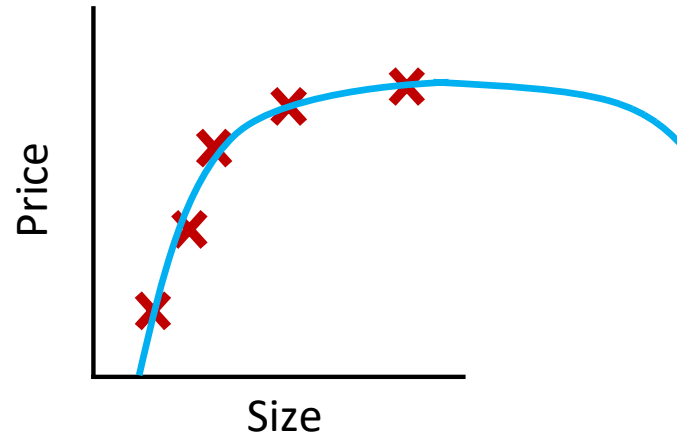
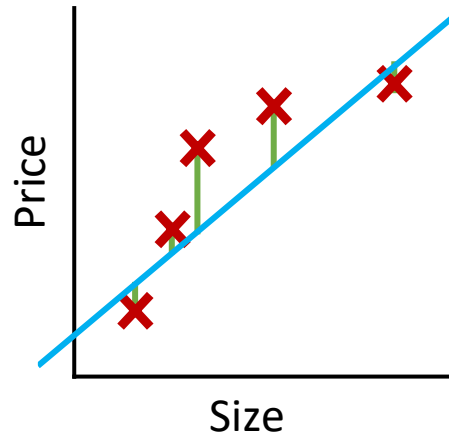
Overfitting: If we have too many features, the learned hypothesis may fit the training set very well

$$(\bar{J}(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=1}^m \left(h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)} \right)^2 \approx 0),$$

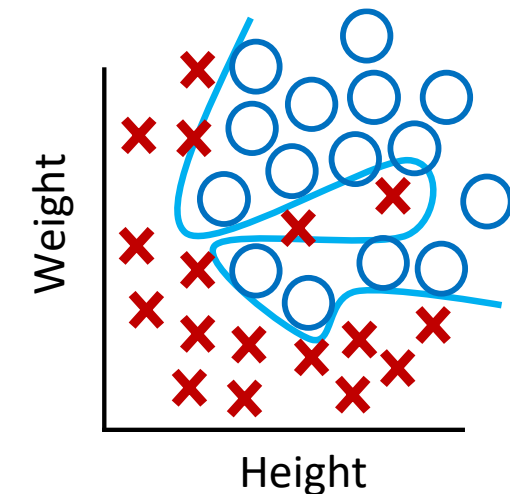
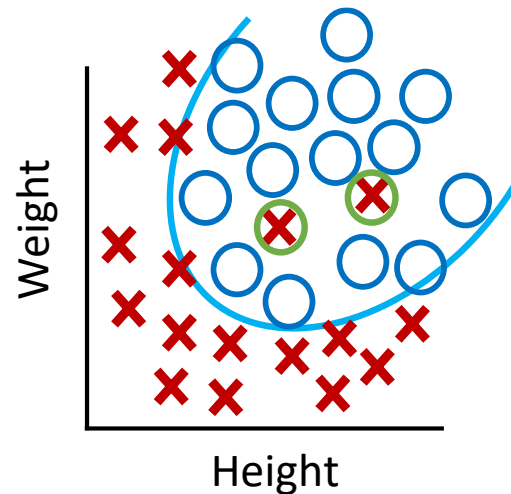
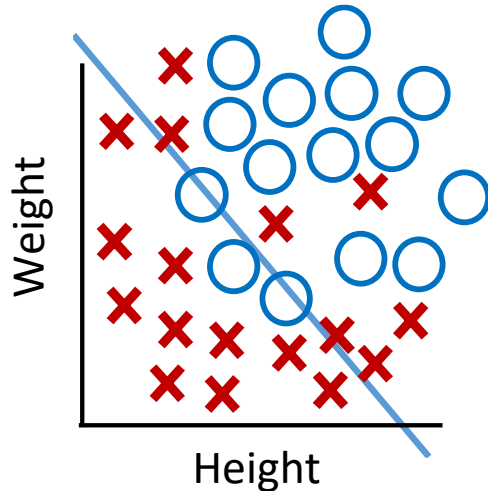
but fail to generalize to new examples (predict prices on new examples)

The Best Model for Testing Unseen Data

- An example of predicting housing prices



- An example of classifying people with diabetes



When testing unseen data, the second models will perform best.

