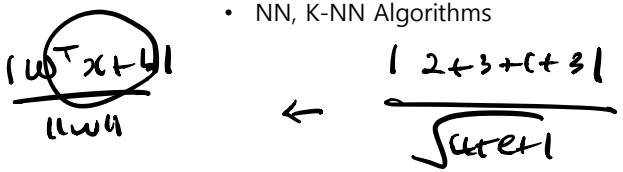
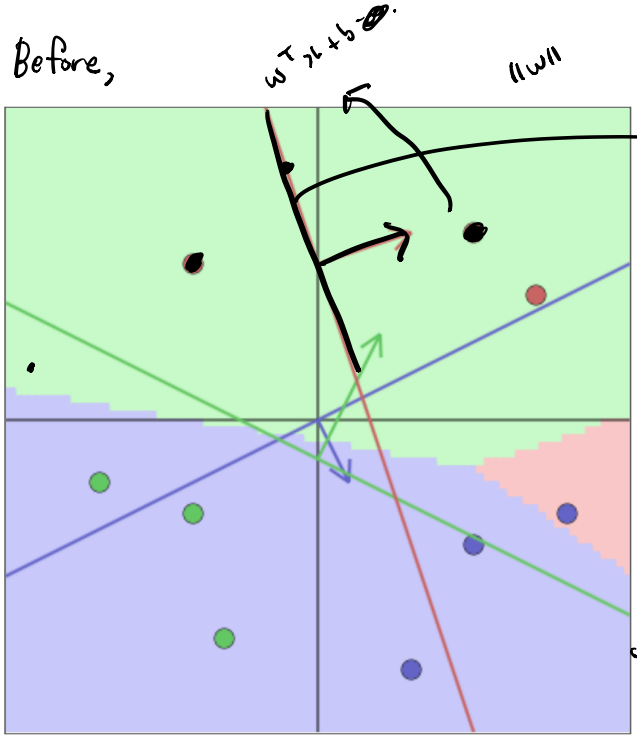
Lecture 2.
Linear Classifier

Review

1. **What is Machine Learning?**
   - Definition
   - Fields of ML
   - Narrow down to Image Classification
2. **Making a Model I**
   - Narrow down to Iris Classification
   - Data-Driven Approach
     - NN, K-NN Algorithms
3. **How to Test a Model**
   - Hyperparameter
   - Cross Validation
4. **Making a Model II**
   - Limitaions of Data-Driven Approach
   - Parametric Approach
   - Linear Classifier : Algebraic & Geometric



Before,



$w^T x + b = 0$

red label의 decision hyperplane.

이 hyperplane 위에 점이 있다면?

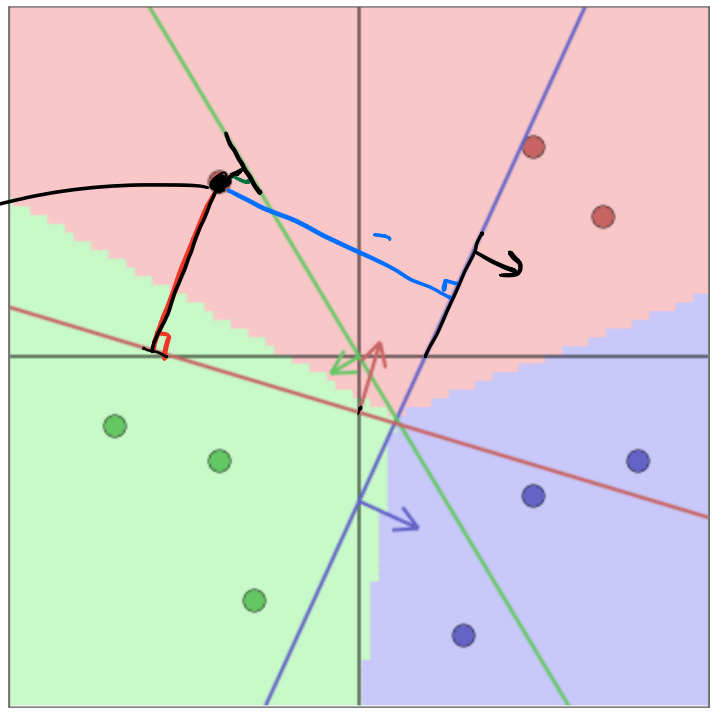
score = 0.

이 hyperplane에서 w의 방향으로 멀어질수록,

score ↑.

e 같은 param. incorrect.

After,

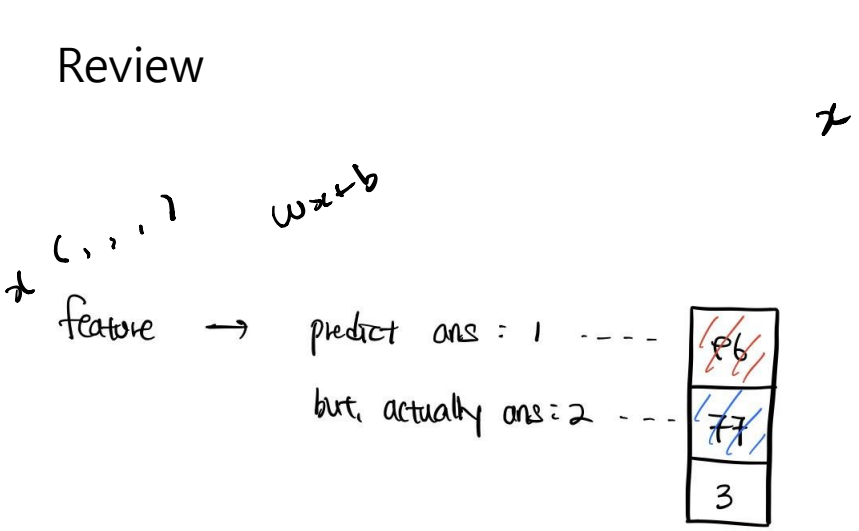


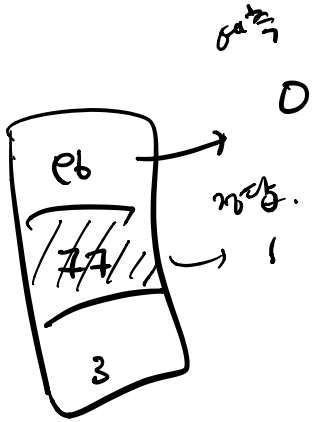
red label의 경우

마찬가지, predict as red.

same for other points.

