cs231n lecture 3

1. Loss function (Multiclass SVM loss)

- Loss function 에 여러 종류가 있지만 그 중 Multiclass SVM Loss 에 대해 알아보자.

- Multiclass SVM Loss

  * 각 class 에 점수를 매겨서 점수(score)가 정답과 얼마나 차이가 나는지만 관심을 두는 함수

  * f(x,W) = Wx 로 class 분류

cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

이 classifier는 자동차에 대해서는 잘 분류를 하지만 다른 그림은 분류를 잘 하지 못함

  * 이 classifier이 얼마나 분류를 잘하는지 알아보기 위해 Loss function을 이용하여 수치화

*

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

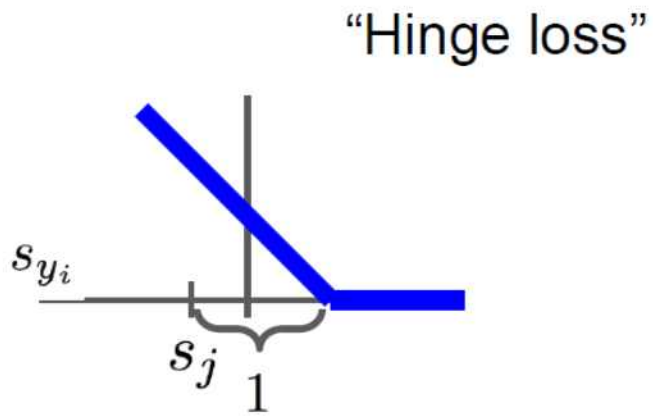
- L i는 사용할 loss function
- f(x_i, W)는 분류기를 사용해서 나온 class score
- y_i는 실제 class 정답 값
- N은 class 수

*

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

- s_j는 분류기를 통해 예측한 각 class score
- s_y_i는 해당 클래스의 정답 점수
- 1은 safety margin
  예측 값과 정답 값에 대한 상대적인 차이를 주기 위해 설정

*



* Loss 값 계산

cat	**3.2**
car	5.1
frog	-1.7
Losses:	2.9

the SVM loss has the form:

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

= max(0, 5.1 - 3.2 + 1)
+max(0, -1.7 - 3.2 + 1)
= max(0, 2.9) + max(0, -3.9)
= 2.9 + 0
= 2.9

* car score이 변하게 된다면 loss의 값은 어떻게 되는가?
-> score 간에 상대적인 차이가 중요하므로
loss의 값에는 크게 영향을 미치지 않는다.

* loss값의 최솟값과 최댓값을 구하면?
-> 최솟값은 0 , 최댓값은 ∞

* 초기에 W가 0에 가까우면 모든 score의 값은 0과 비슷하다.
이때 loss값을 구하면?
-> Loss는 '(클래스) - 1'이 된다. 이는 디버깅할 때 유용한 팁

* L_i 값을 구하는데 합을 구할 때 j≠y_i가 아닌, j를 포함한 모든 값을 다 더하게 되면?
-> j를 포함한 모든 값을 다 더하게 되면
Loss의 값이 '0'이 아닌 '1'이 정답 값이다.

* Loss를 계산하는데, 합 대신 평균을 사용하면
Loss값은 어떻게 되는가?
-> Loss 값을 구하는데는 크게 상관이 없다.
우리가 관심이 있는 것은 정답 클래스의 점수와
그렇지 않은 클래스의 차이이기 때문이다.

* 우리가 Loss function을 이용해서 L을 구했을 때
L=0인 W의 값이 단 하나인가?
-> 또 다른 W의 값이 존재한다.

