

3. Loss Function & Optimization

모델이 어떻게 학습하는지

2024.10.02/전지수

Contents

목차

- 0. Last time
- 1. Loss Function
 - SVM Loss
 - Cross Entropy Loss
- 2. Regularization
- 3. Optimization

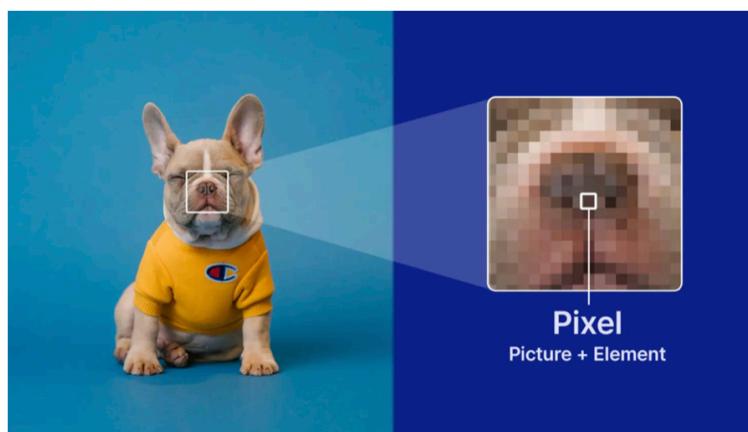
0.Last time : Image classification

Review

2. 이미지 데이터의 구조

디지털 이미지의 구조

- 피셀(Pixel)
- 채널(Channel)
- 해상도(Resolution)



픽셀

- 컴퓨터가 이미지 데이터를 인식할 때
0~255 값으로 픽셀을 표현

채널

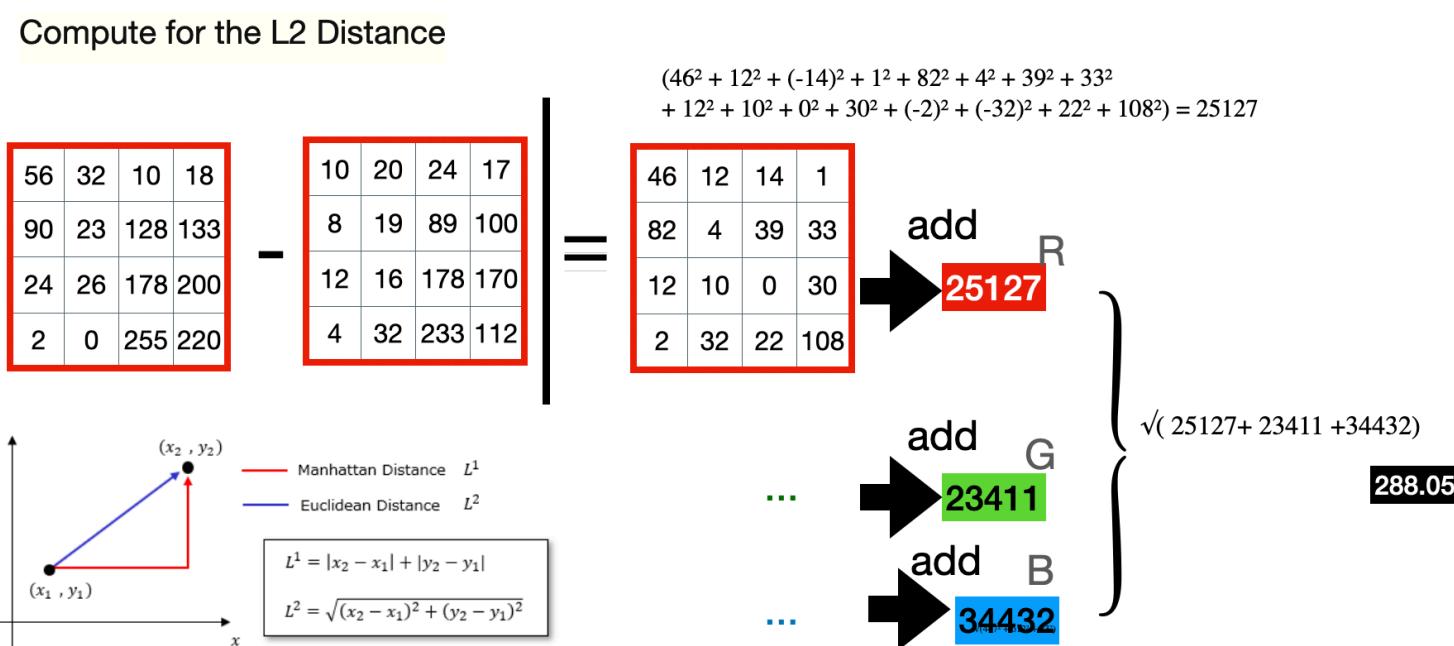
- 흑백 이미지 : 1채널
- 칼러 이미지 : 3채널 (R,G,B)
ex. (255,0,0)

해상도

- 이미지의 크기 (가로x세로) ex.(14x13)

3. 기본적인 이미지 분류 접근 방식

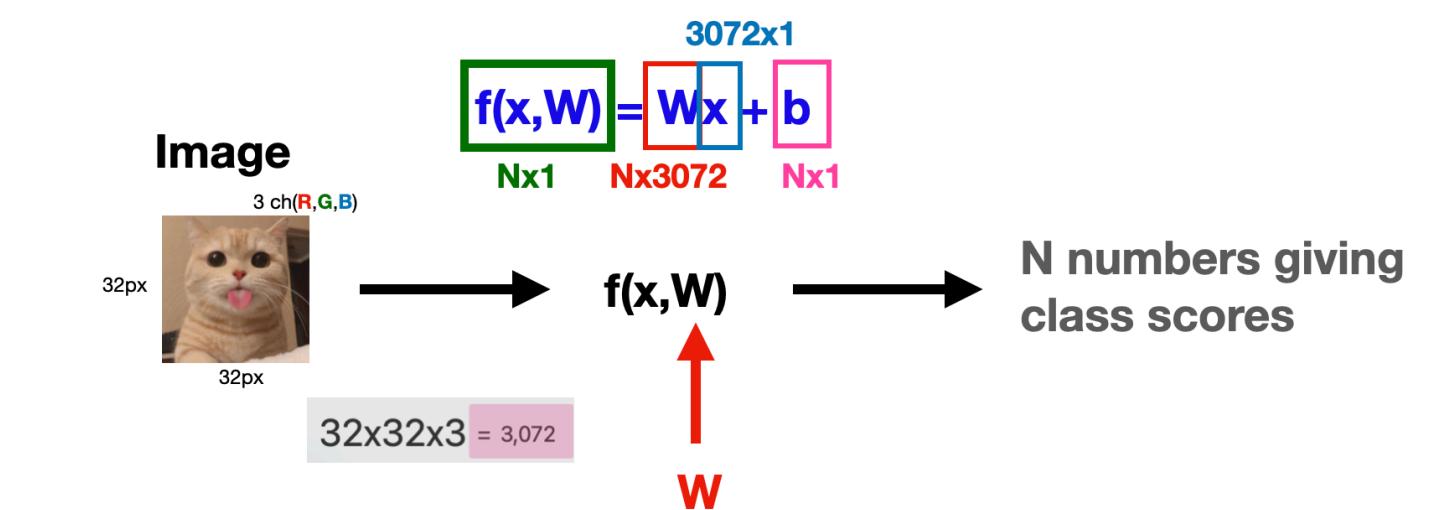
두 이미지가 얼마나 차이가 있는지 측정 방법(L1, L2)



4. 선형분류기

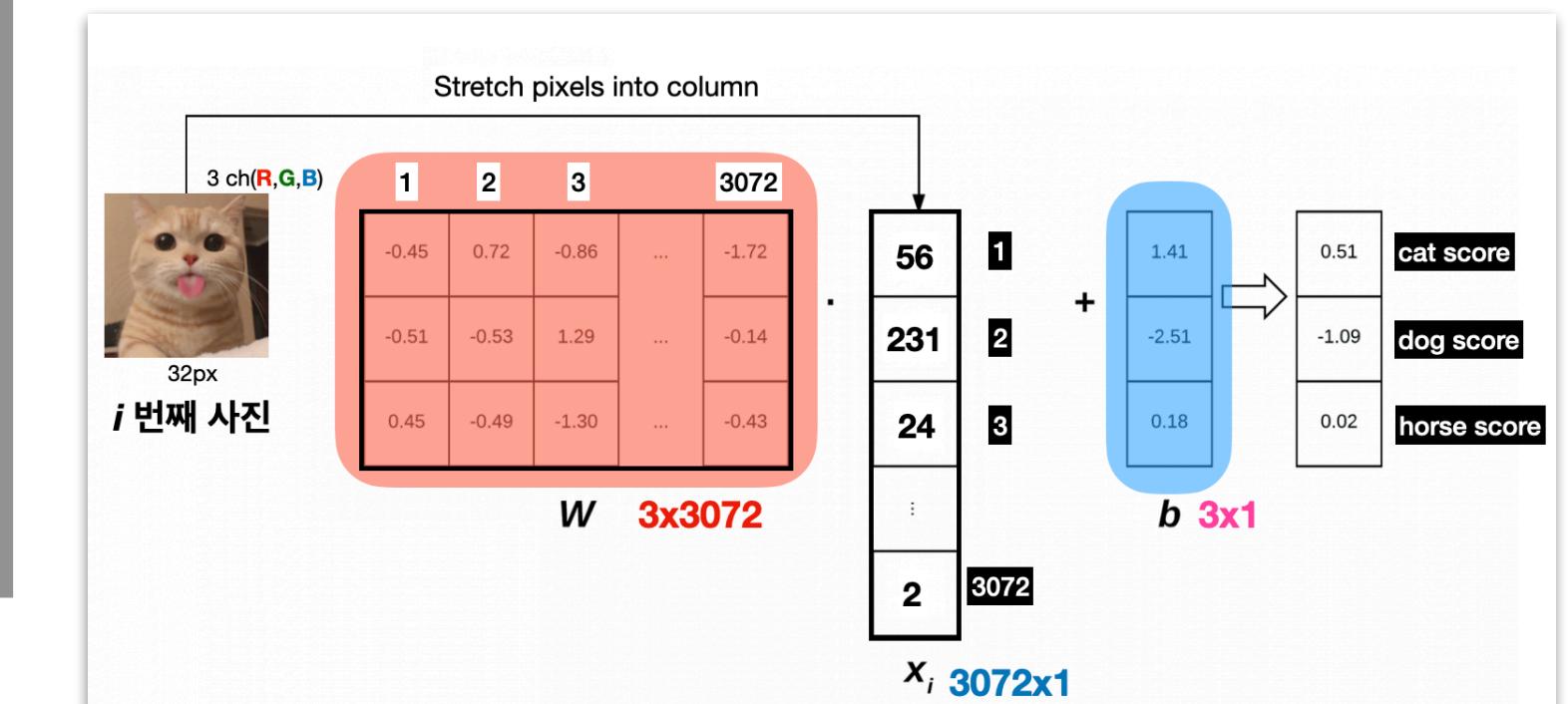
파라미터적인 접근?

