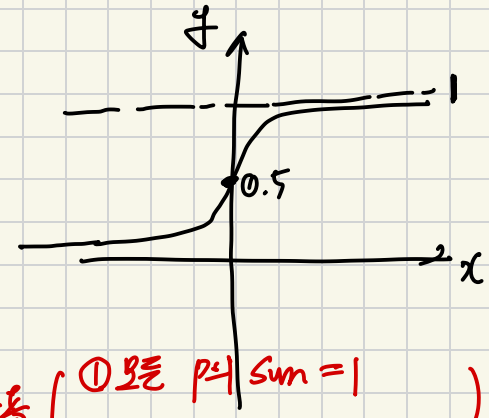
Softmax → model 예측에 해석 가능성 제공 (0과 1 사이로 mapping)

↳ 이진분류에서 class1의 점수 $=S_1$
& 2의 점수 $=S_2$ 인 경우



answer = class1인 경우 $S_1 - S_2$ 가 ⇑ (1에 가까이)
" class2 " $S_1 - S_2$ 가 ⇓ (0에 가까이)

$$\sigma(S_1 - S_2) = \frac{1}{1 + e^{-(S_1 - S_2)}}$$

↳ 확률해석사용 (① 모든 p의 sum = 1
② 모든 p는 $0 \le p \le 1$ 이다.)

즉, $P(Y=1|X) = \sigma(S_1 - S_2) = \frac{1}{1+e^{-(S_1-S_2)}} = \frac{e^{S_1}}{e^{S_1}+e^{S_2}}$

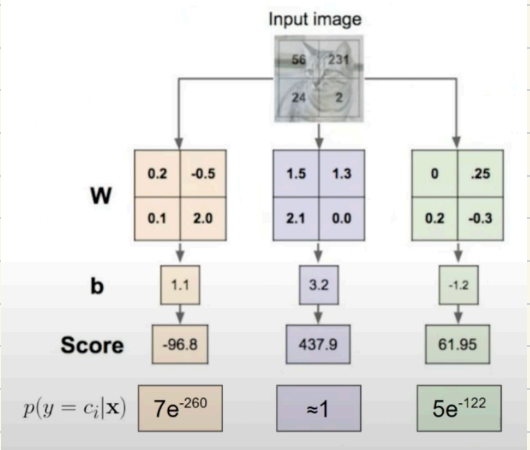
$P(Y=0|X) = \sigma(S_2 - S_1) = \frac{1}{1+e^{-(S_2-S_1)}} = \frac{e^{S_2}}{e^{S_1}+e^{S_2}}$

] sum = 1.

↳ Generalize.

$$P(Y=k|X) = \frac{e^{S_k}}{\sum_{i=1}^{n} e^{S_i}}$$ → 해당 index.

$P(Y=3|X) = 0.8$ 이라고 해서 class=3 예측의 80%가 정말이라는게 아니다. Just confidence 역할.
But threshold 지정에 유용하다.



다음과 같이 Last Layer에 Softmax classifier를 달아서 Model의 예측을 해석가능하도록 만들 수 있다.

< Parameter 학습 >

Human은 model의 틀만 정함. 나머지는 model이 Data로 부터 학습한다.

answer를 보고 $\hat{y} \approx y$ 되도록 W를 조정한다.