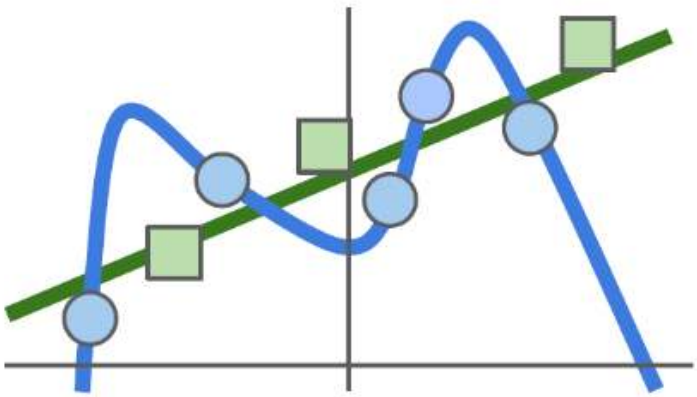
* 수많은 W의 값들 중에서 우리가 선택해야하는 W의 값은?
-> train data를 이용해서 어떤 분류기를 찾고 (즉, W의 값을 찾고)
이 분류기(W)를 test data에 적용했을 때
성능이 잘 나오는 것을 고르면 된다.

* 어떻게 우리가 원하는 W의 값을 찾는가?
-> Regularization 개념 등장

2. Regularization

- training set에 model이 완벽하게 fit하지 못하도록 모델의 복잡도에 페널티를 부여하는 것

-



* train data로 model을 만들었을 때는 파란색 그래프이지만,
  우리가 원하는 것은 초록색 그래프일 수 있음

* 우리가 원하는 model은 좀 더 단순화한 overfitting이 일어나지 않는 model이고, 이를 만들기 위해 사용하는 것이 Regularization

*

$$L(W) = \frac{1}{N}\sum_{i=1} L_i(f(x_i, W), y_i) + \lambda R(W)$$

$\lambda R(W)$
위 부분이 Regularization

* Regularization 종류: L1 regularization, L2 regularization,
  Max norm regularization, Dropout, Batch Normalization 등

3. Loss function (Softmax)

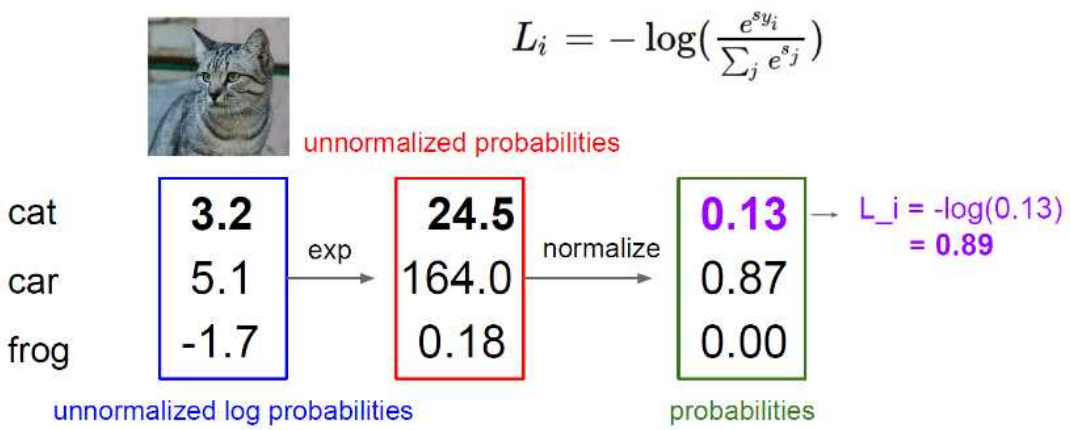
- Multi class SVM 같은 경우 정답 score과 오답 score의 차이에만 관심을 보였다면, Softmax의 경우 그 차이를 모두 수치화하여 score에 대한 해석