- 선형분류기 (Linear Classifier)



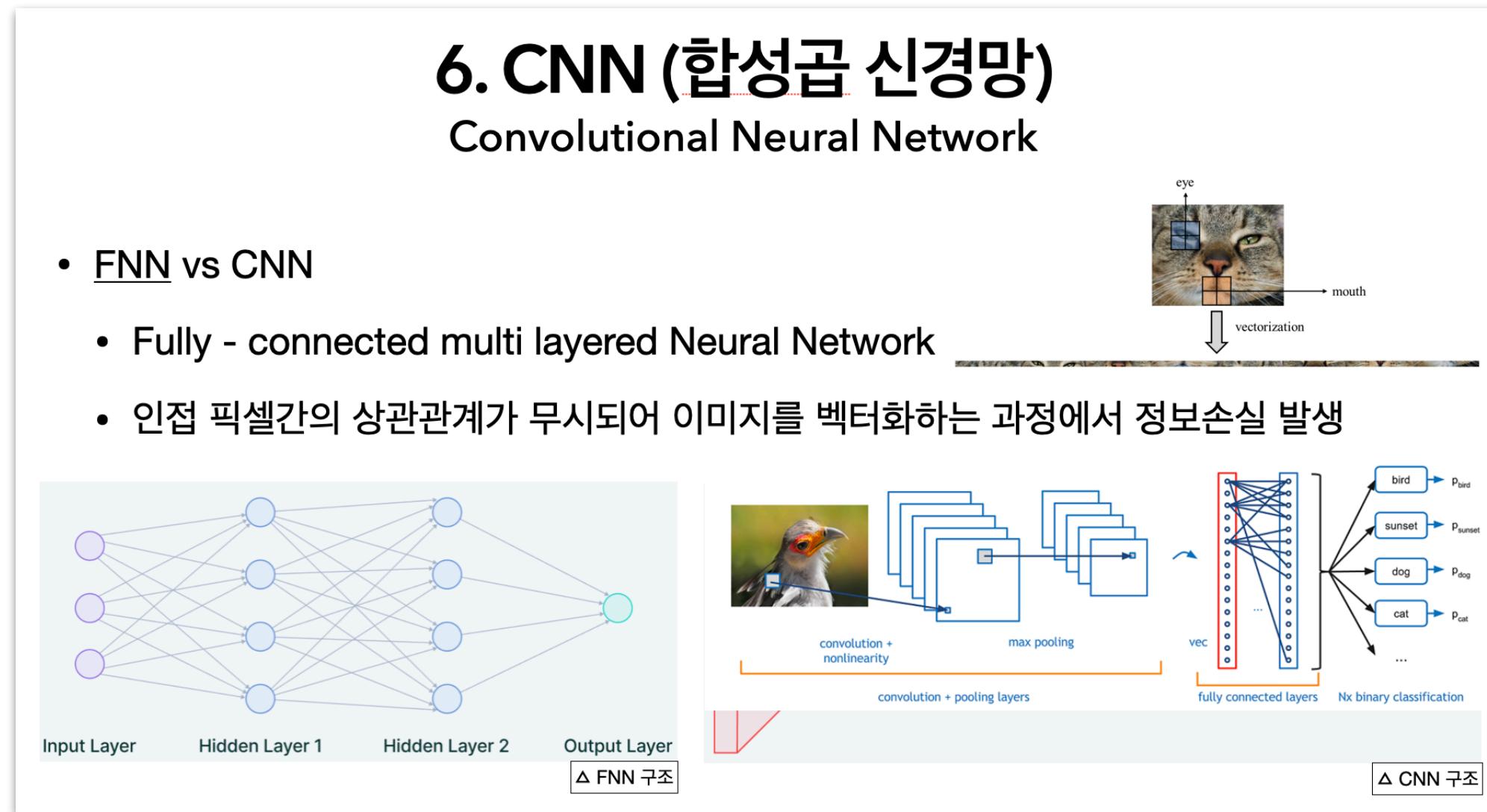
선형분류기

- 32x32x3 pixel 이미지를 가중치 W,b만
으로 총 N개의 class로 분류할 수 있는
Linear Classifier도 존재



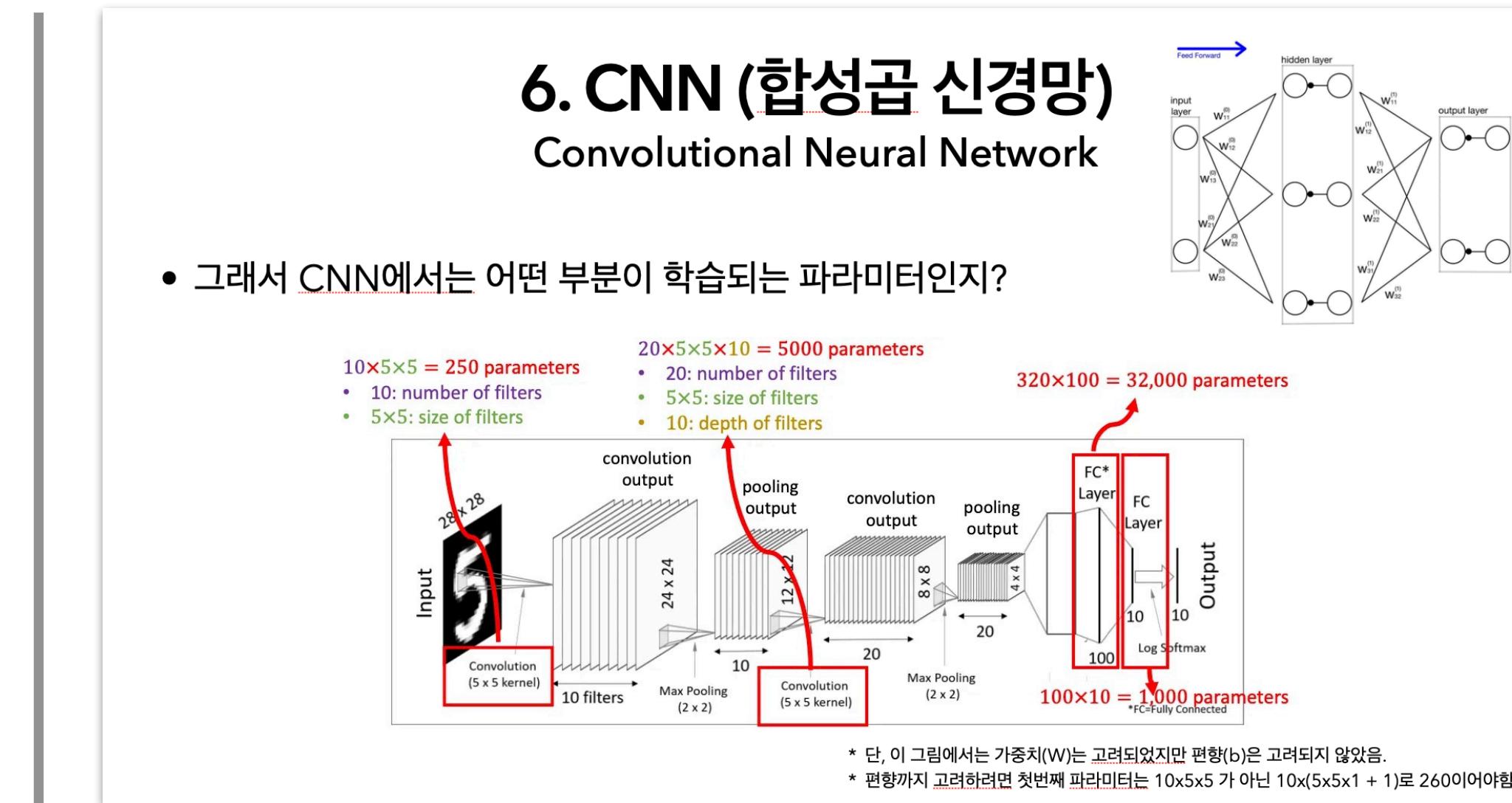
0.Last time : Image classification

Review



선형분류기의 한계와 CNN의 등장

- FNN처럼 이미지를 단순하게 벡터화시키면 정보 손실 발생 문제로 CNN 등장
- 커널로 공간적 정보 보존 가능
- CNN에서는 마지막에 FNN을 사용

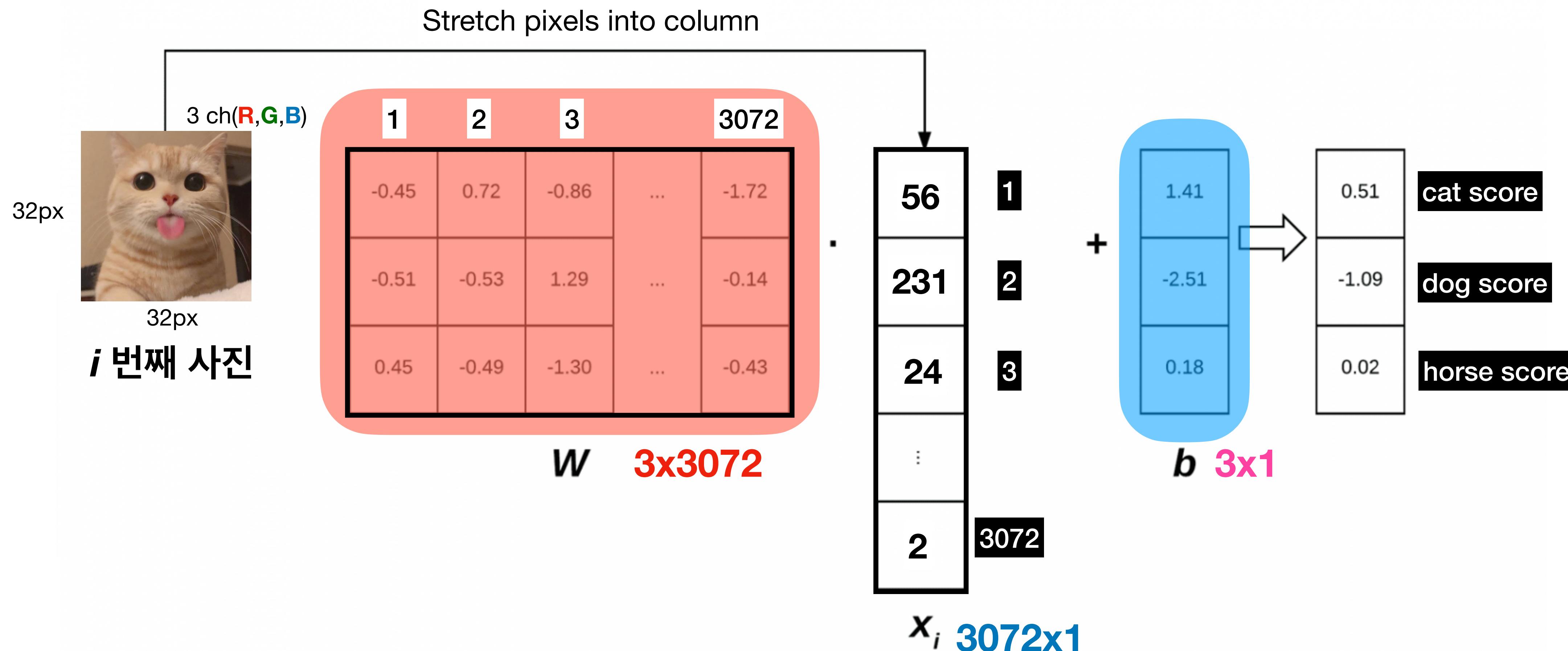


CNN에서의 파라미터 업데이트 (=학습)

- 커널(=필터) K개
- Bias
- FNN에서의 파라미터

1. Loss Function

Weight matrix 를 어떤 값으로 정해야 할까?



1. Loss Function

Weight matrix 를 어떤 값으로 정해야 할까?

- Weight matrix의 역할
 - 목적 : 입력 이미지를 정확하게 분류

입력 이미지	기대 결과
고양이 사진	Cat score가 최댓값
강아지 사진	Dog score가 최댓값
말 사진	Horse score가 최댓값

- Weight Matrix 최적화 방법
 - Loss Function 사용
 - 현재 Weight Matrix의 예측과 실제 정답 간의 차이(오차의 정도)를 수치화
 - 측정 결과를 바탕으로 Weight Matrix 값 업데이트
 - 목적 : Loss 최소화 = 분류 정확도 최대화

1. Loss Function

Weight matrix 를 어떤 값으로 정해야 할까?

- Loss Function 사용 : 현재 Weight matrix의 예측과 실제 정답 간의 차이를 수치화
 - 가중치 업데이트 방향 제시

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

일반적인 손실함수형태

1. Loss Function

Weight matrix 를 어떤 값으로 정해야 할까?