Addressing Overfitting (1) Feature Reduction

- An example of predicting housing prices

$x_1 = \text{size of house}$

$x_2 = \text{no. of bedrooms}$

$x_3 = \text{no. of floors}$

$x_4 = \text{age of house}$

$x_5 = \text{average income in neighborhood}$

$x_6 = \text{kitchen size}$

$x_7 = \text{size of bricks}$

\vdots

x_{100}



These two variables may be overlapping variables with high correlation.

This variable may be a variable unrelated to the prediction of housing prices.

How to Solve Overfitting

Options:

1. Reduce number of features
 - Manually select which features to keep.
 - Model selection algorithm (later in course).
2. Regularization
 - Keep all the features, but reduce magnitude/values of parameters θ_j
 - Works well when we have a lot of features, each of which contributes a bit to predicting y

Regularization Formula

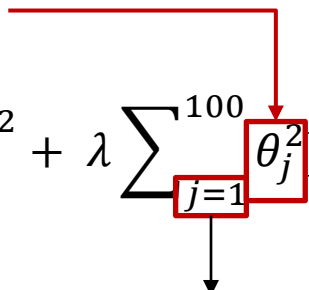
- Small values for parameters $\theta_0, \theta_1, \dots, \theta_n$

- “Simpler” hypothesis
- Less prone to overfitting

- Housing:

- Features: x_1, x_2, \dots, x_{100}

- Parameters: $\theta_0, \theta_1, \theta_2, \dots, \theta_{100}$

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \left[\sum_{i=1}^m (h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{100} \theta_j^2 \right]$$


Regularization parameter It does not apply to constant term

λ Value Selection for Regularization

- In regularized linear regression, we choose θ to minimize

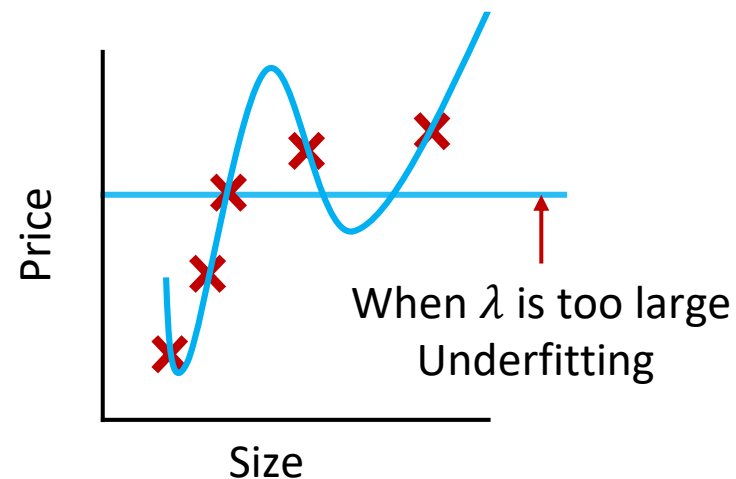
$$J(\boldsymbol{\theta}) = \frac{1}{2m} \left[\sum_{i=1}^m (h_{\boldsymbol{\theta}}(\mathbf{x}^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^n \theta_j^2 \right]$$

- What if λ is set to an extremely large value (perhaps far too large for our problem, say $\lambda = 10^{10}$)?

- Algorithm works fine; setting λ to be very large can't hurt it.
- Algorithm fails to eliminate overfitting.
- Algorithm results in underfitting. (Fails to fit even training data well).
- Gradient descent will fail to converge.

e.g., $\lambda = 10000$?