- 전체적인 과정

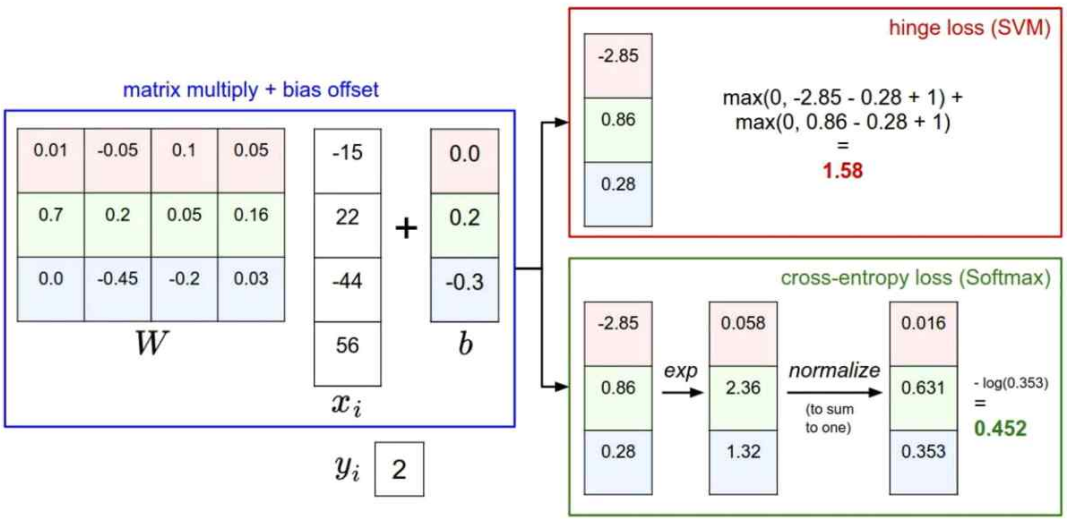
  먼저 각 클래스에 대해 score를 먼저 구한다.

  모든 클래스의 score를 구하고 각 클래스마다 score를 구하여 확률에 대한 식으로 바꾸어 준다. (unnormalized probabilities)

  정규화를 통해 score값을 0 ~ 1사이로 바꾸어준다.



-



-

Recap

How do we find the best W?

- We have some dataset of (x,y)
- We have a **score function:** $s = f(x; W) \overset{\text{e.g.}}{=} Wx$
- We have a **loss function**:

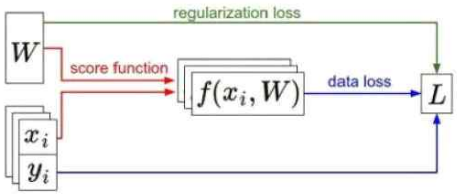
Softmax

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

SVM

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

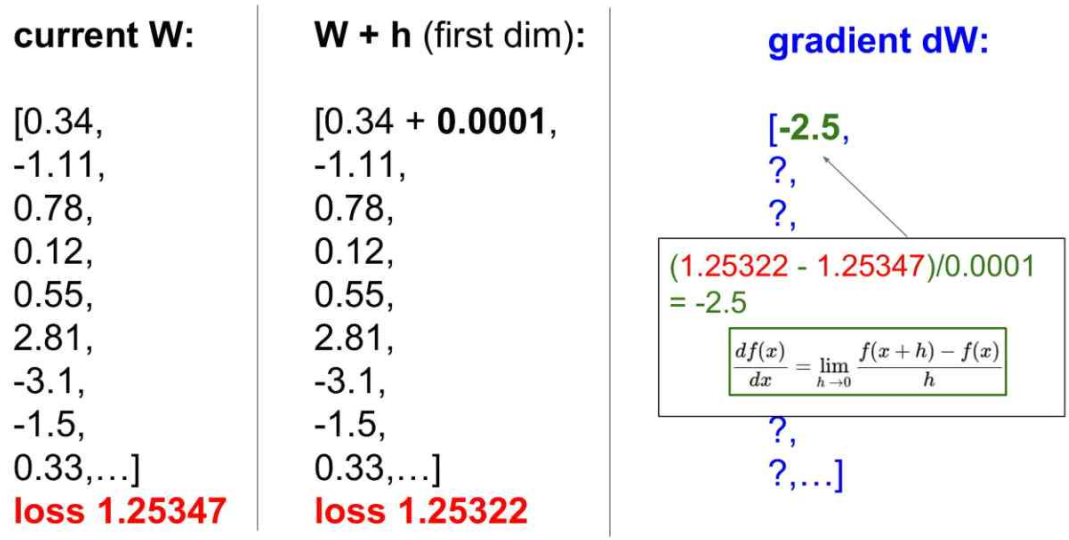
$$L = \frac{1}{N}\sum_{i=1}^{N} L_i + R(W)$$ Full loss