Review





Review

Questions

- **Train이 잘 되었는지 판단할 수치적 척도 필요**

: define a **Loss Function** that quantifies our unhappiness with the scores across the training data

- **Parameter를 update하는 algorithm 필요**

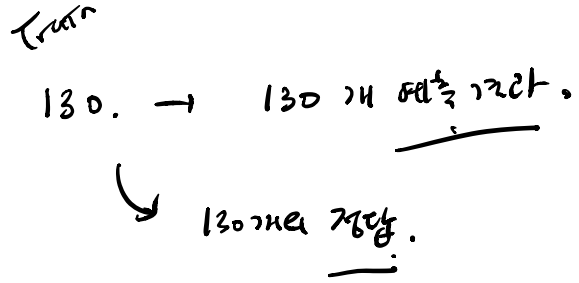
: come up with a way of efficiently finding the parameters that minimize the **Loss Function**

21

Today's Contents

1. **Loss**
2. **Loss Function**
3. **Regularization**

Loss



How is a Model Optimized / Updated?

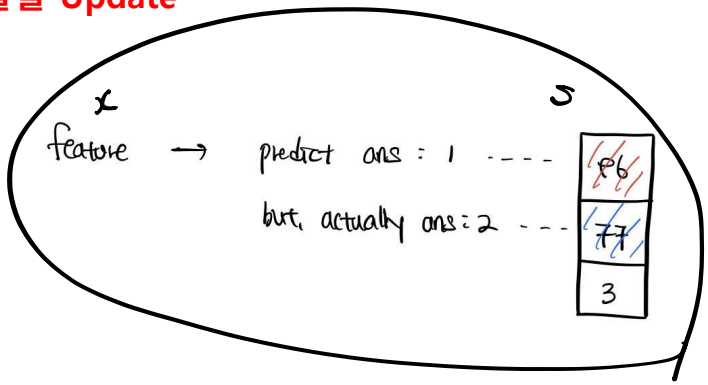
1. Training Set의 Data들을 Linear Classifier에 통과시켜서, 그 결과들을 정답과 비교
2. 1에서의 결과를 바탕으로 Parameter의 값들을 Update
3. 다시 1로

Loss

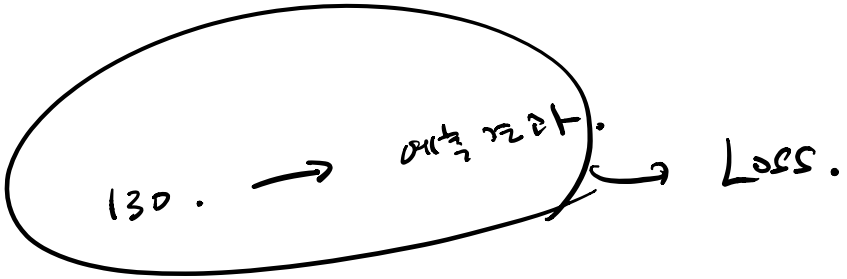
2. 1에서의 결과를 바탕으로 Parameter의 값들을 Update

정답과의 차이가 "많다"면?

정답과의 차이가 "작아" 서 거의 비슷하다면?



Loss

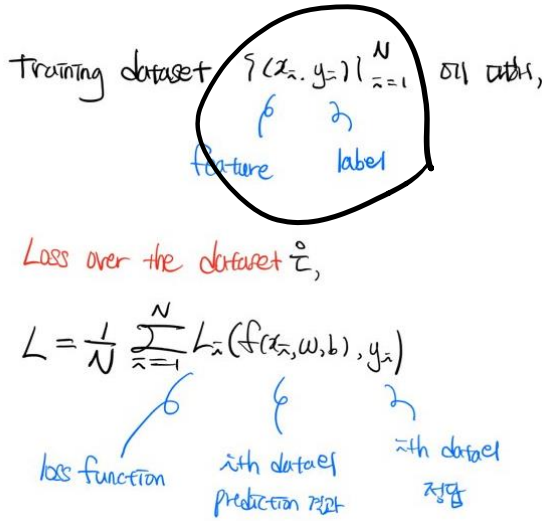


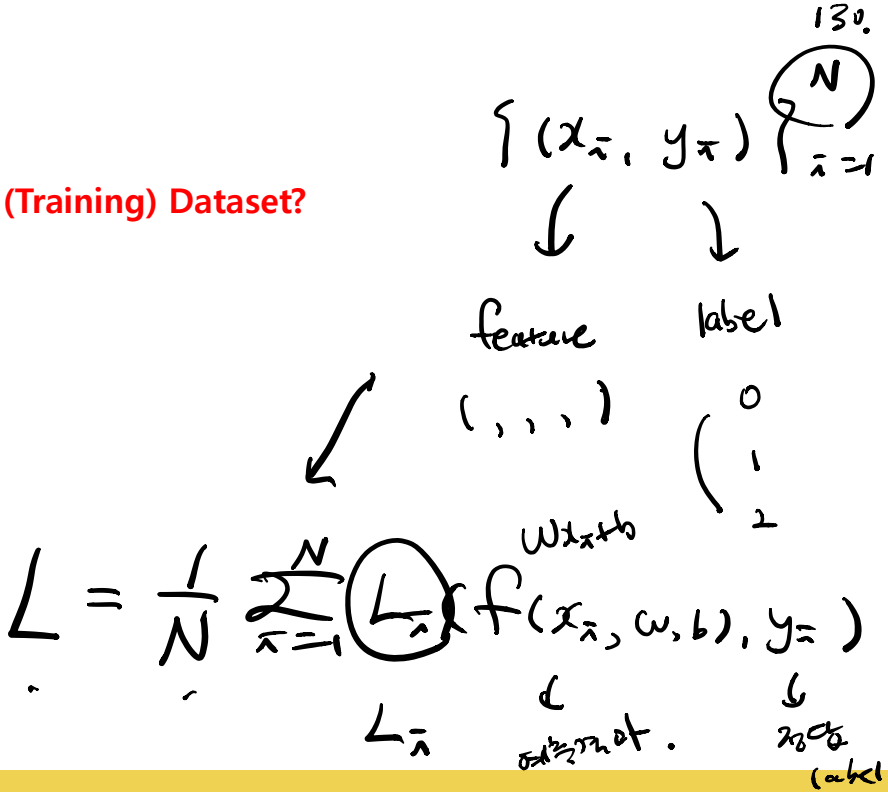
Loss Over Dataset : 현재 Model의 Parameter가 주는 결과가 얼마나 부정확한지를 숫자로 표현

