Loss function and Optimization

1. Loss function : tells how good our classifier is

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

$f(x_i, W)$: 예측 결과 / y_i: 실제 (ground truth)

① Multiclass SVM loss

$s = f(x_i, W)$; $(s_1, s_2, \cdots, s_K)$ — K개의 클래스 존재, N개의 데이터셋 존재 가정

(s: score)

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

$L \in [0, \infty)$

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

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

	x_1	x_2	x_3
cat	**3.2**	1.3	2.2
car	5.1	**4.9**	2.5
frog	-1.7	2.0	**-3.1**

$L_1 = \max(0, 5.1-3.2+1) + \max(0, -1.7-3.2+1) = 2.9 + 0 = 2.9$

$L_2 = \max(0, 1.3-4.9+1) + \max(0, 2.0-4.9+1) = 0 + 0 = 0$

$L_3 = \max(0, 2.2-(-3.1)+1) + \max(0, 2.5-(-3.1)+1)$

$= 6.3 + 6.6 = 12.9$

$\rightarrow L = \frac{1}{3}(2.9 + 0 + 12.9) = 5.27$

Regularization term : prevent the model from doing too well on training data So we don't fit noise in the data

(1) $\lambda R(W)$

- λ: regularization strength ↓ hyperparameter

- L1 regularization : $R(W) = \sum_k \sum_l |W_{k,l}|$
- L2 regularization : $R(W) = \sum_k \sum_l W_{k,l}^2$
- Elastic net : $R(W) = \sum_k \sum_l \beta W_{k,l}^2 + |W_{k,l}|$

ex) $x = [1,1,1,1]$ $w_1 = [1,0,0,0]$ $w_2 = [.25, .25, .25, .25]$

↳ $W^T x$ 결과는 같지만 L2는 w_2를 더 선호

(2) Dropout

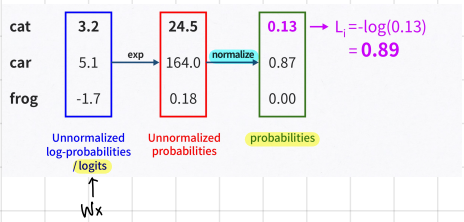
(3) Batch normalization

(4) Stochastic depth, fractional pooling, etc

→ 왜 사용?

- Express preferences over weights
- Make the model simple so it works on test data
- Improve optimization by adding curvature

② Softmax Classifier



$L_i = -\log P(Y=y_i | X=x_i)$; correct label의 probability에 (-log) 씌움. 커지면 loss ↓

$\Rightarrow L_i \in [0, \infty)$

$$L = \frac{1}{N}\sum_i L_i + \lambda R(W)$$

2. Optimization

① Random Search : expensive, 좋은 성능 ⊗

② Follow the slope

(1) Numerical gradient

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

(2) Analytic gradient (in practice)

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

⇨ want $\nabla_W L$ $\left(\frac{\partial L}{\partial W}\right)$ $\nabla_W L = \sum_{j \neq y_i} \nabla_W L_i(x_i, y_i, W) + \lambda \nabla_W R(W)$

③ Gradient Descent

→ Weight += $\underbrace{-\text{step\_size}}_{lr} * \underbrace{\text{weights\_grad}}_{\frac{\partial L}{\partial W}}$

→ Stochastic Gradient Descent : 점점 데이터셋이 크면 expensive ⇒ minibatch를 사용해서 weight update