Regularization term: $10000(\theta_1^2 + \theta_2^2 + \theta_3^2 + \theta_4^2)$



Addressing Overfitting (2) Regularization

- An example of predicting housing prices

$x_1 = \text{size of house}$

$x_2 = \text{no. of bedrooms}$

$x_3 = \text{no. of floors}$

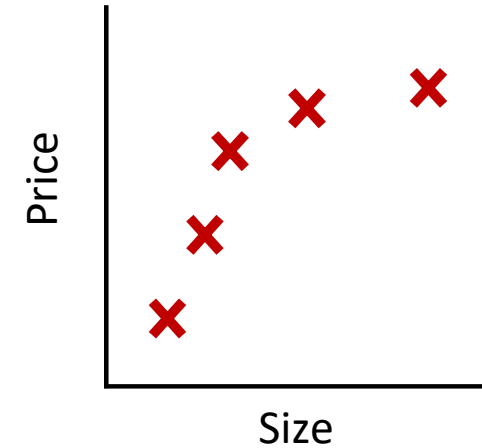
$x_4 = \text{age of house}$

$x_5 = \text{average income in neighborhood}$

$x_6 = \text{kitchen size}$

$x_7 = \text{size of bricks}$

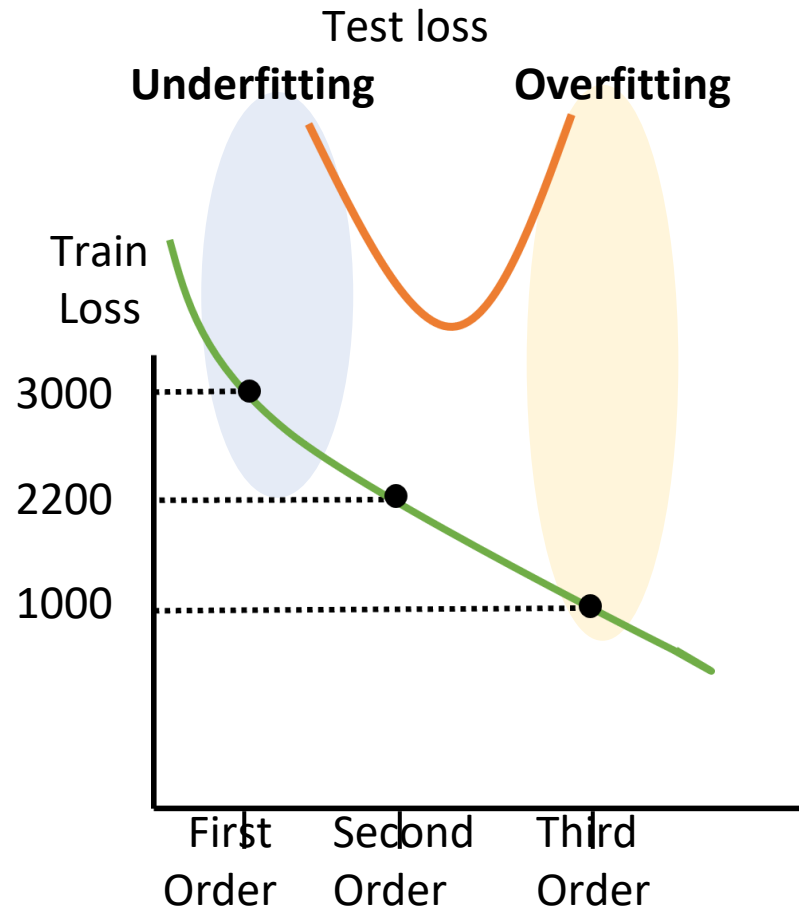
\vdots



If reducing the number of variables, we can adopt **feature reduction method**

If not reducing the number of variables, we can apply **regularization (increase λ) method**