We want to, (MINMIZE / MAXMIZE) the LOSS

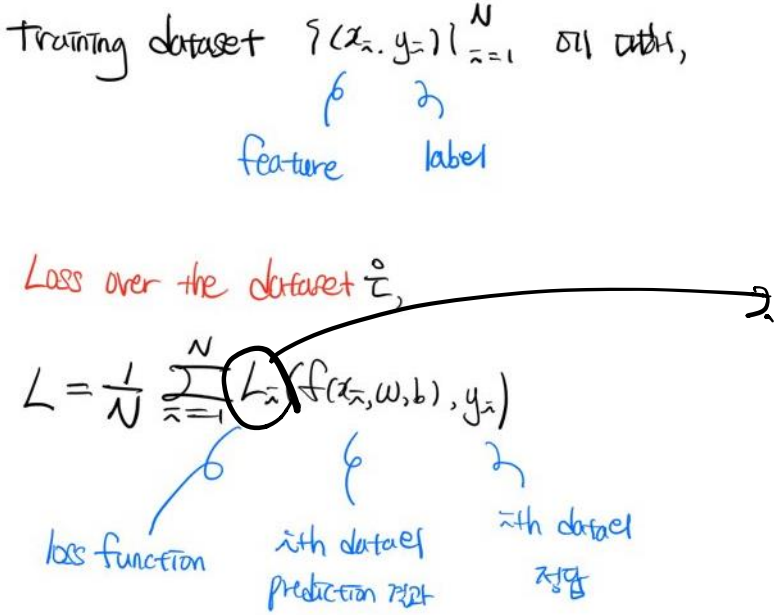
Loss

How do we calculate the Loss Over (Training) Dataset?





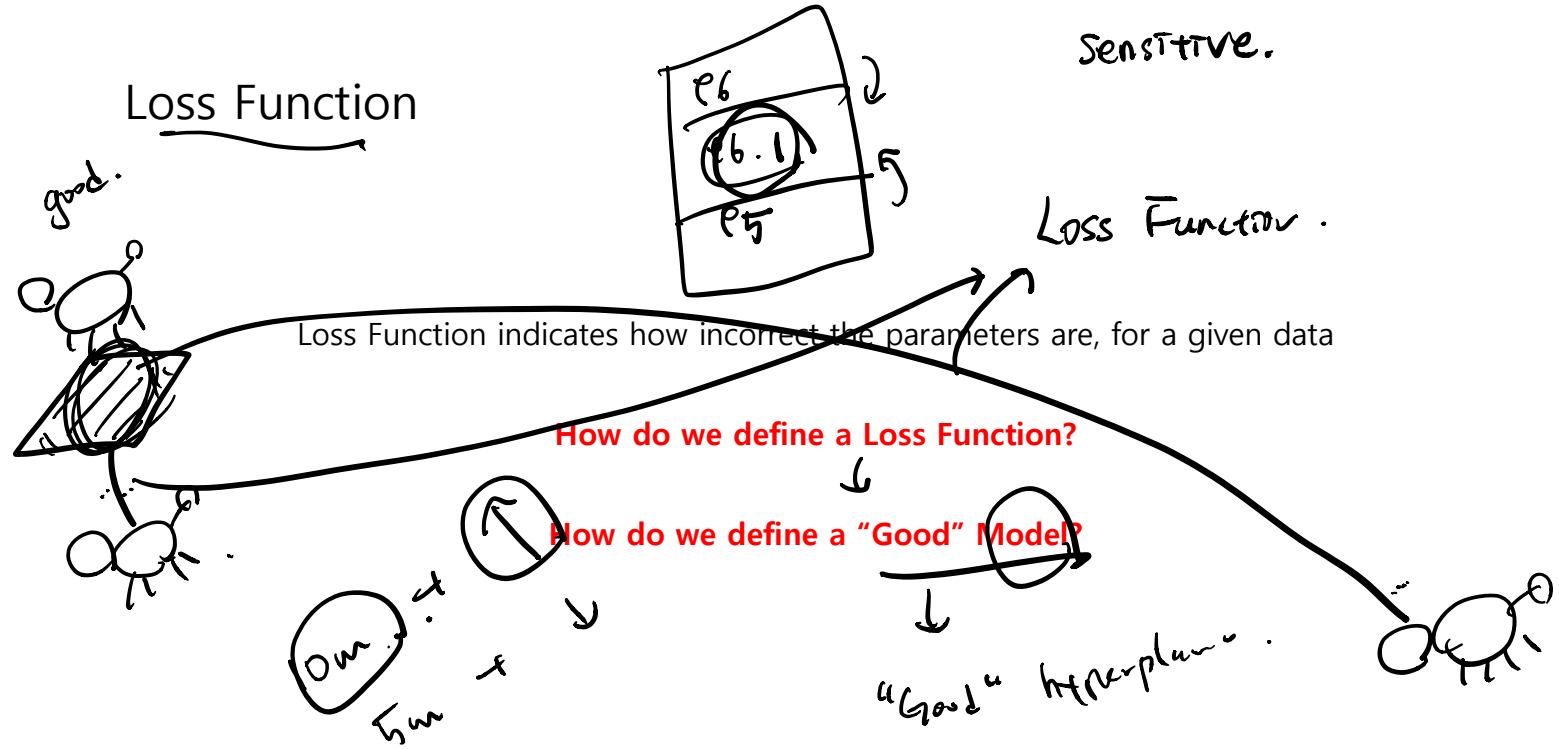
Loss Function



We calculate the Loss via,

Loss Function

Loss Function



Loss Function indicates how incorrect the parameters are, for a given data

How do we define a Loss Function?

How do we define a "Good" Model?