- 문제의 특성과 데이터 유형에 따라 적합한 Loss Function 선택
 - 회귀 문제에서 사용되는 Loss
 - MSE : Mean Squared Error
 - 분류 문제에서 사용되는 Loss
 - **SVM** Loss : Support Vector Machine에서 사용하는 hinge loss
 - Cross-Entropy Loss

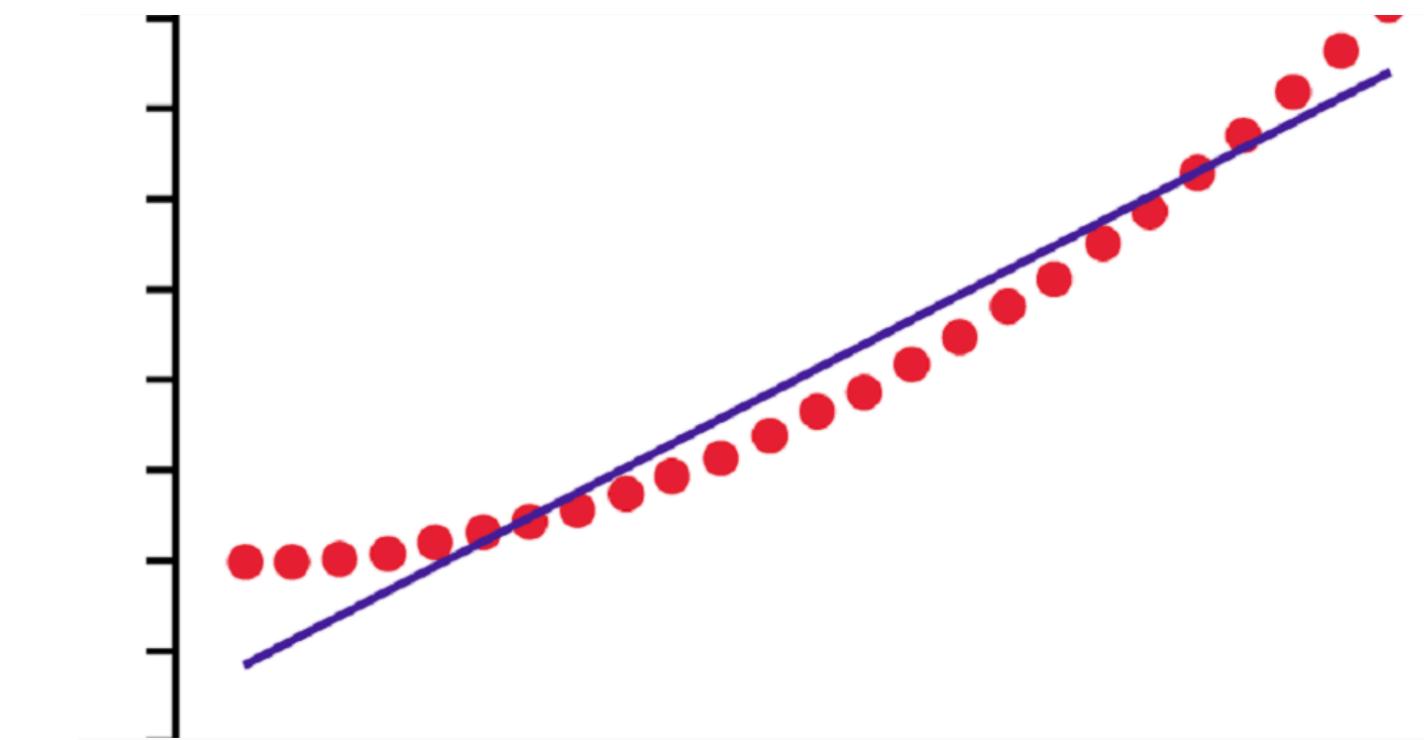
1. Loss Function

Mean Squared Error

- 연속형 변수를 예측할 때 사용

$$MSE = \frac{1}{n} \sum \left(\underbrace{y - \hat{y}}_{\text{The square of the difference between actual and predicted}} \right)^2$$

The square of the difference
between actual and
predicted



1. Loss Function

Support Vector Machine의 Hinge Loss

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$
$$= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$$

Multiclass SVM loss:

Given an example (x_i, y_i) where x_i is the image and where y_i is the (integer) label,

and using the shorthand for the scores vector: $s = f(x_i, W)$



i = 1 i=2 i=3

j = cat	cat	3.2	1.3	2.2
j = car	car	5.1	4.9	2.5
j = frog	frog	-1.7	2.0	-3.1
Losses:		2.9		

the SVM loss has the form:

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

$$\begin{aligned} &= \max(0, 5.1 - 3.2 + 1) \\ &\quad + \max(0, -1.7 - 3.2 + 1) \\ &= \max(0, 2.9) + \max(0, -3.9) \\ &= 2.9 + 0 \\ &= 2.9 \end{aligned}$$

1. Loss Function

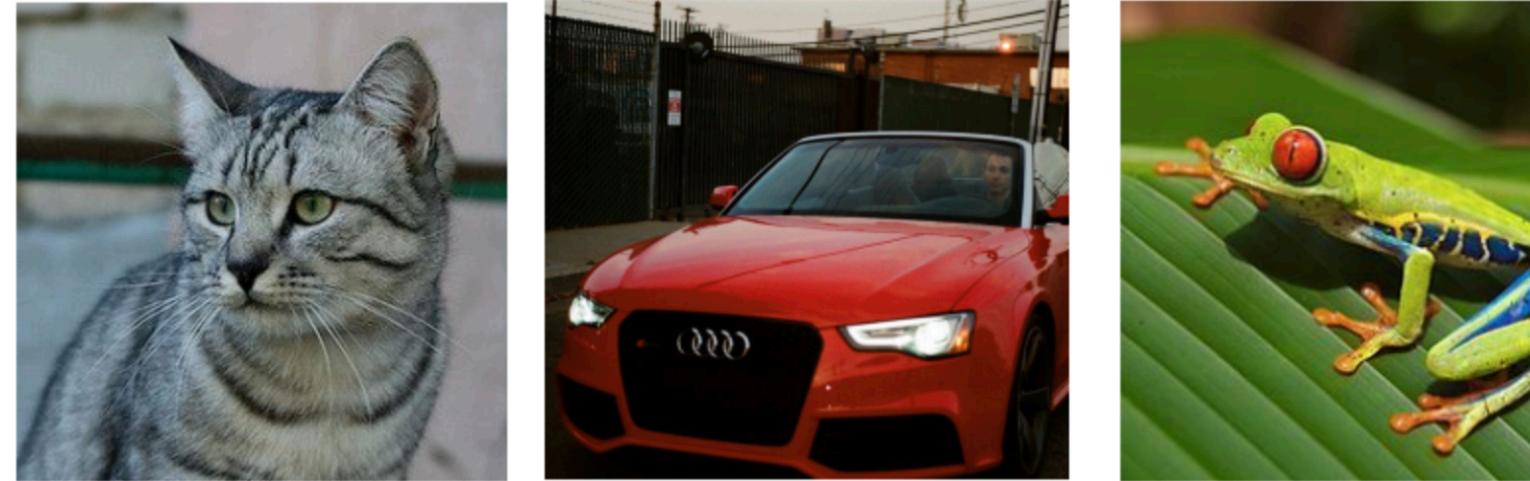
Support Vector Machine

$$L = \frac{1}{N} \sum_i L_i(f(x_i, W), y_i)$$

일반적인 손실함수 형태

Suppose: 3 training examples, 3 classes.

With some W the scores $f(x, W) = Wx$ are:



cat	3.2	1.3	2.2
car	5.1	4.9	2.5
frog	-1.7	2.0	-3.1
Losses:	2.9	0	12.9

$$L_i = \sum_{j \neq y_i} \begin{cases} 0 & \text{if } s_{y_i} \geq s_j + 1 \\ s_j - s_{y_i} + 1 & \text{otherwise} \end{cases}$$

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

safety margin = 1

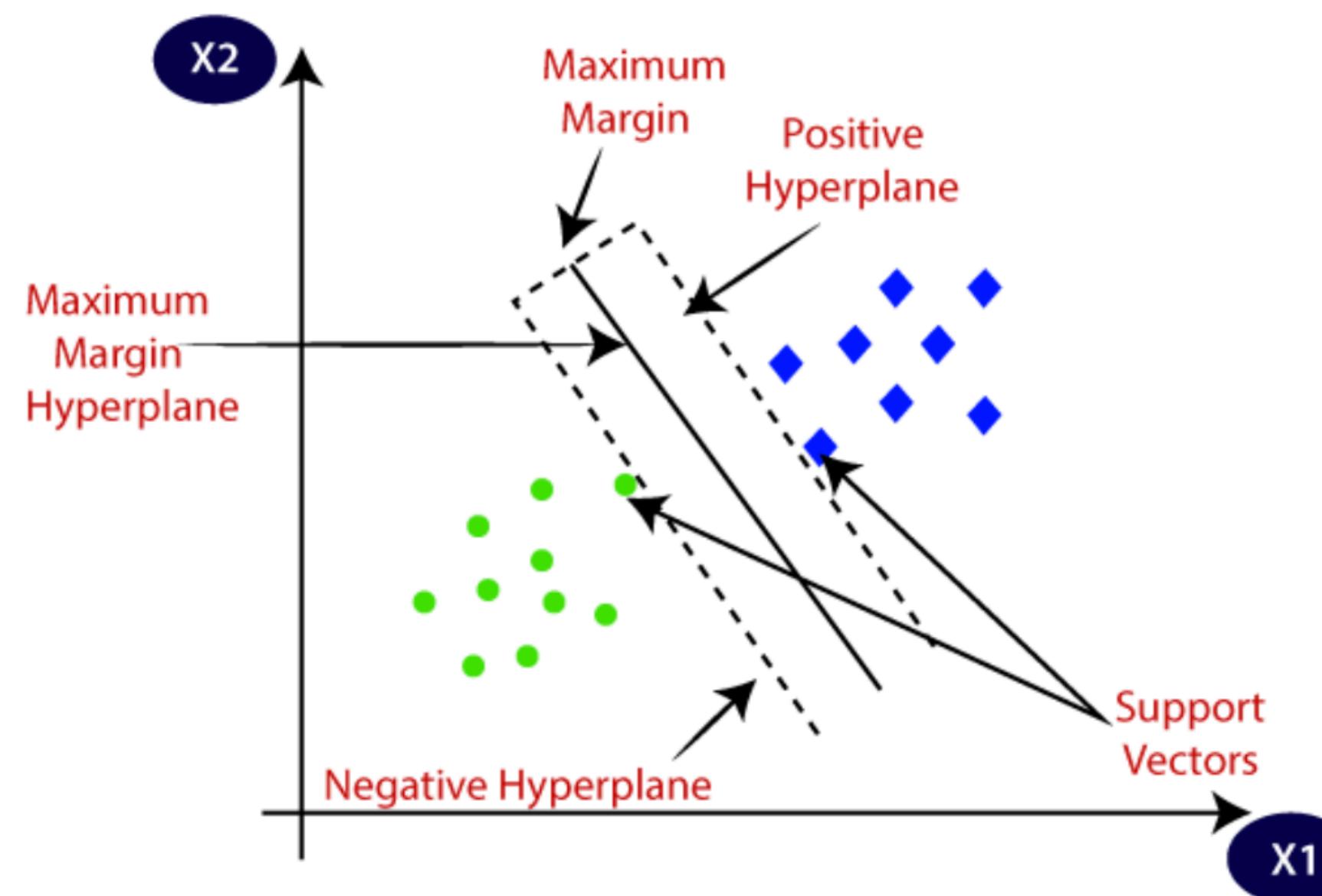
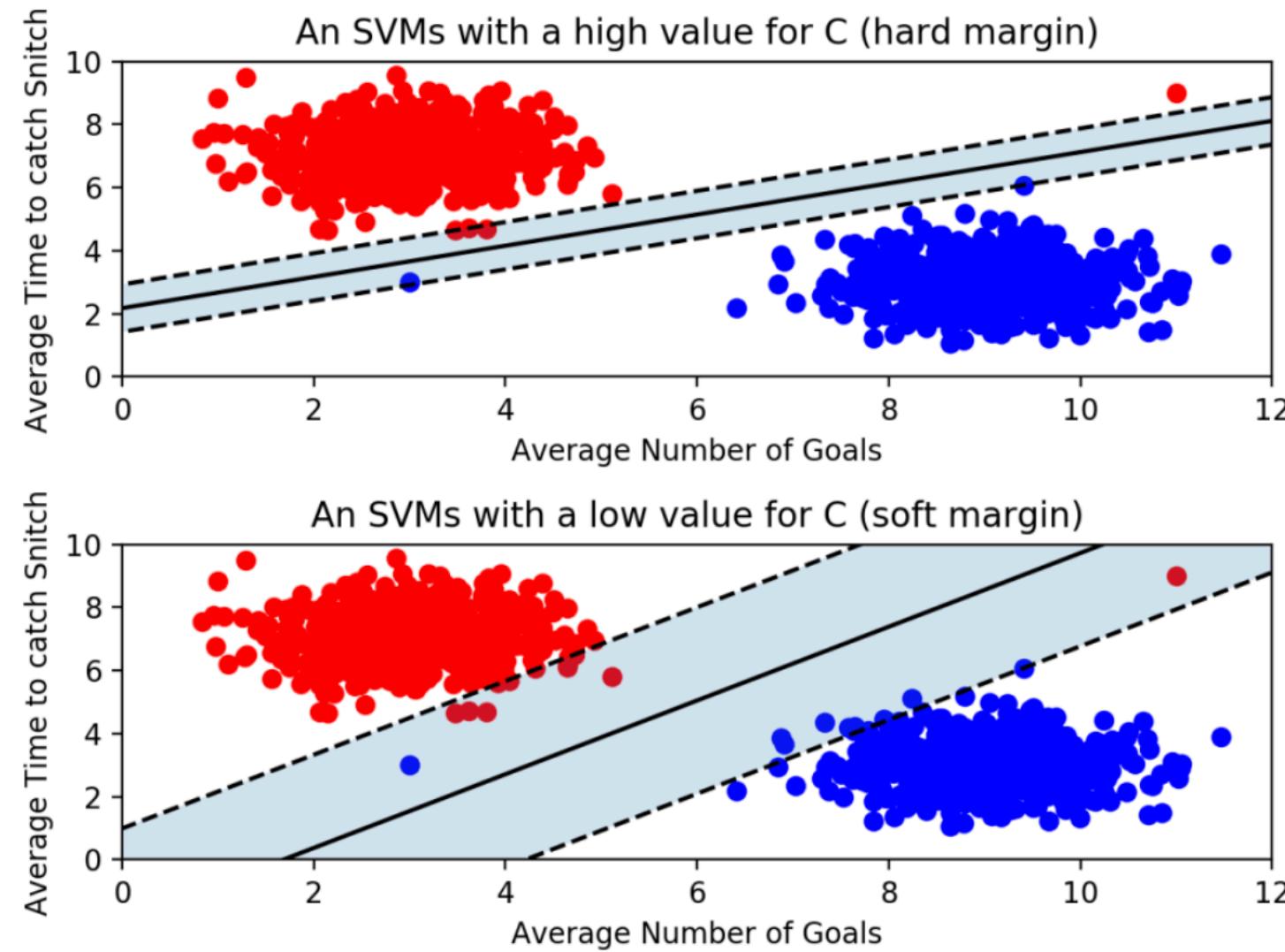
Multiclass **SVM** Loss