Model Training Process While λ Adjustment



$\lambda = 1000$ $\lambda = 10$ $\lambda = 0.1$

If the number of data is 100,

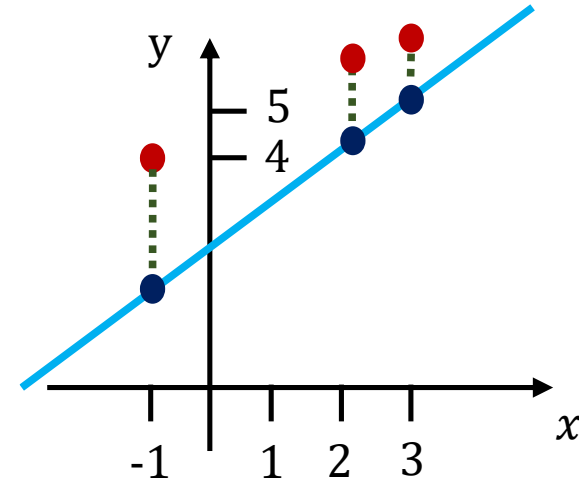
1. Data split

Train set: Test data = 70 : 30

2. Comparison between train loss and test loss while λ adjustment

An Example of Gradient Descent in Linear Regression

x	y	$h_{\theta}(x)$	$h_{\theta}(x) - y$
2	5	4	-1
3	7	5	-2
-1	4	1	-3



1. First-order linear regression

$$h_{\theta}(\mathbf{x}) = \theta_0 + \theta_1 x$$

2. To minimize θ ,

$$J(\theta_0, \theta_1) = \frac{1}{2m} \left[\sum_{i=1}^m (h_{\theta}(\mathbf{x}^{(i)}) - y^{(i)})^2 + \frac{10 \cdot \theta_1^2}{\lambda} \right]$$

When differentiating ($\theta_1 = 1$),

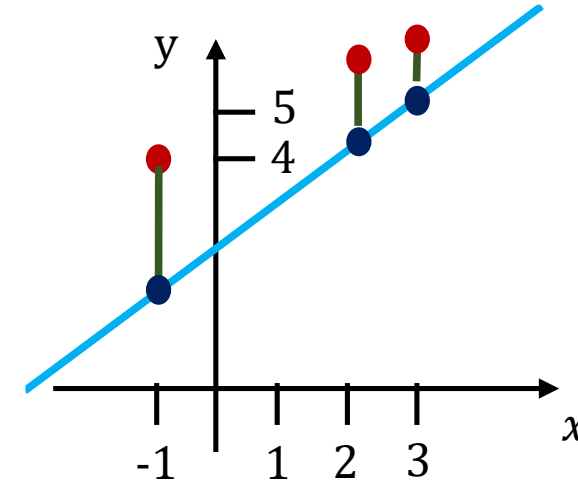
$\theta_1 > 0$: Gradient ($= 10 \cdot 2\theta_1 = 20$) > 0

$\theta_1 < 0$: Gradient ($= 10 \cdot 2\theta_1 = 20$) < 0
→ Close to zero in the end

An Example of Gradient Descent in Linear Regression

x	y	$h_{\theta}(x)$	$h_{\theta}(x) - y$
2	5	4	-1
3	7	5	-2
-1	4	1	-3

When $\theta_0 = 2$, $\theta_1 = 1$, $\lambda = 0.1$,



$$J(\theta_0, \theta_1) = \frac{1}{2 \cdot 3} ((\theta_0 + \theta_1 \cdot 2 - 5)^2 + (\theta_0 + \theta_1 \cdot 3 - 7)^2 + (\theta_0 + \theta_1 \cdot (-1) - 4)^2)$$

$$= \frac{1}{2 \cdot 3} ((-1)^2 + (-2)^2 + (-3)^2) = \frac{14}{6}$$

$$\frac{dJ(\theta_0, \theta_1)}{d\theta_0} = \frac{1}{3} ((-1) + (-2) + (-3)) = -2$$

$$\frac{dJ(\theta_0, \theta_1)}{d\theta_1} = \frac{1}{3} ((-1) \cdot 2 + (-2) \cdot 3 + (-3) \cdot (-1)) = -\frac{5}{3}$$

- Step size: 0.1

$$\theta_0 \leftarrow 2 - 0.1 \cdot (-2) = 2.2$$

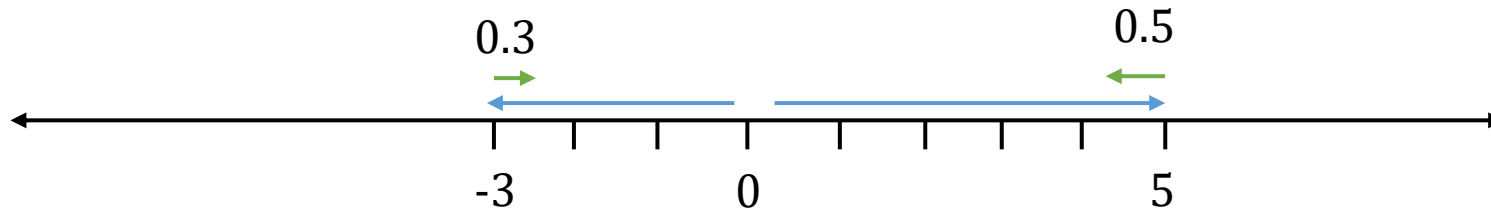
$$\theta_1 \leftarrow 1 - 0.1 \cdot (-5/3) \cong 1.167$$