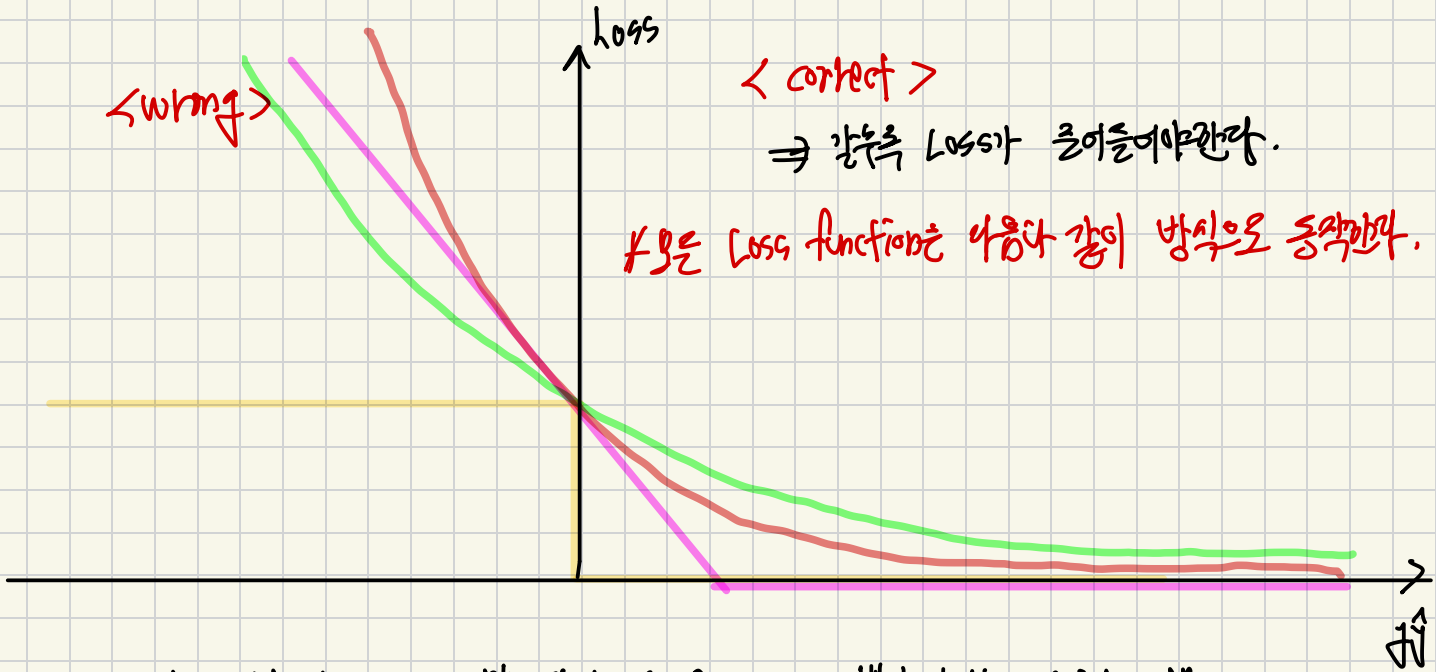
Loss function. : Model이 잘 학습하는지 수치화 해준다.

$L(y, \hat{y})$를 계산한다.

$y = +1, -1$ 인 경우. ($\hat{y}$의 부호를 가지고 예측한다)

즉, $y\hat{y} > 0$ 인 경우 = model의 예측이 맞은 경우

$y\hat{y} < 0$ 인 경우 = model의 예측이 틀린 경우.



< correct >

⇒ 같을수록 Loss가 줄어들어야한다.

K들은 Loss function은 다음과 같이 방식으로 동작한다.

— : 가장 이상적인 Loss : 맞으면 Loss=0, 틀리면 Loss=k) ⇒ 미분이 안되는 point가 너무 critical해서 사용X.

— : $\log loss = \log(1+e^{-y\hat{y}})$ ≃ Logistic Regression Loss

— : $e^{-y\hat{y}}$ ↳ Noise에 취약

— : Hinge Loss : $\max(0, 1-y\hat{y})$ ⇒ 미분X point가 있지만 critical X 에서 상관X. → 상수곱제외

≃ SVM

$y = [0, 1]$ 인 경우

$p(y=0|x) = \hat{y}$, $p(y=1|x) = 1-\hat{y}$ 이다.

$y = [0, 0, 0, 1, 0, 0, 0]$
$\hat{y} = [0.2, 0.0, 0.0, 0.4, 0.1, 0.1, 0.3]$
] 다음과 같은 형태를 갖는다.

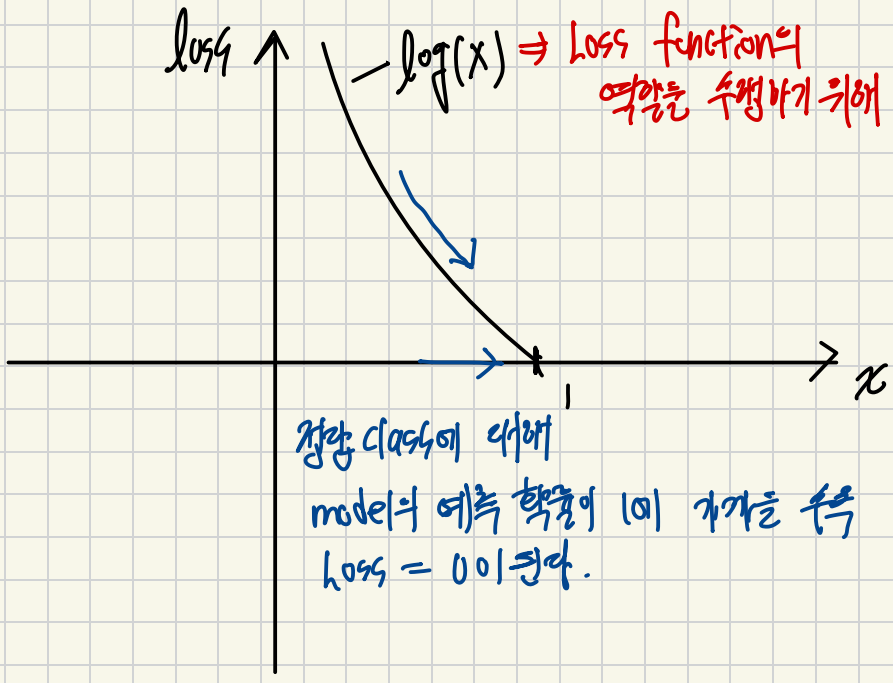
* y와 $\hat{y}$의 확률분포가 비슷하면 model이 잘 예측했다고 생각 가능.