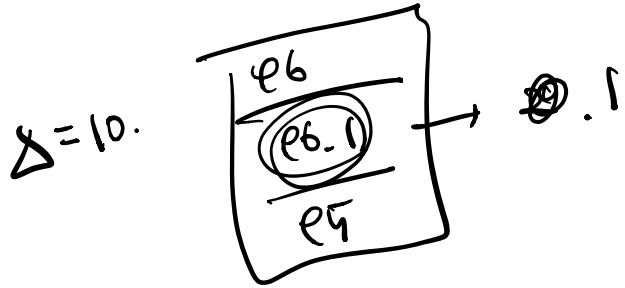
Loss Function

Two Popular Loss Functions,

1. Multiclass SVM Loss

2. Cross-Entropy Loss

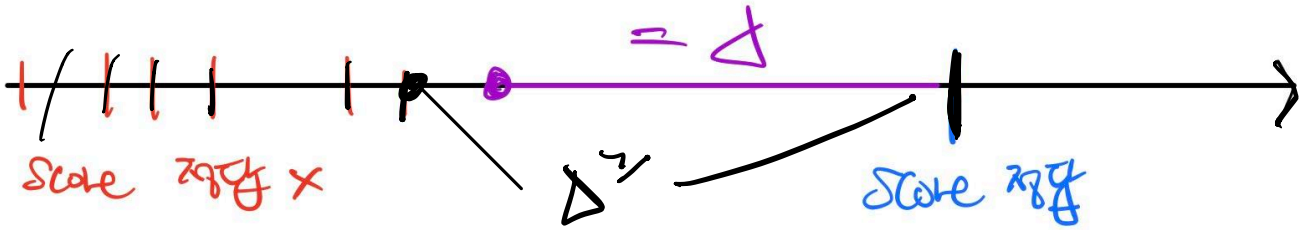
Loss Function : Multiclass SVM Loss



"*The SVM Loss is set up so that*
the SVM wants the correct class for each input
to have a score higher than the incorrect classes by some fixed margin △."

정답 label의 score가 나머지 incorrect label의 score보다 △만큼 크기를 바람.

7 label



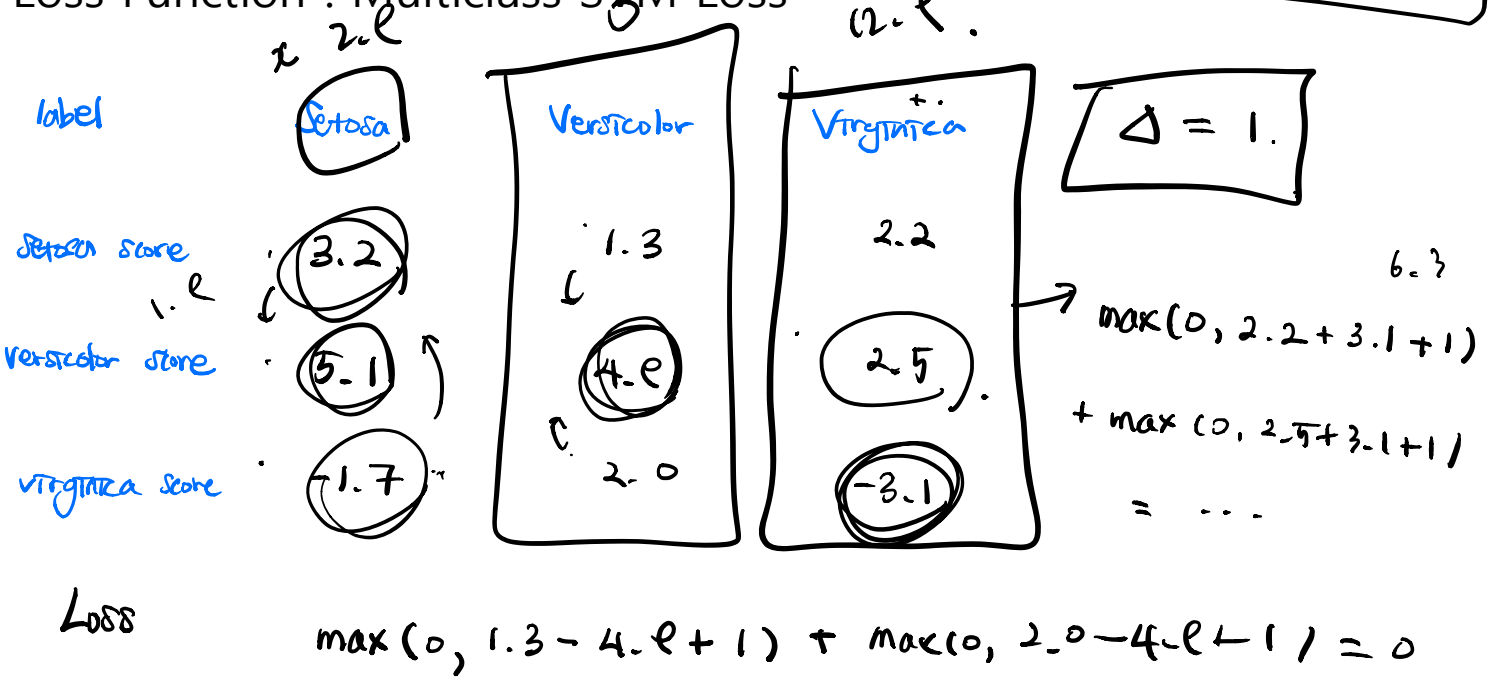
Loss Function : Multiclass SVM Loss

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

$$L = \frac{1}{N} \sum_{i=1}^{N} L_i$$

Loss Function : Multiclass SVM Loss

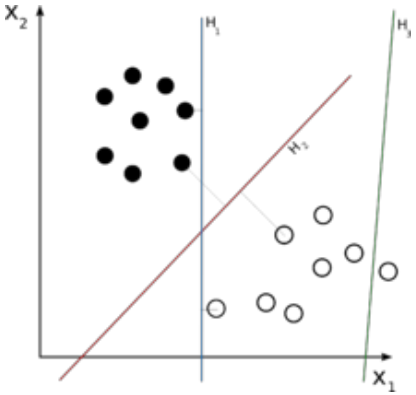
$L = (2.\text{error} + 12.\text{e})/3$

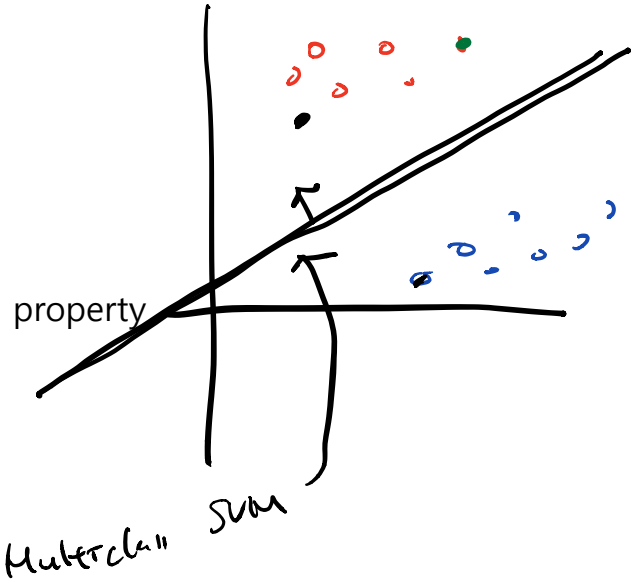


More on SVM (Optional)

