



🌙 Type something...

#### 01) Loss Functions and Optimization

01\_SVM

02\_Regularization(규제)

03\_Softmax Classifier

04\_Optimization

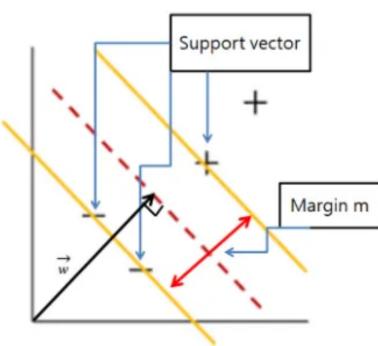
## 01) Loss Functions and Optimization

### 01\_SVM

- Multiclass SVM loss(Hinge loss)

#### SVM의 decision rule

가중치 벡터  $w$ 와 직교하면서 margin이 최대가 되는 선형을 찾는다. 이 말이 받아들이기 어렵다면 위의 그림을 좀 더 수학적으로 표현한 아래의 그래프를 통해 이해해보자.



- support vector: 두 가지 카테고리(짬뽕/짜장)에 각각 해당되는 data set들 ('-'샘플이 모인곳 '+'샘플이 모인곳)의 최외각에 있는 샘플들을 support vector라고 부른다. 이 가장 최 외각에 있는 벡터들을 토대로 margin M을 구할 수 있기 때문에 중요한 벡터들이다.
- margin: support vector를 통해 구한 두 카테고리 사이의 거리를 m이라고 하고 이를 최대화 해야한다.

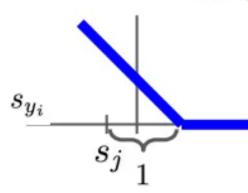
- SVM은 가중치 벡터  $w$ 와 수직이면서 margin이 최대가 되는 선형을 찾는 방식이다.

Suppose: 3 training examples, 3 classes.  
With some  $W$  the scores  $f(x, W) = Wx$  are:

**Multiclass SVM loss:**

"Hinge loss"





cat	<b>3.2</b>	1.3	2.2
car	5.1	<b>4.9</b>	2.5
frog	-1.7	2.0	<b>-3.1</b>

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

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

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 3 - 12

April 11, 2017

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

$$\text{if } s_{y_i} \geq s_j + 1$$

- margin이 1보다 크다면 score가 실제로 작은 것이고, 분류가 잘 된거라고 볼 수 있다.

$$\text{if } s_{y_i} < s_j + 1$$

- 만약 둘이 다른 객체인데 같은 것으로 분류하였다면 loss로 계산해준다. 이 차이가 더 커질수록 Loss는 커진다.

Example numpy code:

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

```
def L_i_vectorized(x, y, W):
    scores = W.dot(x)
    margins = np.maximum(0, scores - scores[y] + 1)
    margins[y] = 0
    loss_i = np.sum(margins)
    return loss_i
```

- Hinge Loss를 코드로 한번 구현해보자.

Numpy 여러 array 비교 최대값 함수 : np.maximum

반면, np.maximum 함수의 기능은 여러 array 사이에서 각 위치의 최대값을 가져오는 함수로, 단일 array 내부에서 최대값을 가져오는 np.max 함수와는 사용 용도가 전혀 다릅니다.



따라서, np.maximum 함수는 여러 array를 동시에 input으로 받습니다.  
이 때, input으로 받는 array들의 차원은 모두 같아야합니다.

np.maximum 함수의 사용 예제 코드를 살펴보겠습니다.

```
a = np.array([1, 3, 5])
b = np.array([6, 4, 2])
np.maximum(a, b)

array([6, 4, 5])
```

```
a = np.array([[2, 10], [3, 7]])
b = np.array([[5, -2], [1, 14]])
np.maximum(a, b)

array([[5, 10],
       [3, 14]])
```

- `np.maximum` 은 두 배열을 비교해서 최대값을 구하기 때문에 array를 반환한다.

$$f(x, W) = Wx$$

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1)$$

E.g. Suppose that we found a  $W$  such that  $L = 0$ .  
Is this  $W$  unique?

**No!  $2W$  is also has  $L = 0$ !**

Fei-Fei Li & Justin Johnson & Serena Yeung

Lecture 3 - 26

April 11, 2017

- 하지만 여기서 문제는 Loss = 0으로 만드는  $W$ 가 유일하지 않다는 것이다.

02\_Regularization(규제)

## Weight Regularization

$$L = \frac{1}{N} \sum_{i=1}^N \sum_{j \neq y_i} \max(0, f(x_i; W)_j - f(x_i; W)_{y_i} + 1) + \lambda R(W)$$

\lambda = regularization strength  
(hyperparameter)

In common use:

data loss

**L2 regularization**

$$R(W) = \sum_k \sum_l W_{k,l}^2$$

**L1 regularization**

$$R(W) = \sum_k \sum_l |W_{k,l}|$$

**Elastic net (L1 + L2)**

$$R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$$

**Max norm regularization** (might see later)

**Dropout** (will see later)

Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 3 - 23

11 Jan 2016

- 이 문제를 해결하기 위해 Regularization이 도입되었다.
- a way of trading off training loss and generalization loss on test set
- 예라는 더 커지만 결과적으로 test set에 대한 퍼포먼스가 더 좋아진다.