: safety margin을 두고 정답 클래스 점수와 예측 클래스 점수의 대소 비교

$$L = (2.9 + 0 + 12.9) / 3$$

1. Loss Function

Support Vector Machine



SVM : 분류 경계에 가장 가까운 데이터 포인트들(Support Vectors)을 이용해 최적의 결정 경계를 찾는 알고리즘

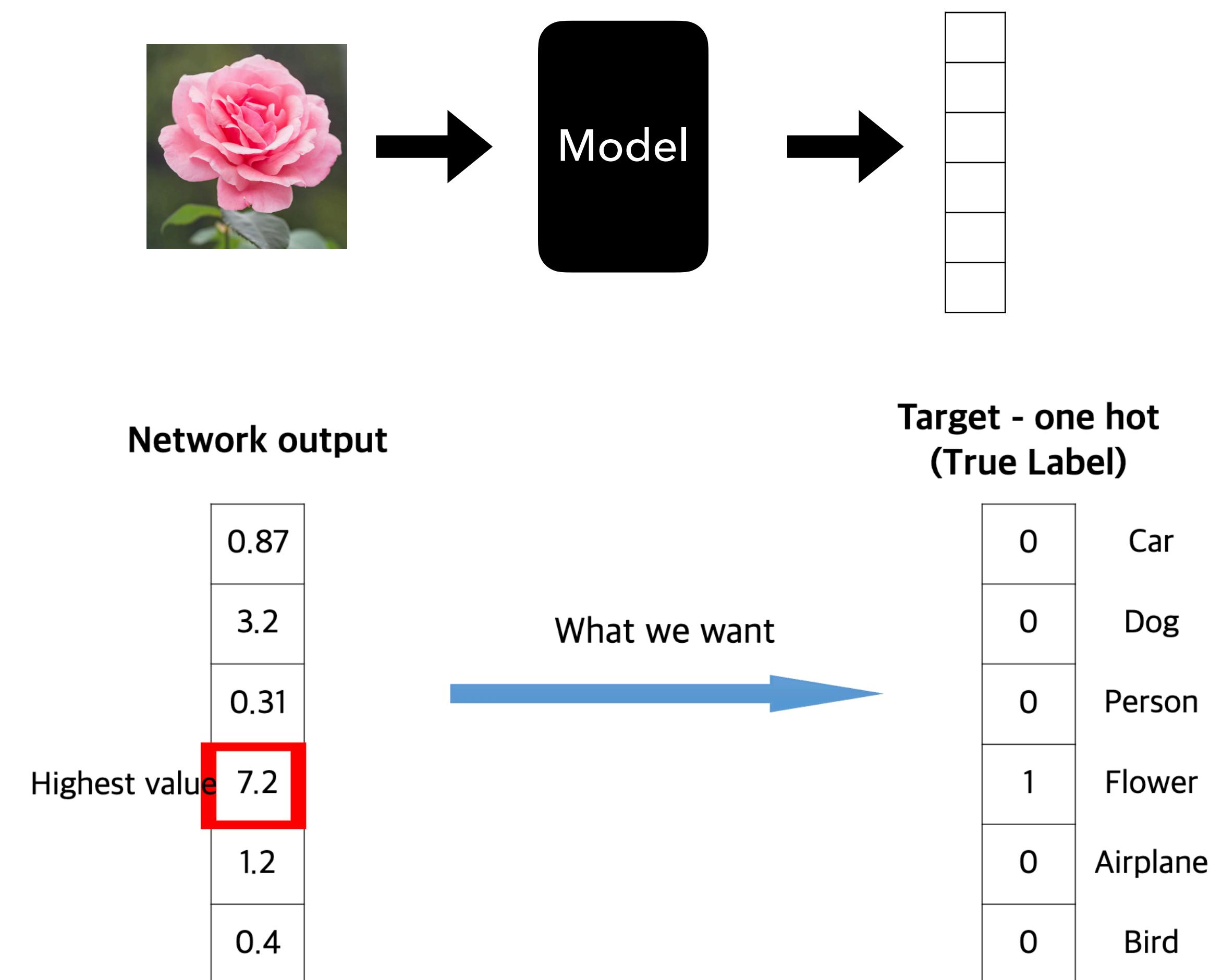
Support Vector 란,
결정경계와 인접한 데이터 포인트

margin이란, 결정경계를 기반으로 Support Vector까지의 거리
-> margin이 클수록 강건한 모델

SVM은 허용가능한 오류 범위 내에서 최대 margin을 만드는 것을 목표

1. Loss Function

Cross-Entropy Loss



- 네트워크 출력을 확률로 해석하고 싶다.
 - 신경망의 raw 출력은 보통 실수값
 - 이 값을 직접적으로 확률로 해석하기 어려움
- 정답 label은 one-hot vector

1. Loss Function

Cross-Entropy Loss

Network output

Z_1	0.87
Z_2	3.2
Z_3	0.31
Z_4	7.2
Z_5	1.2
Z_6	0.4

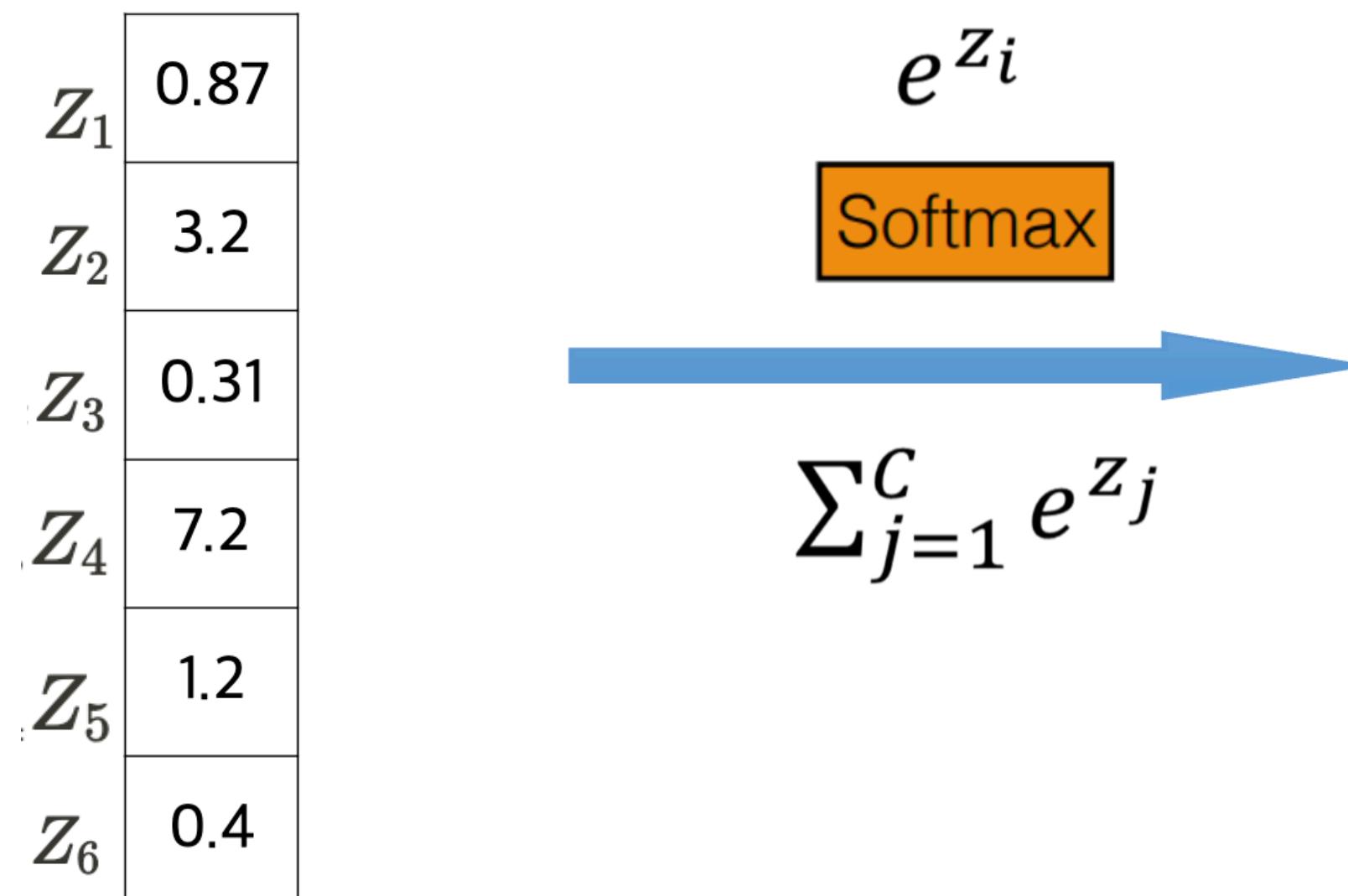
$$p_i = \frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}}$$

- 네트워크 출력을 확률로 해석하고 싶다.
 - 신경망의 raw 출력은 보통 실수값
- 출력을 의미 있는 확률 분포로 변환
- 분류 문제에서 각 클래스에 대한 확신도를 표현하기 위함
- Softmax 함수를 사용

1. Loss Function

Cross-Entropy Loss

Network output



Result after
Softmax

- 네트워크 출력을 확률로 해석하고 싶다.
- 신경망의 raw 출력은 보통 실수값
- 이 값을 직접적으로 확률로 해석하기 어려움
- 출력을 의미 있는 확률 분포로 변환
- 분류 문제에서 각 클래스에 대한 확신도를 표현하기 위함