Multiclass SVM Loss의 objective는,

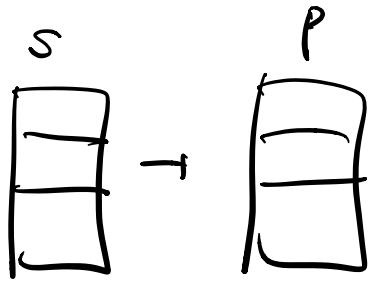
Try to find the decision hyperplane with max-margin property





** For more detail,
https://ko.wikipedia.org/wiki/%EC%84%9C%ED%8F%AC%ED%8A%B8_%EB%B2%A1%ED%84%B0_%EB%A8%B8%EC%8B%A0
cs229.stanford.edu/notes/cs229-notes3.pdf

Loss Function : Cross-Entropy Loss



An approach based on probability

Softmax function maps the scores to probabilities

Softmax function :

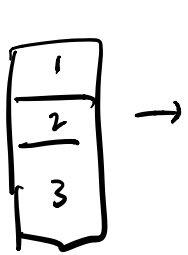
$$\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$$

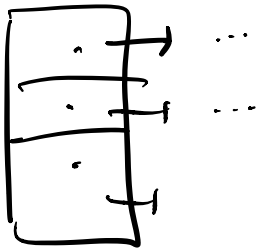
정답 score

e^{score} → sum

e

[0, 1]





$\frac{e^1}{sum} + \frac{e^2}{sum} + \frac{e^3}{sum} = 1$

$\frac{e^{3_y}}{e^1+e^2+e^3}$ 1

Loss Function : Cross-Entropy Loss

Softmax function : $\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$ (정답 score / e^{score}의 sum)

↓

interpret as, $P(y_i | x_i ; W, b) = \frac{e^{f_{y_i}}}{\sum_j e^{f_j}}$

(parameter는 W, b. x_i가 input일 때, y_i가 나올 확률)

$P(y_i | x_i ; W, b) = \cdots$

input.

y_i W, b.

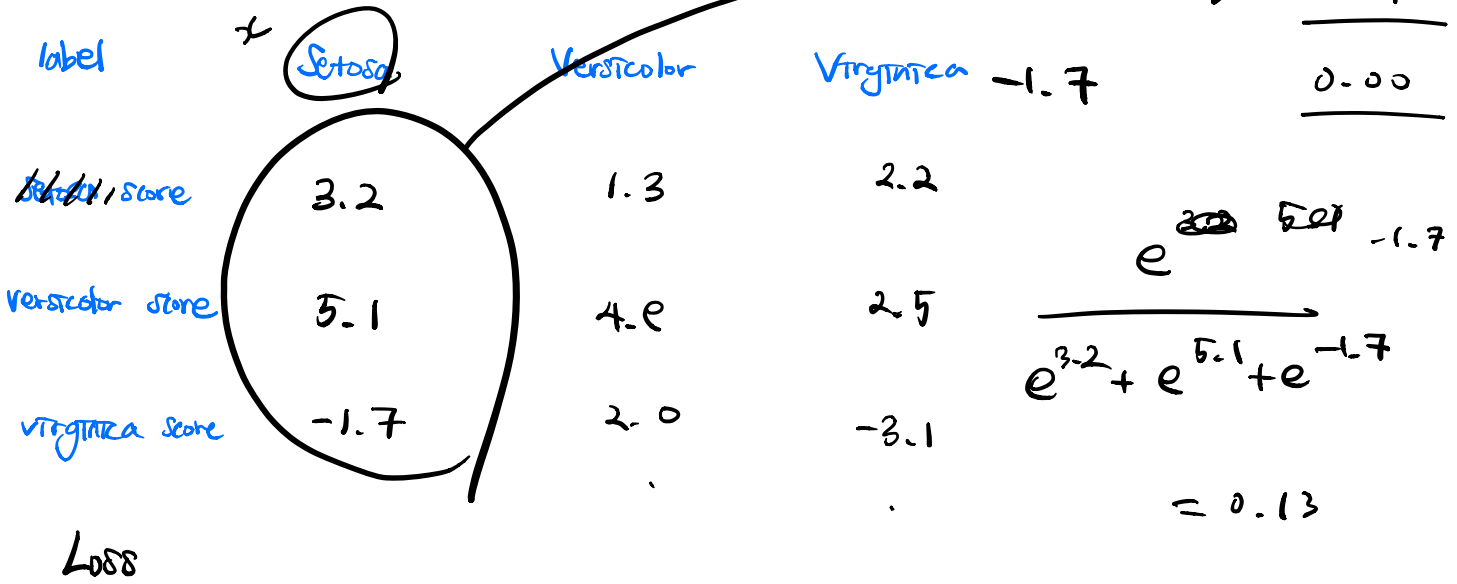
Loss Function : Cross-Entropy Loss

$$p_i = \frac{e^{s_i}}{\sum_j e^{s_j}}$$

$$L_i = -log(p_i)$$

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i$$

Loss Function : Cross-Entropy Loss



Loss Function : Cross-Entropy Loss

In reality,

"Stable" Softmax.

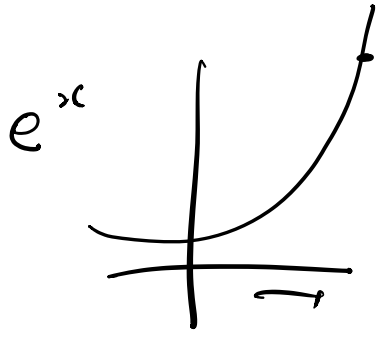
$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e^{a_k}} \times \frac{C}{C}$$

$$= \frac{Ce^{a_i}}{C\sum_{k=1}^{N} e^{a_k}}$$

$$= \frac{e^{a_i+\log(C)}}{\sum_{k=1}^{N} e^{a_k+\log(C)}}$$

$$\log(C) = -max(a)$$

$-max(s)$



$$\frac{e^{s_{y_i}}}{\sum_j e^{s_j}}$$

Loss Function : Cross-Entropy Loss

SVM

CE:

이때, Unknown Parameter W와 b를 어떻게 estimate?