$$(1 - 0.1 \cdot 0.2)$$

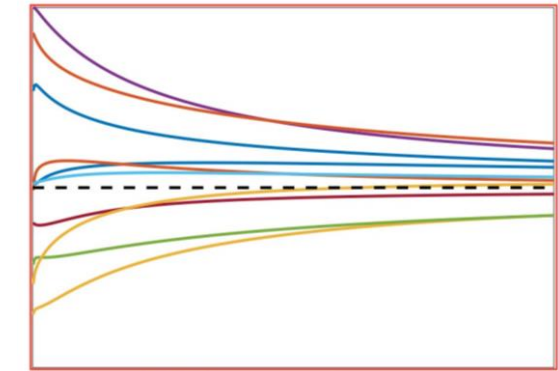
$$\lambda \cdot 2\theta_1$$

Gradient Descent with L1, L2 Regularization

- L2 regularization : $\lambda \cdot \theta_1^2 \rightarrow \lambda \cdot 2\theta_1$, Approaching zero but not converging to zero

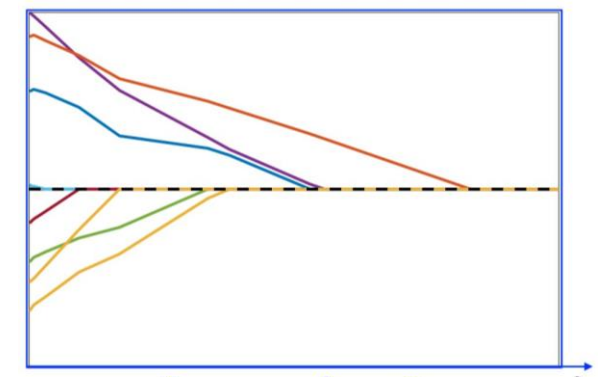
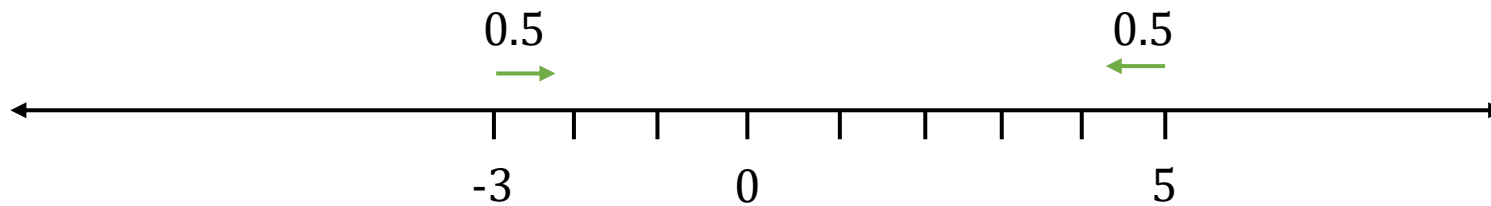


L2 regularization is similar to the nature of returning as the spring has increased.



Ridge $\theta = 1$

- L1 regularization : $\lambda \cdot |\theta_1| \rightarrow \lambda$ (constant), Eventually converging to zero (Feature selection effect)