1. Loss Function

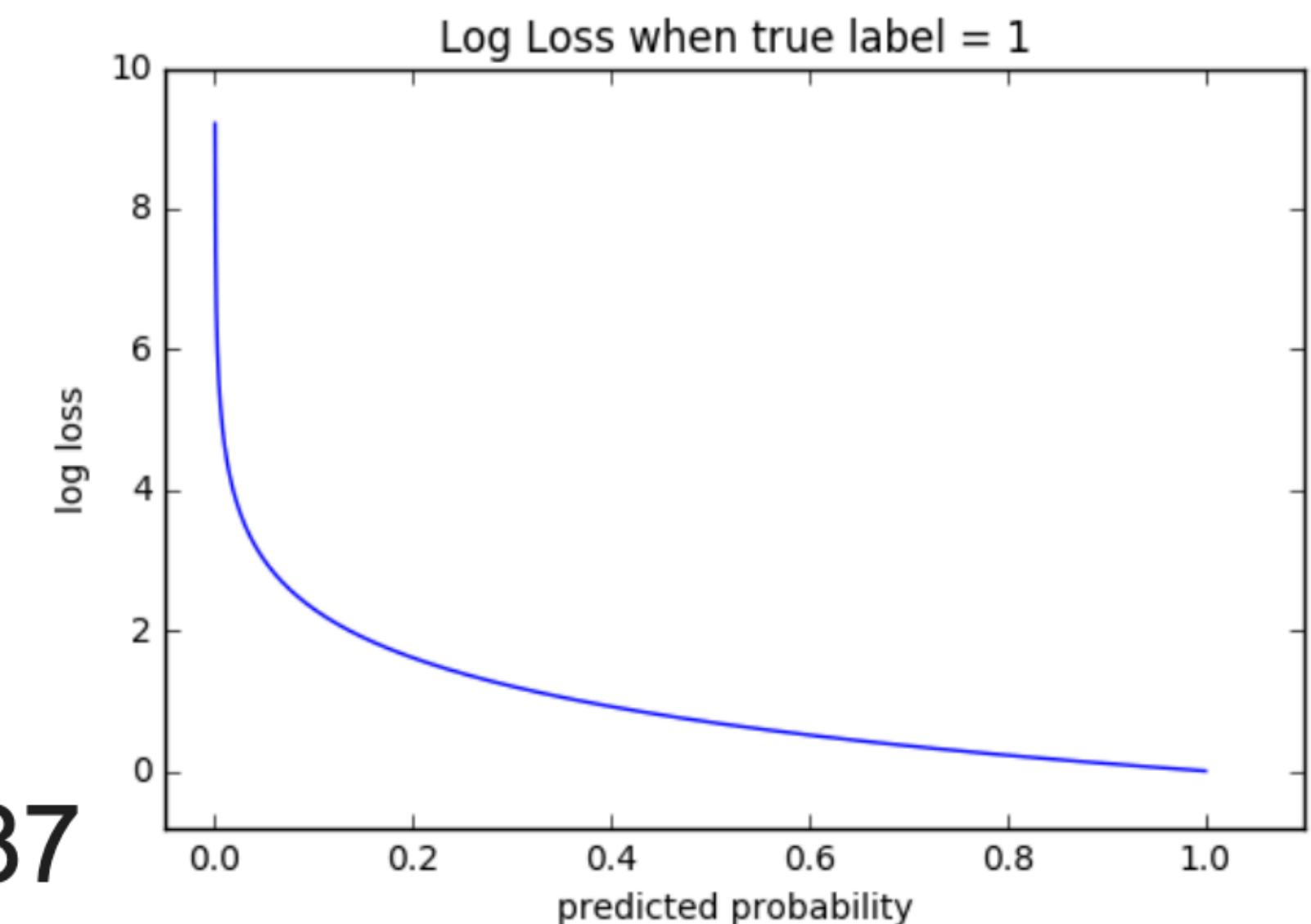
Cross-Entropy Loss

- 잘한 예측 => 작은 패널티

Softmax		True Label
p_1	1.7e-3	y_1 0
p_2	1.8e-2	y_2 0
p_3	9.9e-4	y_3 0
p_4	0.98	y_4 1
p_5	2.4e-3	y_5 0
p_6	1.1e-3	y_6 0

$$-\sum_{i=1}^C y_i \log(p_i)$$

$$-\sum_{i=1}^C y_i \log(p_i) = -\log(0.98) = 0.0087$$



1. Loss Function

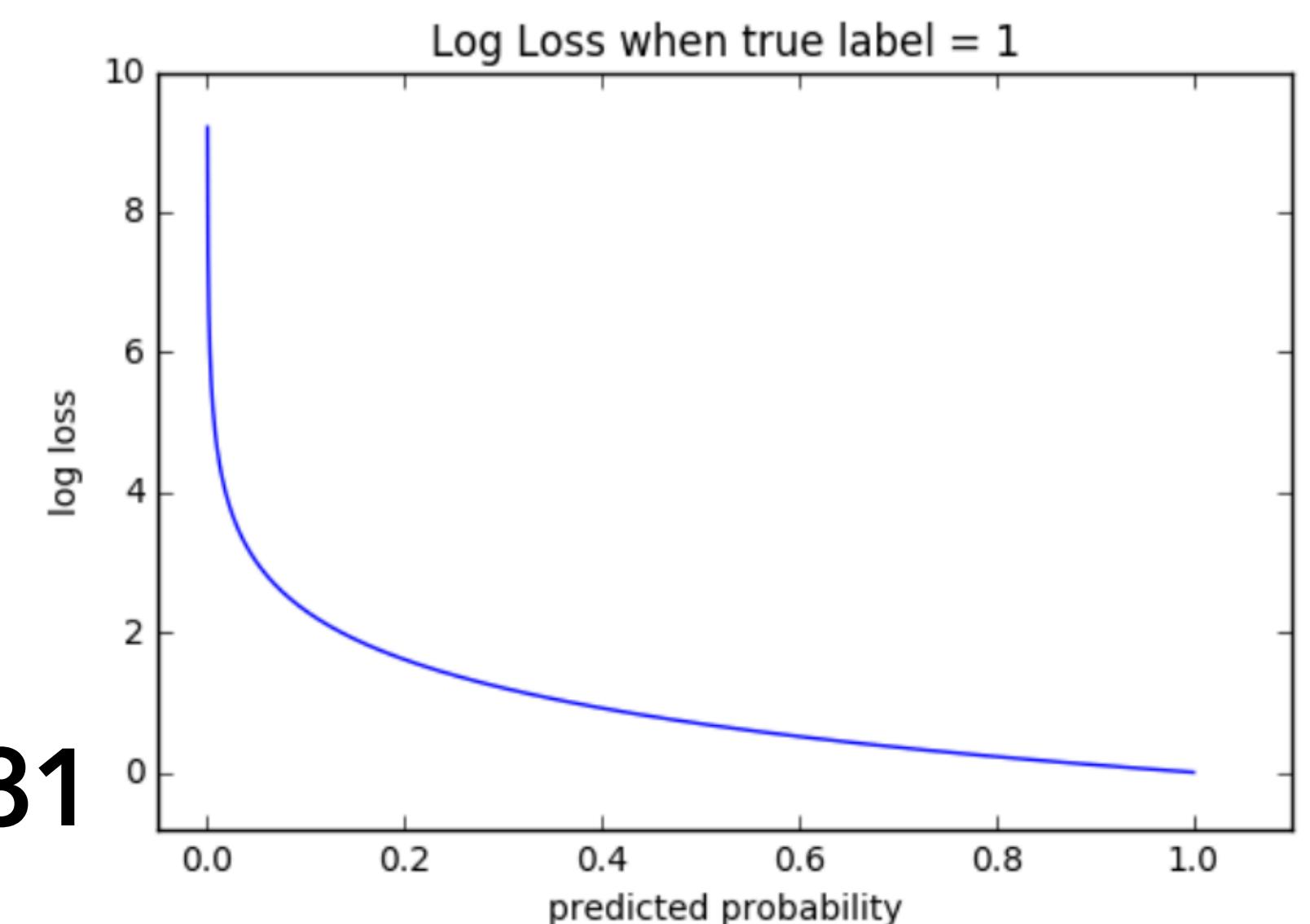
Cross-Entropy Loss

- 잘못된 예측 => 큰 패널티

Softmax		True Label
p_1	1.7e-3	y_1 0
p_2	1.8e-2	y_2 0
p_3	0.98	y_3 0
p_4	9.9e-4	y_4 1
p_5	2.4e-3	y_5 0
p_6	1.1e-3	y_6 0

$$-\sum_{i=1}^C y_i \log(p_i)$$

$$-\sum_{i=1}^C y_i \log(p_i) = -\log(9.9e^{-4}) = 1.7081$$





i = 1

j = cat cat
j = car car
j = frog frog

10
-2
3

1. Loss Function

Cross-Entropy Loss vs SVM

데이터 포인트의 점수를 약간 변경할 경우,
두 손실 함수에서 손실이 어떻게 변화하는가?

Softmax vs. SVM

SVM Loss는 두 경우 모두 0

Cross-Entropy Loss는 [10, 9, 9] 경우에 loss가 더 높다.

>> Cross-Entropy Loss는 확률 분포의 차이에 더 민감하게 반응

$$L_i = -\log\left(\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}\right)$$

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

assume scores:

[10, -2, 3]

1) [10, 9, 9]

2) [10, -100, -100]

and $y_i = 0$

Cross-Entropy Loss

1) [10, 9, 9] \rightarrow [0.5761, 0.2119, 0.2119]

$$L = -\log(0.5761) = 0.5515$$

SVM

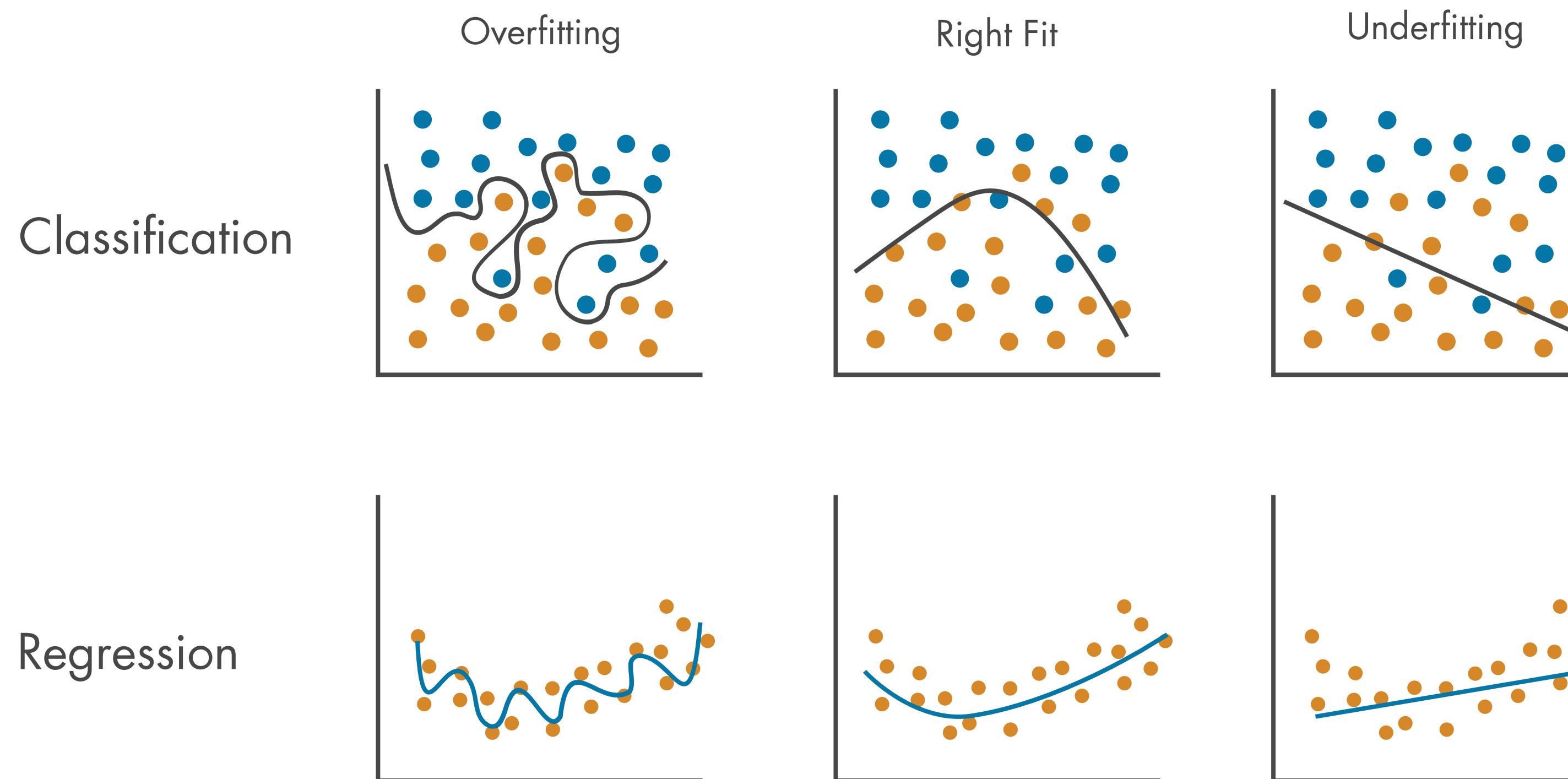
$$\begin{aligned} L &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \\ &= \max(0, 9 - 10 + 1) + \max(0, 9 - 10 + 1) \\ &= \max(0, 0) + \max(0, 0) = 0 + 0 = 0 \end{aligned}$$

$$\begin{aligned} L &= \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1) \\ &= \max(0, -100 - 10 + 1) + \max(0, -100 - 10 + 1) \\ &= \max(0, -109) + \max(0, -109) = 0 + 0 = 0 \end{aligned}$$

2. Regularization

Overfitting이 왜 안좋은데?