via **MLE(Maximum Likelihood Estimation)**

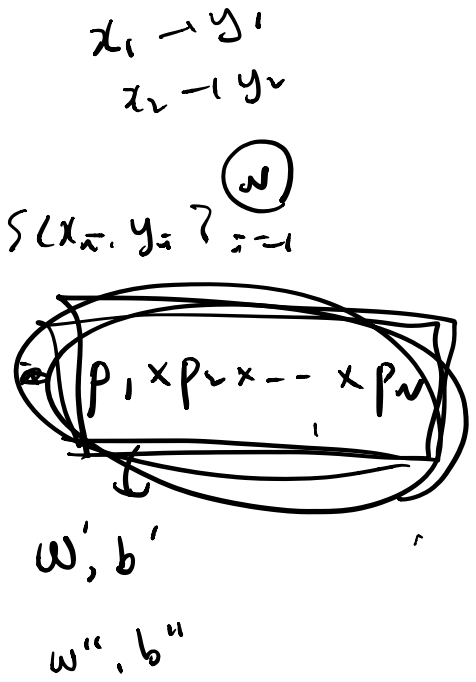
More on MLE (Optional)

conditional probability를 계산할 때,

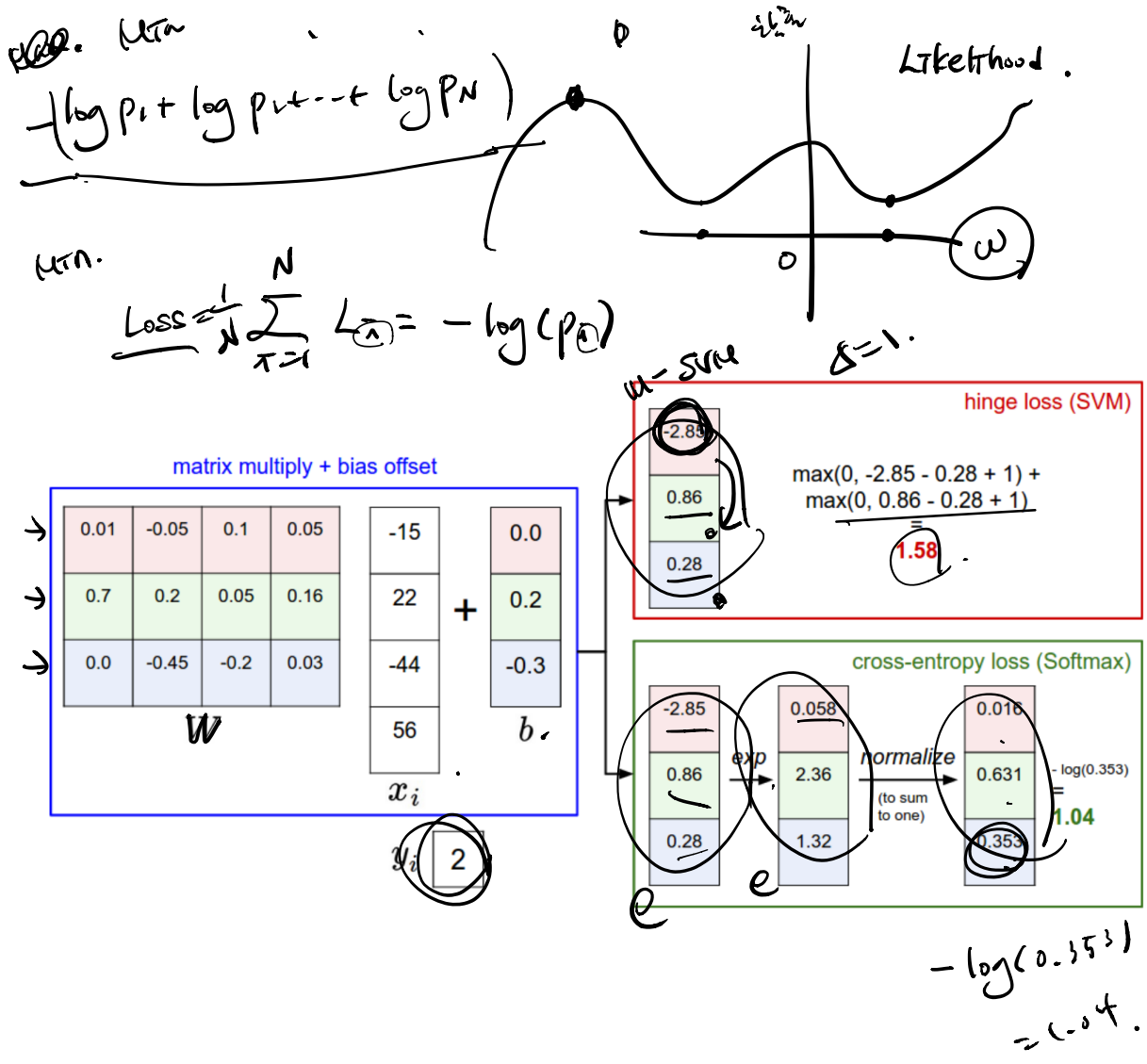
(unknown parameter : W, b) 뽑.

We want to find W, b that maximizes the (Log) Likelihood Function,

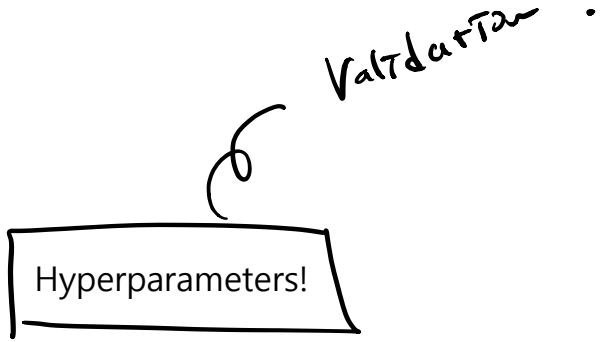
so define the loss as, $L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$



** For more detail, https://en.wikipedia.org/wiki/Multinomial_logistic_regression / http://jaejunyoo.blogspot.com/2018/02/minimizing-negative-log-likelihood-in-kor-3.html
https://ratsgo.github.io/deep%20learning/2017/09/24/loss


Likelihood
0
w
Loss
hinge loss (SVM)
max(0, -2.85 - 0.28 + 1) +
max(0, 0.86 - 0.28 + 1)
1.58
matrix multiply + bias offset
0.01 -0.05 0.1 0.05
0.7 0.2 0.05 0.16
0.0 -0.45 -0.2 0.03
W
-15
22
-44
56
x_i
0.0
0.2
-0.3
b
y_i 2
cross-entropy loss (Softmax)
-2.85
0.86
0.28
exp
0.058
2.36
1.32
normalize
(to sum to one)
0.016
0.631
0.353
- log(0.353)
=
1.04