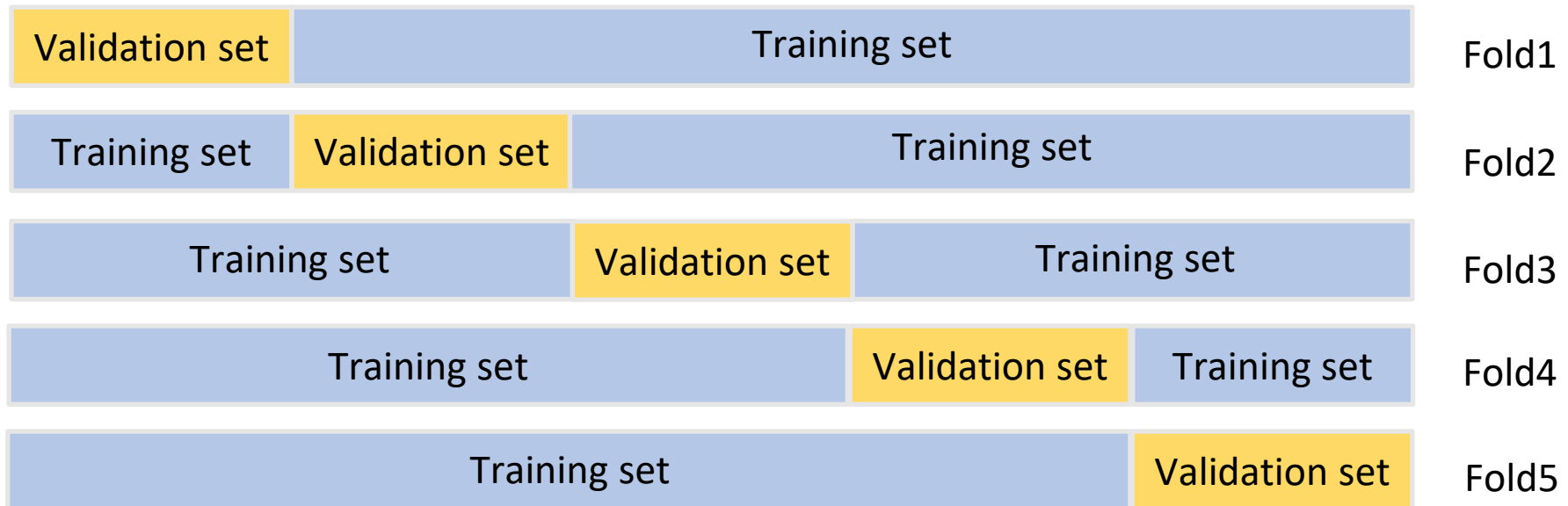
Lasso $\theta = 0$

Data Split

K-fold Cross Validation

- Divide data into K folds equally
- Use each fold as validation set one by one

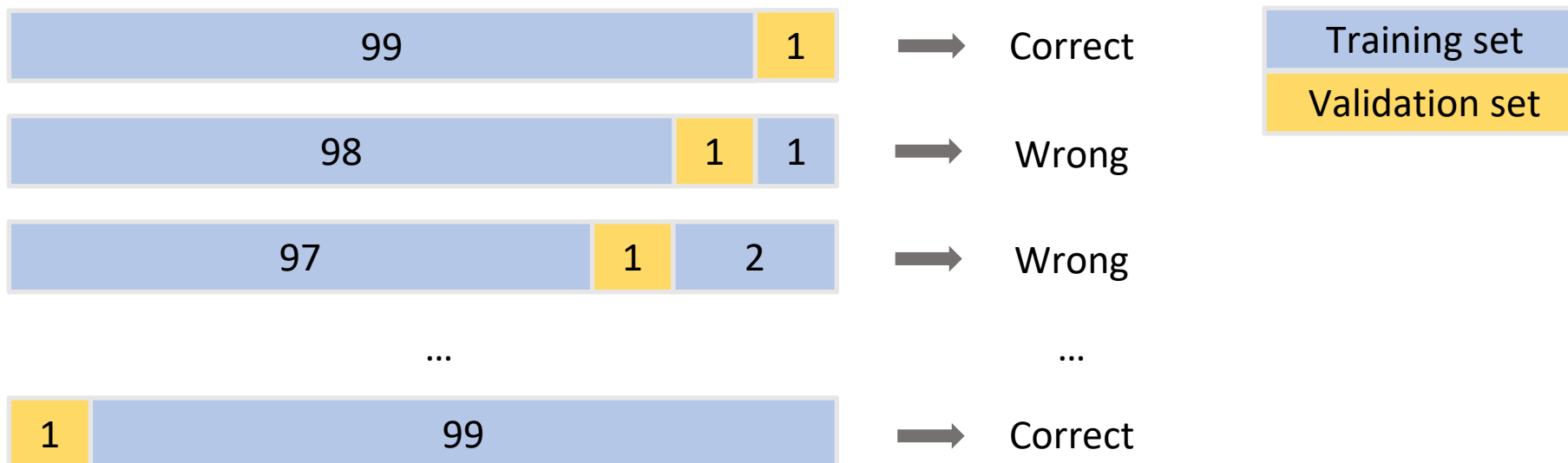
5-fold Cross Validation



Leave-One-Out Cross Validation (LOOCV)

- Train the model using sufficiently large data set
- Validate the model for all data

Leave-One-Out Cross Validation (LOOCV)



Train, Validation, Test data

- Divide data into 3 parts - Train, Validation, Test data
- Test data should not be used until test time
- Can be expanded to Cross-Validation