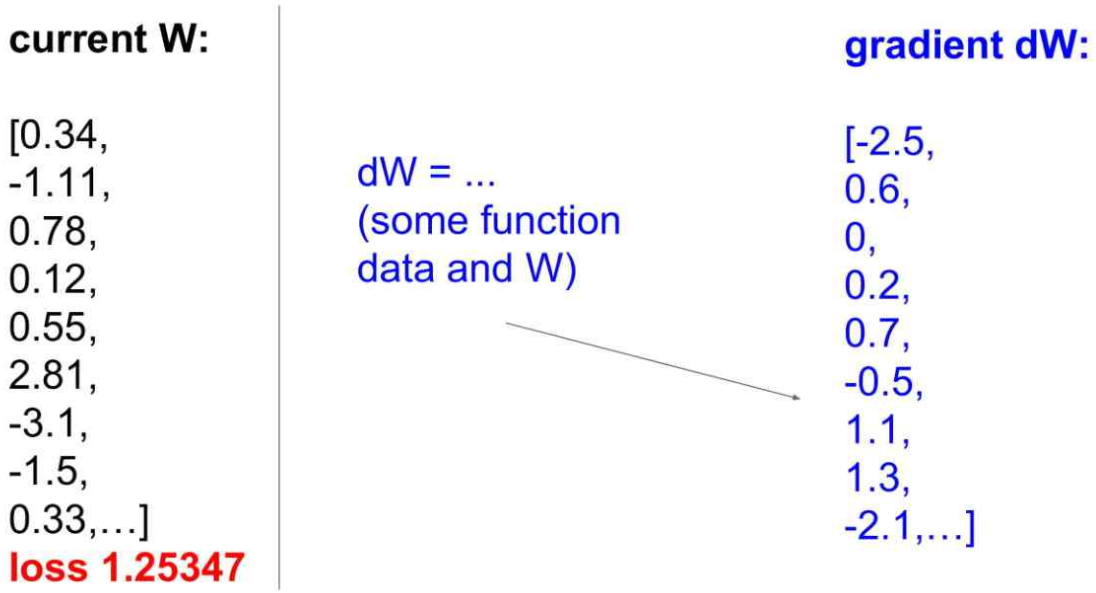


- 궁극적으로 우리가 원하는 것은 최적의 W를 찾는 것

4. Optimization

- Gradient Descent : 처음에 weight를 초기화시키고, loss와 gradient를 계산하여 weight의 값을 gradient 반대 방향으로 update를 계속 시켜주는 것. 모든 data에 대하여 일일이 이 작업을 하기에는 연산량이 너무 많다.





current W:

[0.34,
-1.11,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25347

W + h (second dim):

[0.34,
-1.11 + **0.0001**,
0.78,
0.12,
0.55,
2.81,
-3.1,
-1.5,
0.33,…]
loss 1.25353

gradient dW:

[-2.5,
0.6,
?,
?,

(1.25353 - 1.25347)/0.0001
= 0.6

$$\frac{df(x)}{dx} = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

?,…]

Gradient Descent

```
# Vanilla Gradient Descent

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

- Stochastic Gradient Descent (SGD) :
minibatch를 활용해 Gradient Descent를 사용, minibatch에는 보통 2^n을 사용

-