"Cross Entropy"

$$L = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{K} y_{ik}\log(\hat{y}_{ik})$$

y_{ik}: 정답확률 (정답=1, else=0)
$\hat{y}_{ik}$: model 예측 확률.

∴ 정답에 대해 얼마나 확신을 가지는지를 나타내준다.

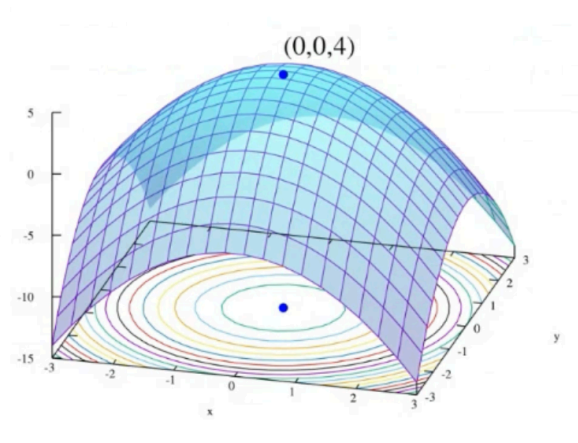


"KL-Divergence"

$$D_{KL}(P||Q) = \sum_i P(i)\log\frac{P(i)}{Q(i)}$$

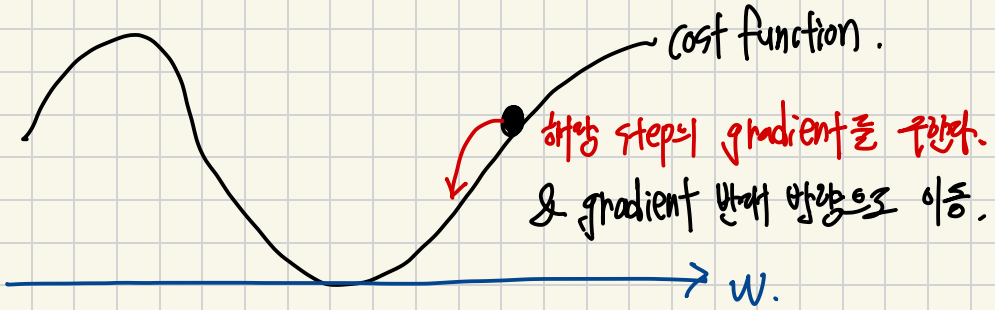
P와 Q의 확률분포 사이의 차이를 구하는 방법.

Optimization



Loss function = Cost function 을 minimize.

※ cost function 에서 최적화하는 model의 가중치이고, 곧 Loss이다.



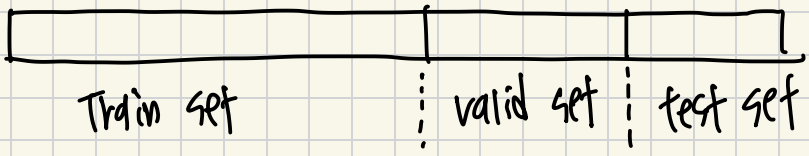
$$\theta^{new} = \theta^{old} - \eta \cdot \frac{\nabla \text{cost\_function}}{\nabla \theta^{old}}$$

mini-batch ⇒ N개의 sample 만 사용해서 이동하자. (N이 증가할 경우 정확. 하나 cost↑)

Cross Validation

Train 성능↑이 목표가 아닌, 일반적인 data를 잘 예측하는 model이 목표이다.

hyperparameter 설정을 위한 validation set을 만들어야한다.



valid set이 마지막에 Train 바로에 이득일수도, 아닐수도 있다. (valid에 overfitting 될수도 있다)