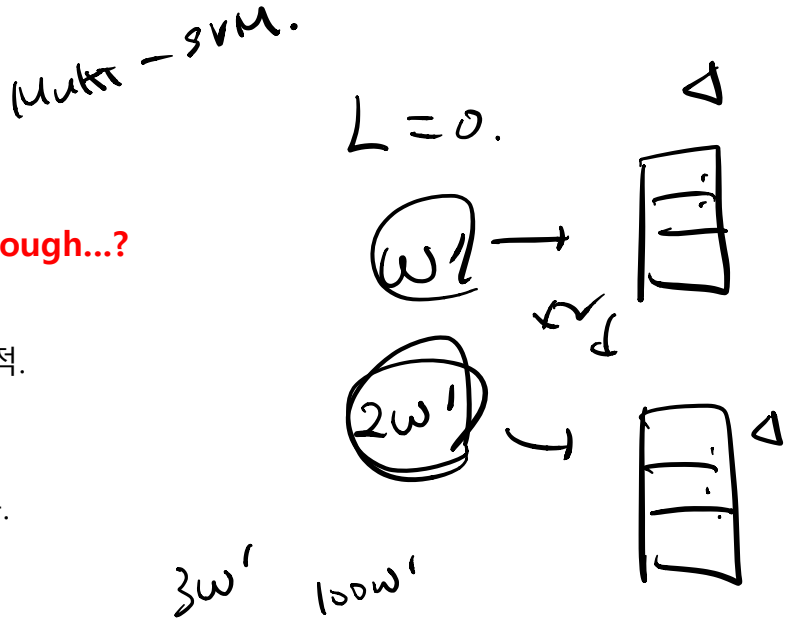
Actually, these are....



Regularization

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i$$

But is this enough...?



loss L을 최소화하는 weight W를 찾는 것이 목적.

이때, L = 0이 되게 하는 W'가 있다면,

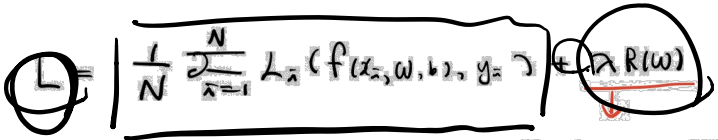
임의의 실수 k > 1에 대해, kW도 L = 0을 만족.

따라서, W is **"NOT" uniquely determined**!

Regularization

among all kW', small ones are preferred (reasons discussed later)

so, add regularization term to discourage large weight



$$R(W) = \sum_k \sum_l |W_{kl}| \quad \text{-- L1}$$

$$= \sum_k \sum_l W_{kl}^2 \quad \text{-- L2}$$

Regularization

$x = (1, 1, 1, 1)$

① $w_1 = (1, 0, 0, 0)$ 1 + 0.

② $w_2 = (.25, \cdots)$ 1 + 0.

더 작은 w가 갖는 이점?

ex) $x = [1, 1, 1, 1]^T$

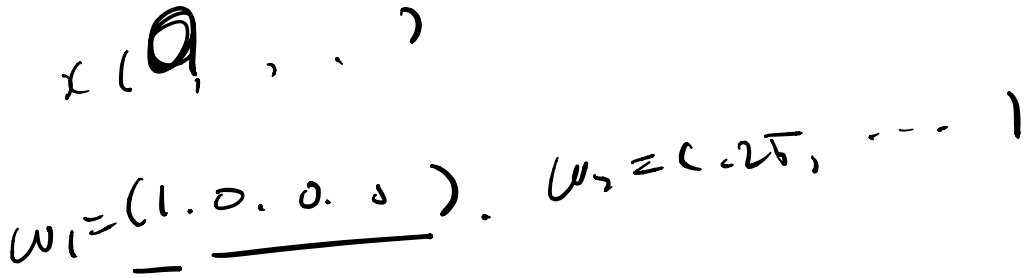
$w_1 = [1, 0, 0, 0]$. $w_2 = [1/4, 1/4, 1/4, 1/4]$

$w_1 x = w_2 x = 1$

but, $R(w_1) = 1 > R(w_2) = 1/4$

$R(w_1) = 1$ $R(w_2) = \frac{1}{4}$.

Regularization



Regularization을 통해,

No input dimension can have a very large influence on the scores all by itself

Regularization

If an input dimension has a very large influence on the scores all by itself,

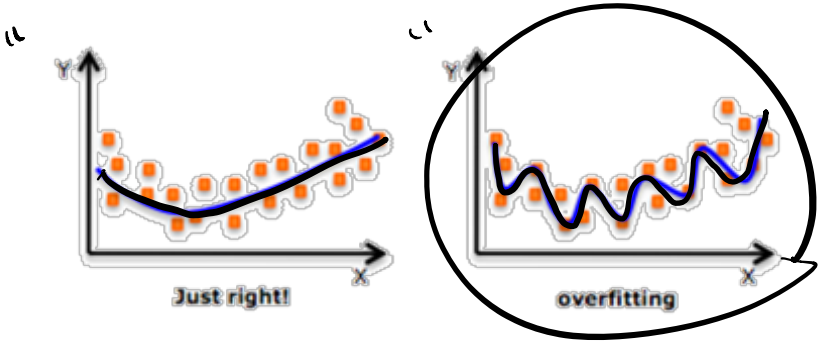
then, model이 training data의 noise까지 학습해버릴 수 있다!