- Overfitting : 모델이 훈련 데이터에 너무 가깝게 맞춰져 새 데이터에 어떻게 대응해야 할지 모를 때

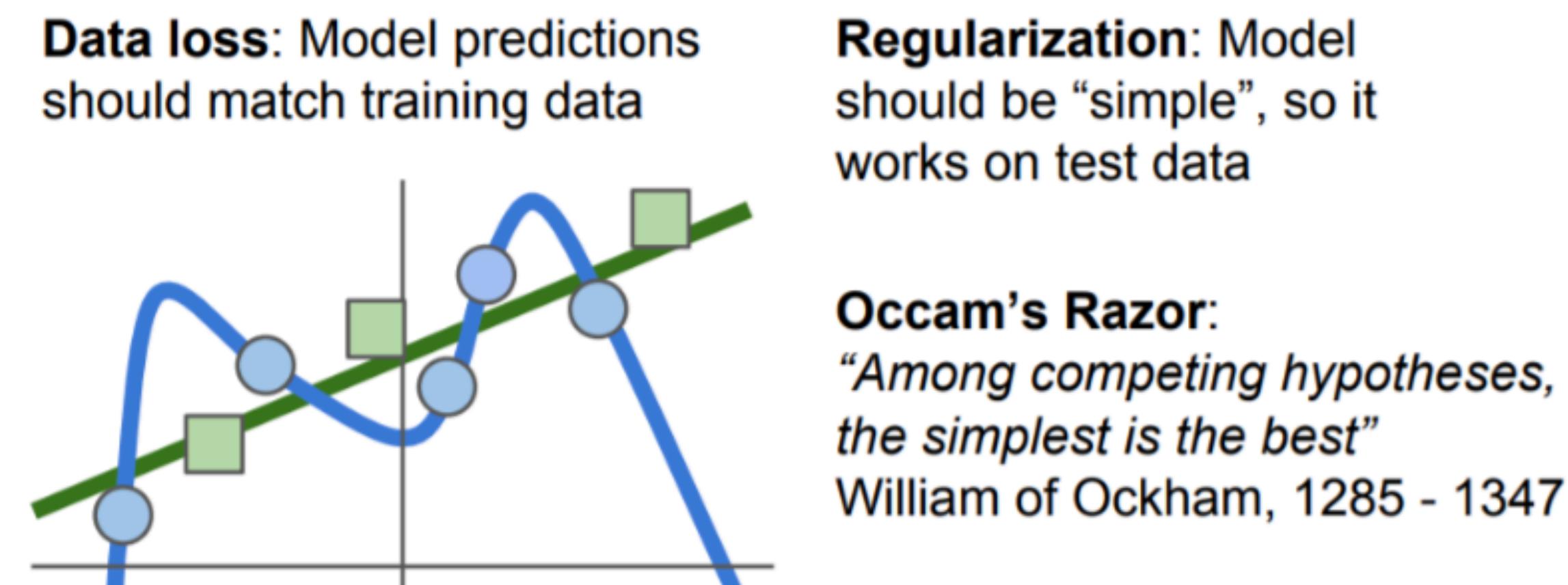


2. Regularization

Overfitting이 왜 안좋은데?

- Regularization은 Loss함수에 parameter에 대한 term을 추가하여 적용
- 가중치 w 가 작아지도록 학습한다 = Local noise 에 영향을 덜 받도록 한다.

$$L(W) = \underbrace{\frac{1}{N} \sum_{i=1}^N L_i(f(x_i, W), y_i)}_{\text{Data loss: Model predictions should match training data}} + \underbrace{\lambda R(W)}_{\text{Regularization: Model should be "simple", so it works on test data}}$$



2. Regularization

Regularization으로 Overfitting을 막자

- 모델의 복잡도를 제어하고 과적합을 방지

$$Cost = \frac{1}{n} \sum_{i=1}^n \{L(y_i, \hat{y}_i) + \frac{\lambda}{2} |w|^2\}$$

- L2 Regularization

- 매끄러운 그래프를 원할때 쓰는 정규화

- 큰 가중치에 더 큰 페널티를 부과,
극단적인 가중치를 피하고, 모든 특성이 적당히 기여하도록 유도

- L1 Regularization

$$Cost = \frac{1}{n} \sum_{i=1}^n \{L(y_i, \hat{y}_i) + \frac{\lambda}{2} |w|\}$$

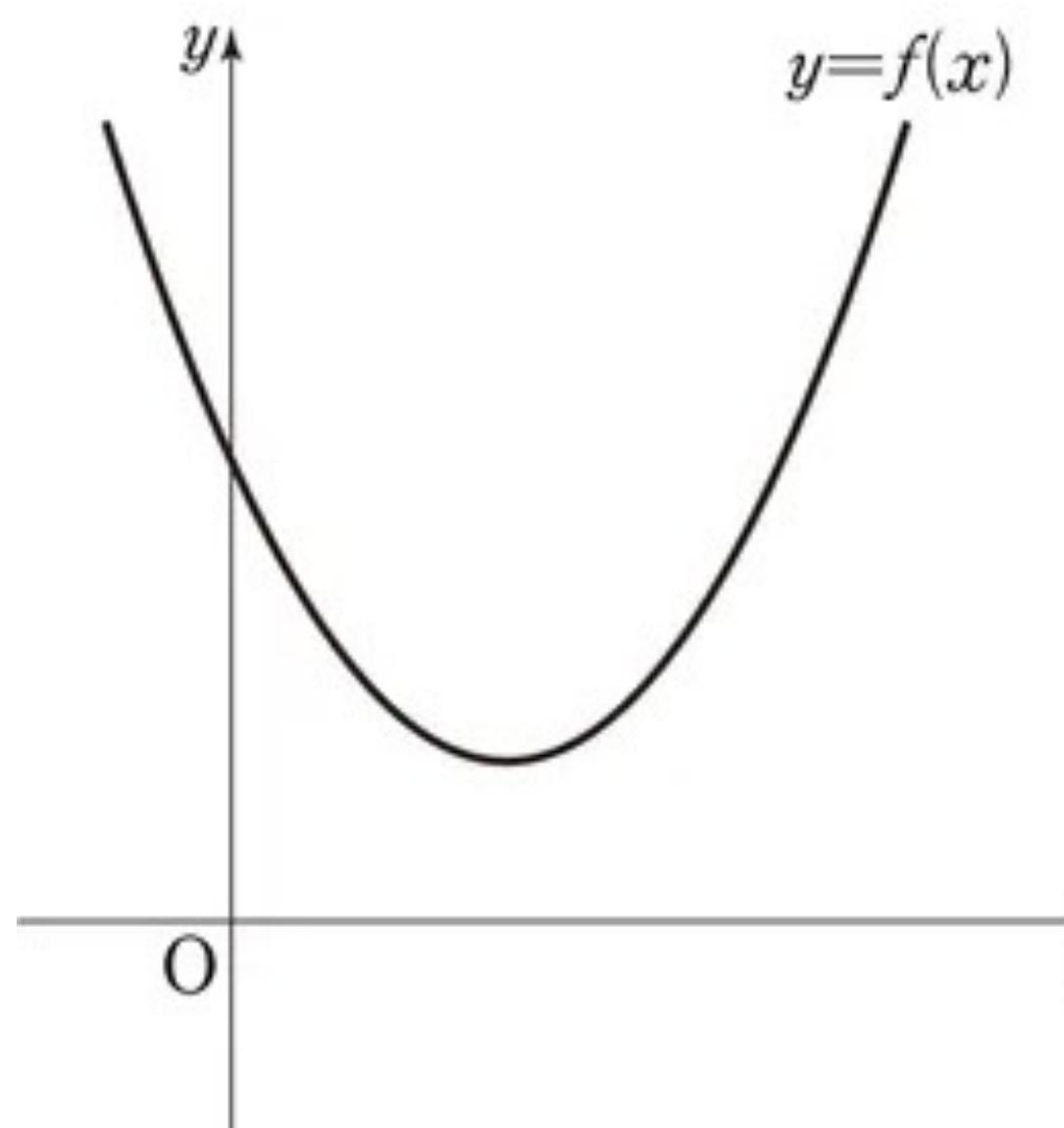
- 분류기가 복잡하다고 느껴질때 쓰는 정규화

- 가중치값에 0이 많도록 하여 보다 더 단순한 식을 만들어준다.

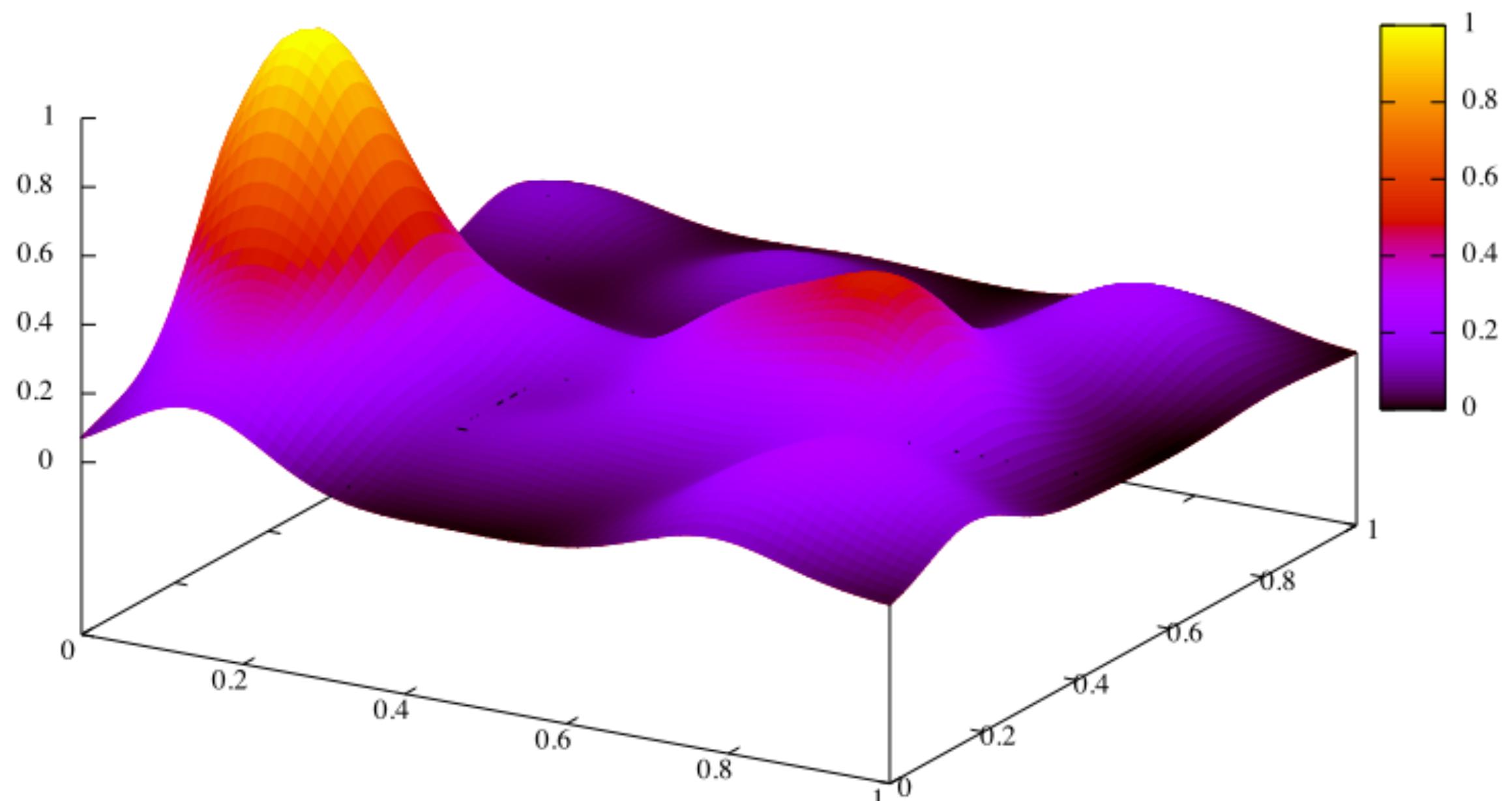
3. Optimization

Loss Function이 줄어들도록 조정하는 방법

- 2차함수에서는 쉽다.