03\_Softmax Classifier



## Softmax Classifier (Multinomial Logistic Regression)



scores = unnormalized log probabilities of the classes.

$$P(Y = k | X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}} \quad \text{where} \quad s = f(x_i; W)$$

Want to maximize the log likelihood, or (for a loss function) to minimize the negative log likelihood of the correct class:

$$L_i = -\log P(Y = y_i | X = x_i)$$

cat	<b>3.2</b>
car	5.1
frog	-1.7

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

- softmax는 이전 모두를 위한 딥러닝 시즌2에서 설명했기에 생략하겠다.

## Softmax Classifier (Multinomial Logistic Regression)



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

Q: What is the min/max possible loss  $L_i$ ?

cat  
car  
frog

<b>3.2</b>
5.1
-1.7

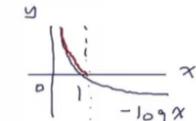
unnormalized log probabilities

<b>24.5</b>
164.0
0.18

normalize

<b>0.13</b>
0.87
0.00

probabilities

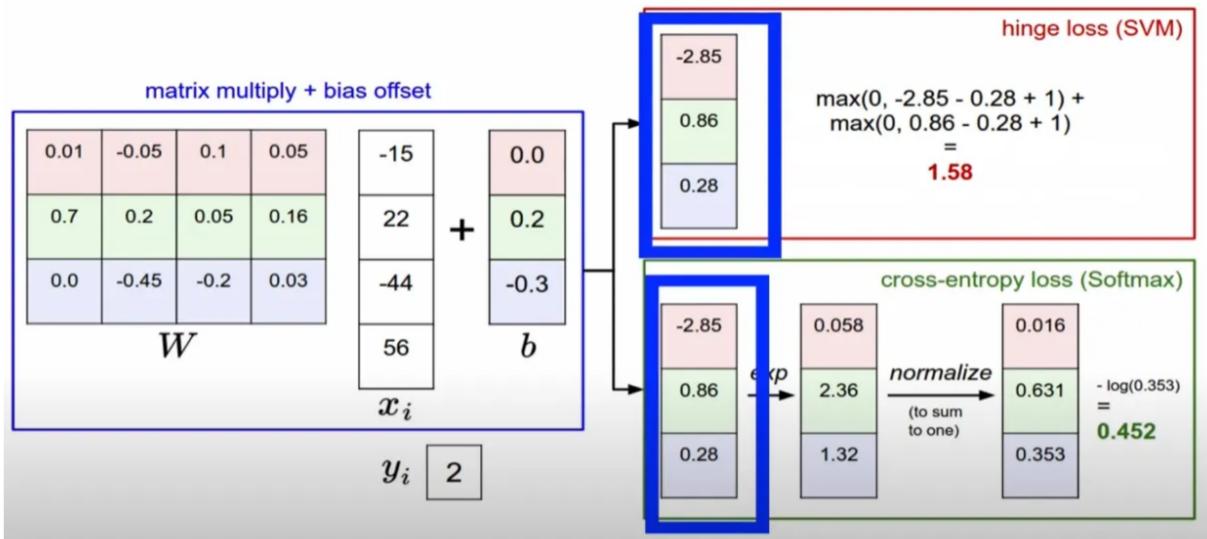


Fei-Fei Li & Andrei Karpathy & Justin Johnson

Lecture 3\_35

11 Jan 2016

- Loss min = 0, Loss max = 무한대, svm과 동일하다.



- 비교해보면 다음과 같다.

### 04\_Optimization

- Loss를 minimize 하는 weight를 찾아가는 과정이다.



## In summary:

- Numerical gradient: approximate, slow, easy to write
- Analytic gradient: exact, fast, error-prone

=>

**In practice:** Always use analytic gradient, but check implementation with numerical gradient. This is called a **gradient check**.

- Analytic gradient는 나중에 배우는 backpropagation이라는 방식으로 계산된다.

## Gradient Descent

```
# Vanilla Gradient Descent

while True:
    weights_grad = evaluate_gradient(loss_fun, data, weights)
    weights += - step_size * weights_grad # perform parameter update
```

Fei-Fei Li & Andrej Karpathy & Justin Johnson

Lecture 3 -63

11 Jan 2016

- weight를 update하는 코드는 다음과 같다.

## Mini-batch Gradient Descent

- only use a small portion of the training set to compute the gradient.

```
# Vanilla Minibatch Gradient Descent

while True:
    data_batch = sample_training_data(data, 256) # sample 256 examples
    weights_grad = evaluate_gradient(loss_fun, data_batch, weights)
    weights += - step_size * weights_grad # perform parameter update
```

Common mini-batch sizes are 32/64/128 examples  
e.g. Krizhevsky ILSVRC ConvNet used 256 examples

- Mini-batch Gradient Descent
- 데이터 셋이 너무 크면 mini-batch로 나누어서 update를 진행한다.