Model performing TOO WELL on training data

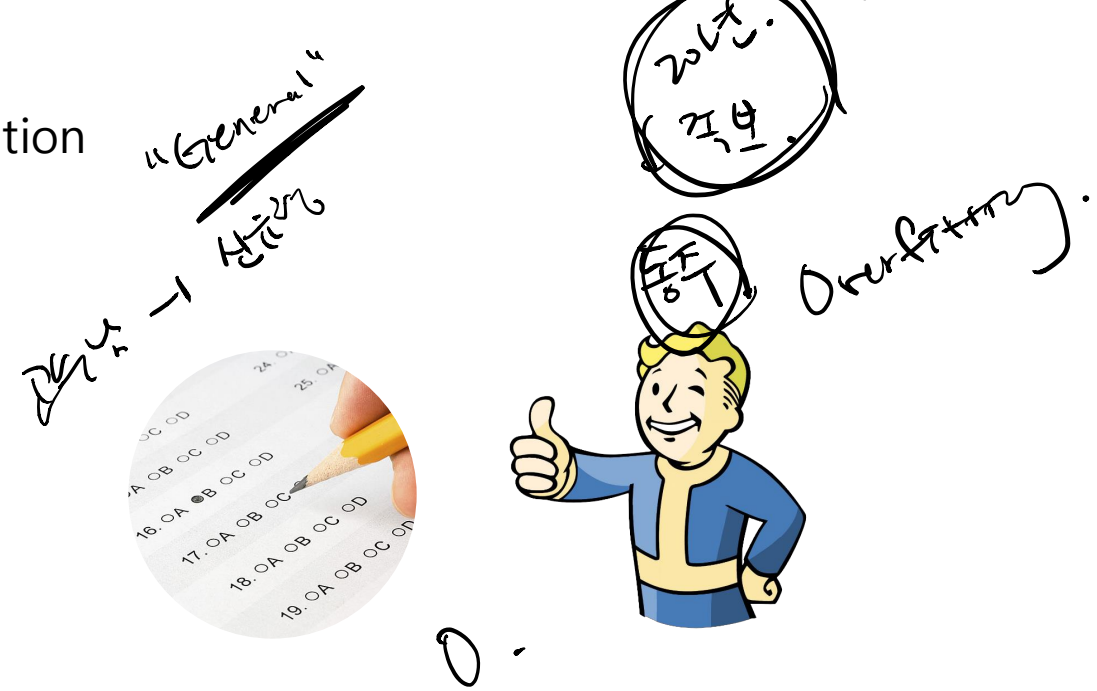
Regularization

We are building a **GENERAL** model for classification

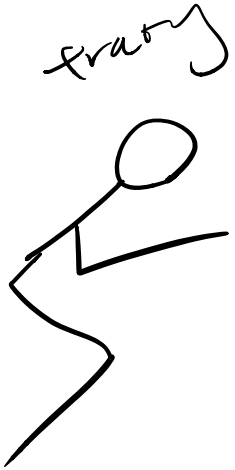
Thus, doing **TOO WELL** on the training data is not desired



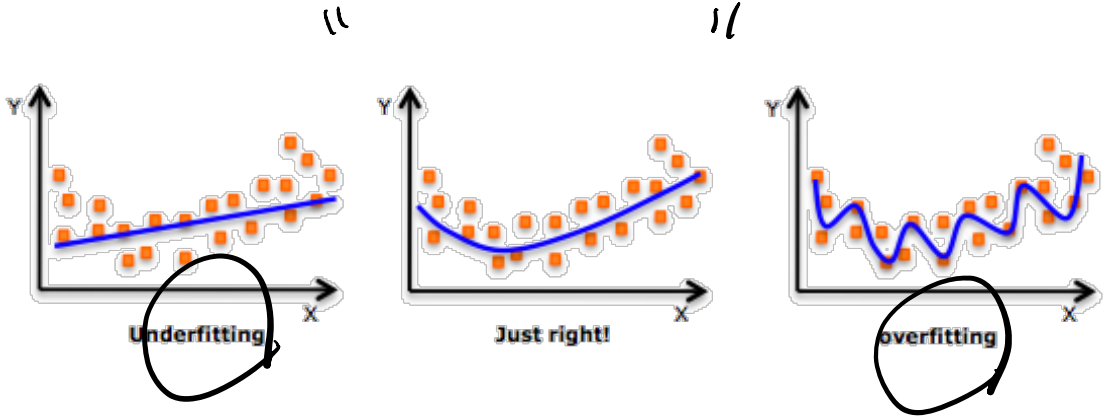
Regularization



Regularization : Overfitting



Regularization



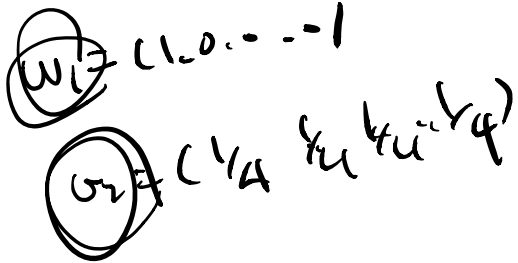
Regularization

$$R(W) = \sum_k \sum_l W_{kl}^2$$

L2

$\times \lambda \pm \rightarrow L$.

L2 Regularizer favors W that is spread out



Regularization : Overfitting

by R(W), we discourage large W = prevent some input dim. from having too much influence on the output

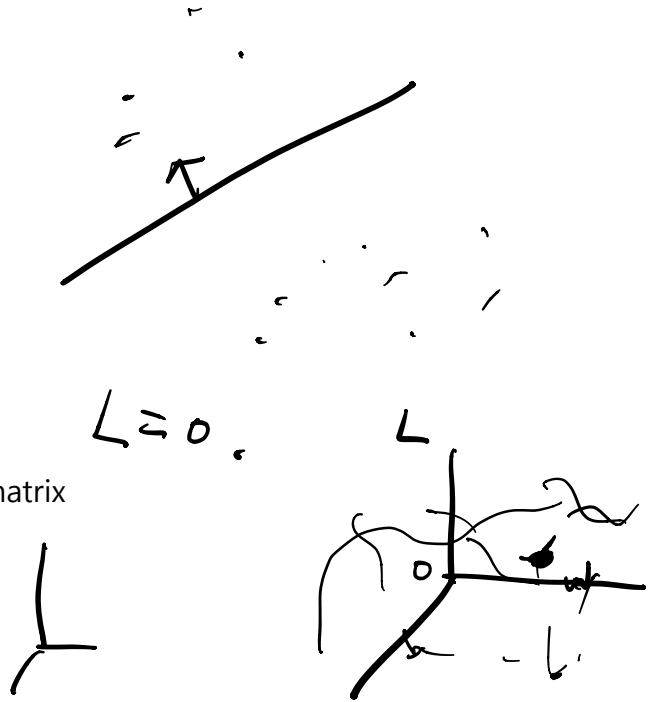
= prevent the model from learning the noises

= prevent overfitting (to some extent)

kw'

Review

1. Loss
2. **Loss Function**
   1. Multiclass SVM Loss
   2. Softmax
3. **Regularization**
   - Why it is needed : to discourage large weight matrix
   - Overfitting



Preview on Next Class

Questions

- **Train이 잘 되었는지 판단할 수치적 척도 필요**

: define a **Loss Function** that quantifies our unhappiness with the scores across the training data

- **Parameter를 update하는 algorithm 필요**

: come up with a way of efficiently finding the parameters that minimize the **Loss Function**

Optimization and Backpropagation

Loss Function은 Scalar Function

Thus, Loss Function의 input W, b에 대해

Gradient를 구해 주어 W, b에서 빼면,

Loss가 줄어드는 방향으로 학습하지 않을까?