- 손실함수는 다변수 함수

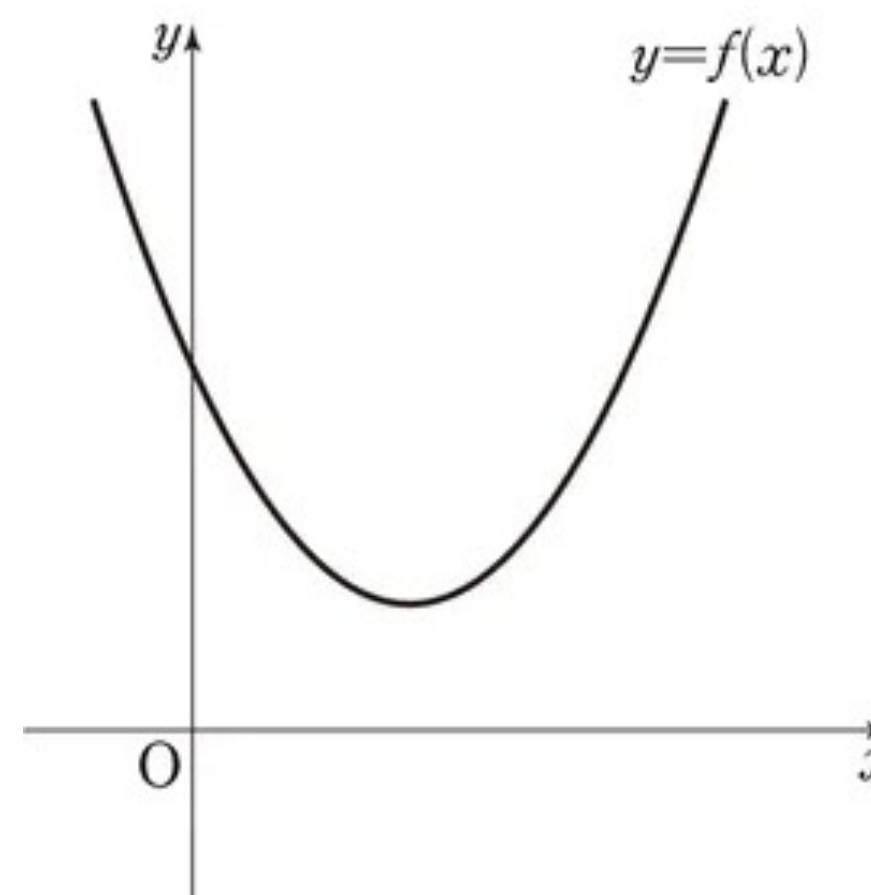


3. Optimization

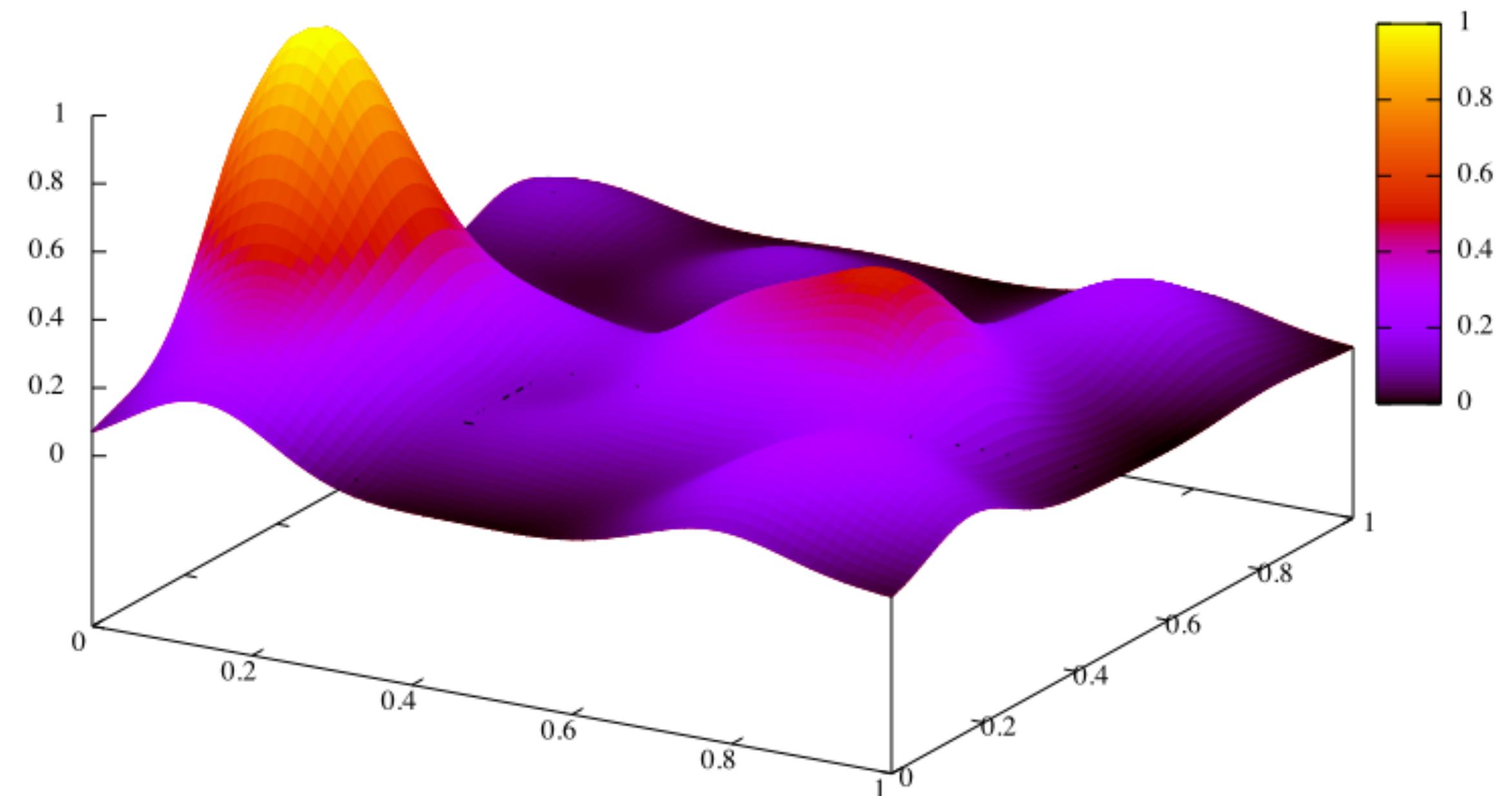
Loss Function이 줄어들도록 조정하는 방법

- 가장 빠르게 손실함수 줄어드는 방향으로 : 가장 가파르게 감소
- 최소지점으로 가는 것이 목표 : 가장 가파르게 감소하는 방향으로 step
- Gradient Descent

$$\nabla L(w_1, w_2, \dots, w_n) = \left(\frac{\partial L}{\partial w_1}, \frac{\partial L}{\partial w_2}, \dots, \frac{\partial L}{\partial w_n} \right)$$



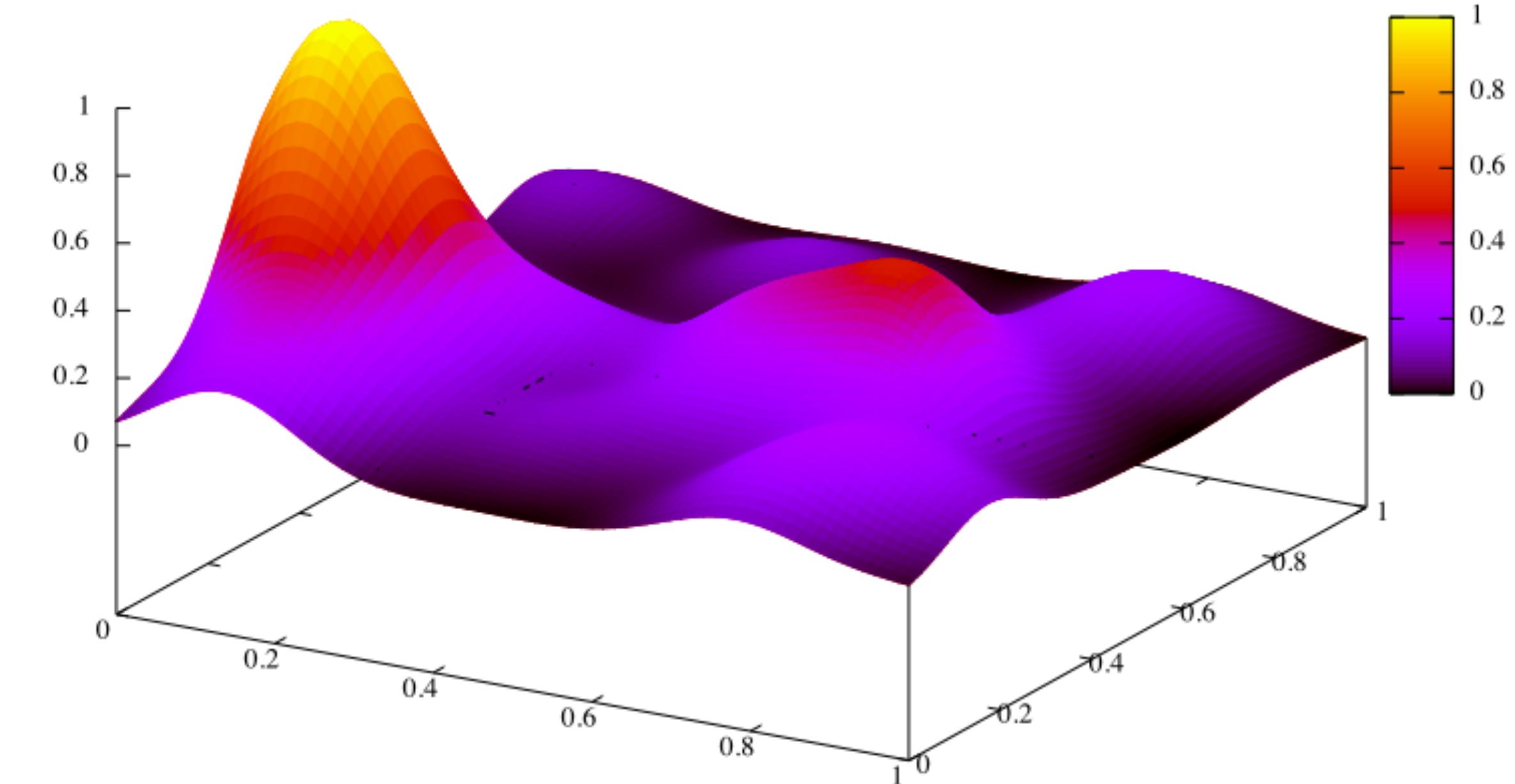
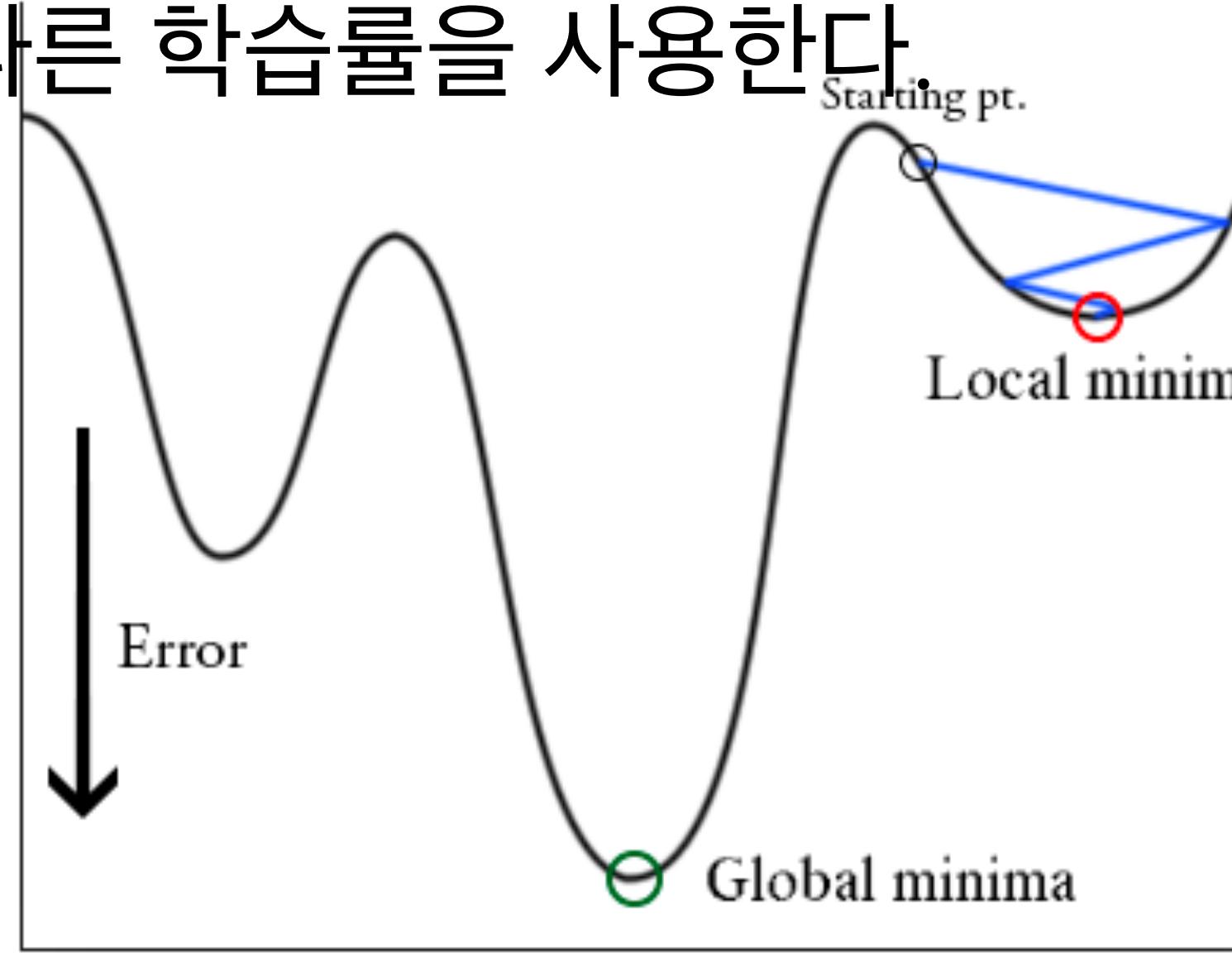
$$\vec{w} = \vec{w} - \eta \times \nabla L(\vec{w})$$



3. Optimization

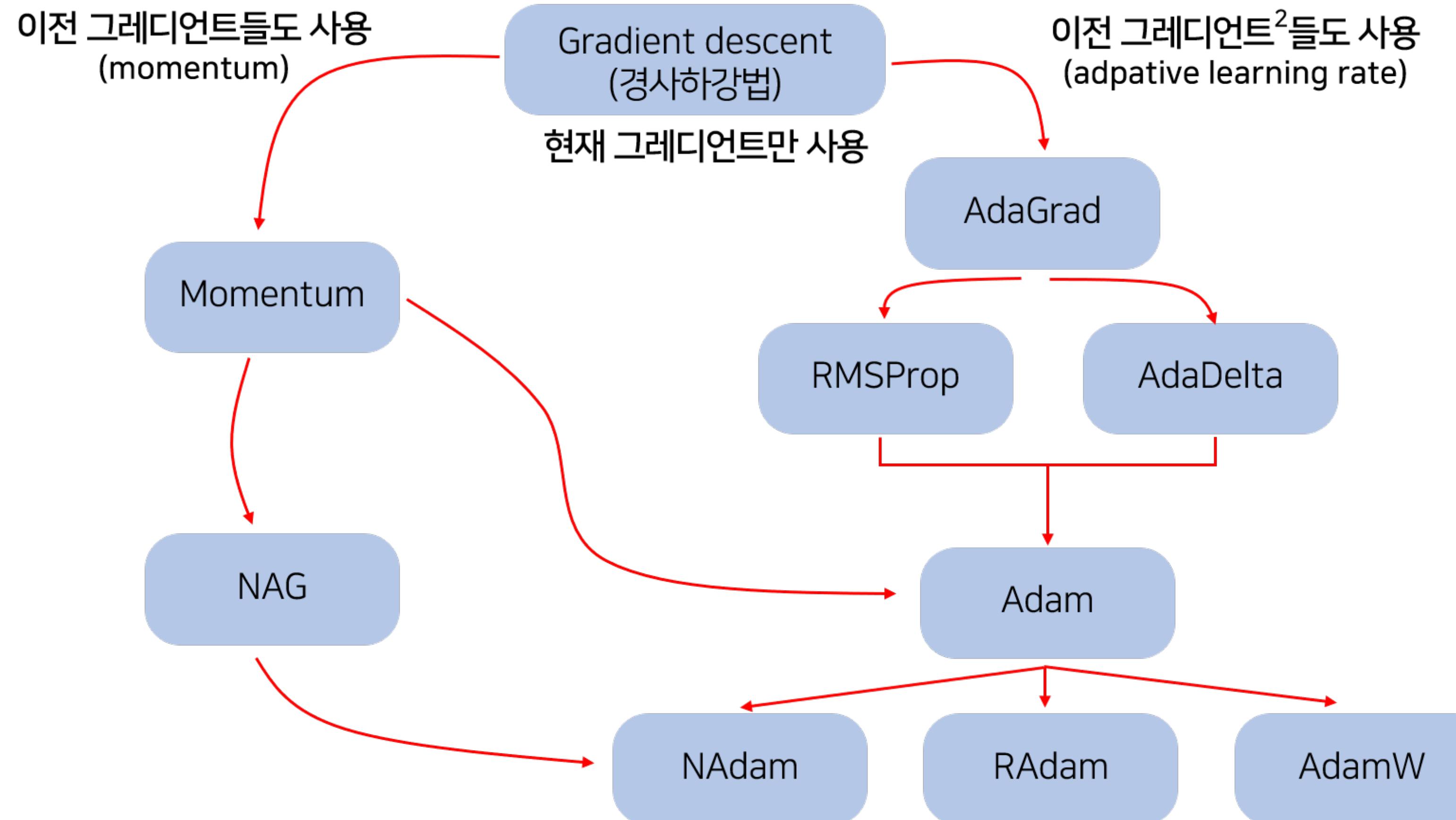
Loss Function이 줄어들도록 조정하는 방법

- Vanilla Gradient Descent : Local minimum
 - Momentum(관성): 이전 단계의 업데이트 방향을 고려하여 현재 업데이트에 반영
 - Adaptive Learning rate: 과거의 그래디언트 정보를 활용하여 각 파라미터마다 서로 다른 학습률을 사용한다.



3. Optimization

Loss Function이 줄어들도록 조정하는 방법



3. Optimization

Loss Function이 줄어들도록 조정하는 방법

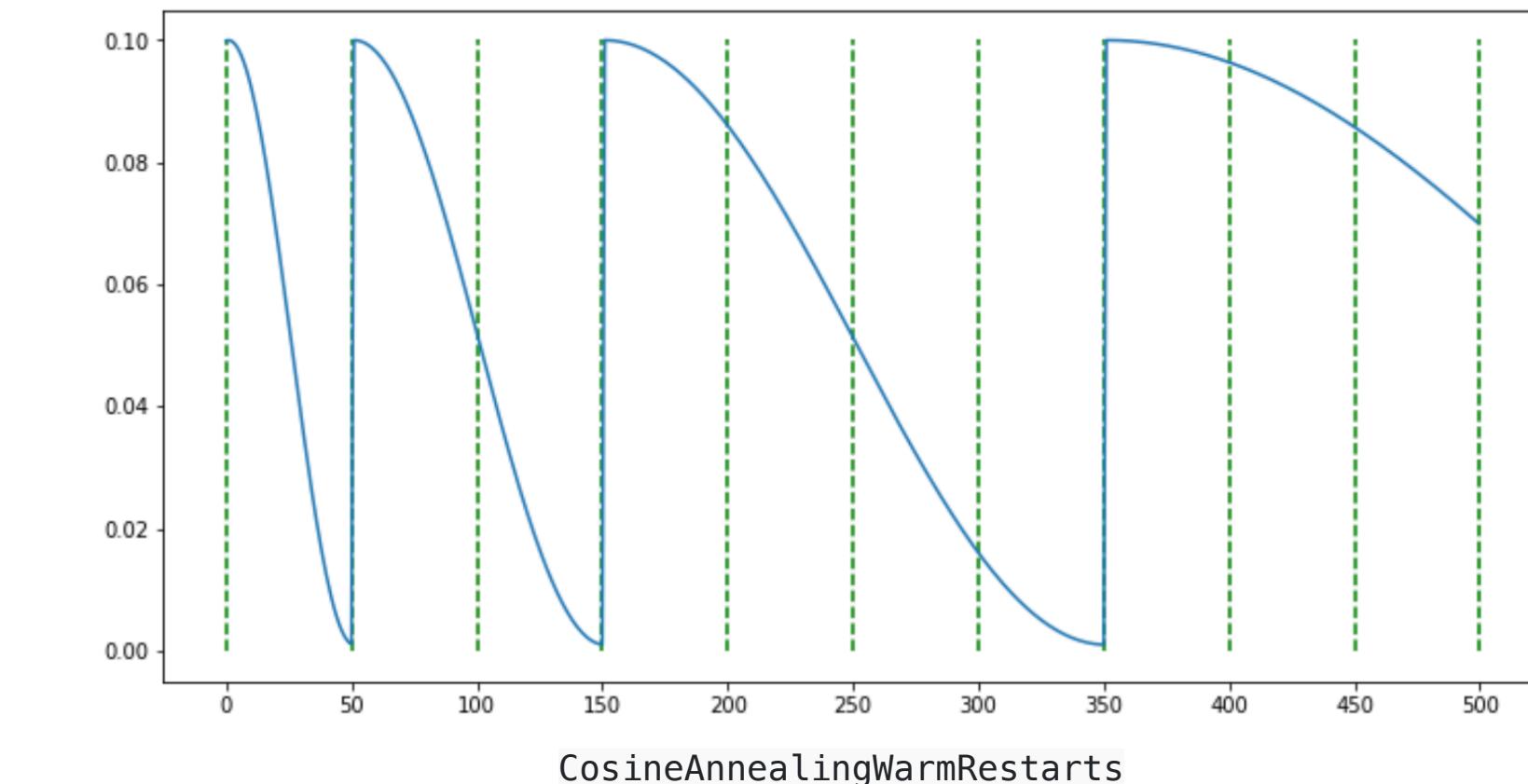
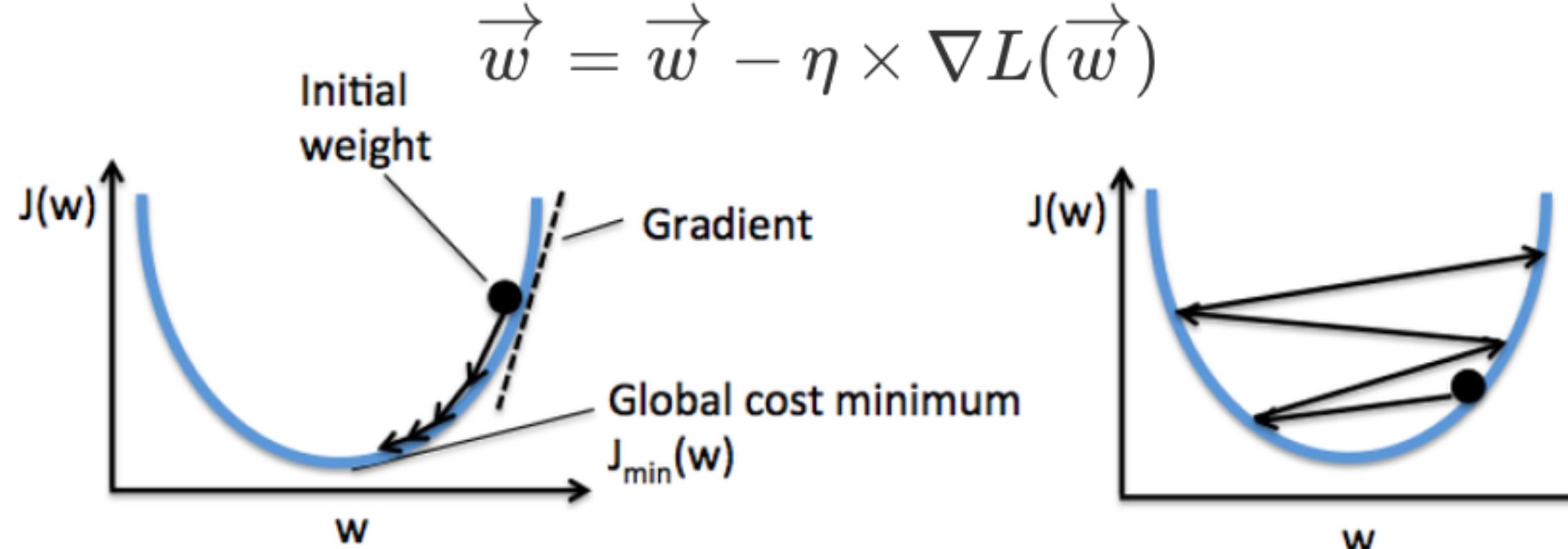
```
# optimizer  
optimizer = optim.Adam(model.parameters(), lr=1e-3)
```

- Gradient Descent

```
# scheduler  
scheduler = optim.lr_scheduler.LambdaLR(optimizer,  
                                         lr_lambda=lambda epoch: 0.95 ** epoch,  
                                         last_epoch=-1,  
                                         verbose=False)
```

- Learning rate scheduler

- 초기에 높은 학습률을 사용하면 빠르게 최적점 근처로 접근
- 적절한 시점에 학습률을 높이면 지역 최소값에서 벗어날 수 있다.



4. Summary

- 1. Loss Function
- 2. Regularization
- 3. Optimization

