Lecture 3) Loss Functions and Optimization

학습 방법(W를 정하는 방법)

- **Loss function**
  - W가 얼마나 나쁜지를 아는 것
  - 주어진 값이 얼마나 틀리는지에 대한 정도를 표현
  - Loss function의 일반적인 식

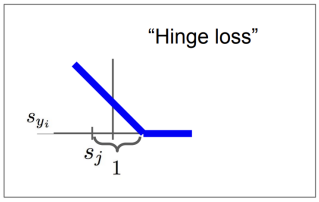
$$L = \frac{1}{N}\sum_{i} L_i(f(x_i, W), y_i)$$

    - x: input
    - y: label (predict, answer)
    - input x와 weight W를 넣어서 y를 예측한 후, 정답 y와 비교하는 식
    - 함수 L: f의 예측 결과와 정답 y를 통해 얼마나 잘 못 맞췄는지를 나타낸다. (작을수록 좋음)
  - ex) binary SVM loss, multi-class SVM loss(support vector machine)
    - Binary SVM loss: 두 클래스,(positive, negative)
- **Optimization**
  - Loss function을 줄이는 것

multi-class SVM loss(support vector machine)

- 식

Multiclass SVM loss:



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

  - S: 각 클래스 별 예측 score(classifier result)
  - S_{y_i} : score of true class(ground truth)
  - 1: safety margin(score의 절대적인 값보다는 W에 대한 scaling하는 역할을 한다.)
- 특징
  - 정답 값의 스코어가 다른 레이블로 예측한 스코어보다 월등히 크면 좋다.
  - 다른 레이블의 스코어와 비슷하면 loss가 크다.
  - 예측값이 정답에 가까울수록 loss값은 0에 가까워진다.

▼ 예시(고양이 분류)

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

- Q. Loss function에 제곱을 한다면?
  - 다른 loss function이 된다.
  - 제곱을 하면, 나쁜게 더 나쁘게 보이게 된다. 안좋은 것을 더 안좋게 보기위해 제곱을 사용한다.
  - 알고리즘이 어떤 에러를 다루느냐에 따라 다르게 적용할 수 있다.

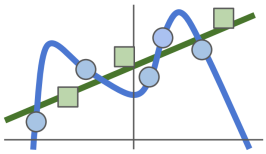
Regularization

- 식

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\text{Data loss}} + \underbrace{\lambda R(W)}_{\text{Regularization}}$$

Data loss: Model predictions should match training data

Regularization: Model should be "simple", so it works on test data



  - λ: 일반화 강도를 의미하는 하이퍼파라미터
- 학습 데이터가 아닌, 테스트 데이터를 더 잘 맞추는 모델(W, weight)을 만들기 위해 사용한다.
- Penalty. 모델의 복잡도를 낮춰주는 역할을 한다. 일반화 역할을 한다.
  - Occam's Razor: "단순한게 최고"
- 종류
  - L2 regularization(= weight decay, euclidean norm, sqared norm, $1/2$ squared norm)
  - L1 regularization: W의 sparsity를 높일 수 있다.
  - Elastic net regularization(L1+L2)
  - Dropout
  - Batchnorm
- W는, x가 output과 얼마나 연관이 있는지를 알려준다.
- L1 vs. L2
  - L1 regularization은 weight에서 0이 아닌 요소들의 개수가 많을 수록, 복잡하다고 판단한다.
    - 0이 많으면 덜 복잡
    - 값을 치중하게 만드는 역할을 한다.
  - L2 regularization은 weight에서 0의 개수에 따라 모델의 복잡도를 결정한다.
    - 값이 퍼져있으면 덜 복잡

- weight가 한 쪽에 치중되어 있으면 복잡하다고 판단
- 값을 퍼뜨리는 역할을 한다.

→ Regularization term 역할: 모델 파라미터에 패널티를 줘서, 더 간단한 가설(모델)을 만들 수 있게 한다.

Softmax loss

- Softmax classifier = Multinomial Logistic Regression
- score를 class에 대한 확률분포로 해석할 수 있게 해준다.
- Softmax 함수는 0~1 사이의 값으로 만들고, 모든 score의 합이 1이 되도록 한다.
- Softmax loss는 0~무한대 사이의 값이다.
  - $loss = -log(score)$
- 식

scores = unnormalized log probabilities of the classes.

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

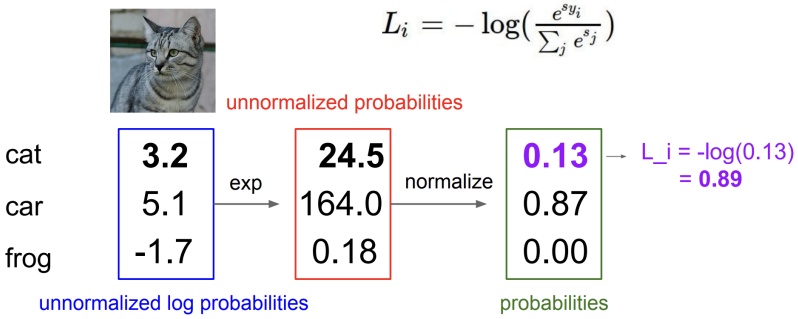
Softmax function

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

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

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

- 과정



Loss 간 차이점

- SVM loss

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

  - 정답레이블의 스코어가 margin보다만 높으면 된다.
- Softmax loss

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

  - 확률적으로 정답에 양의 무한대의 score, 오답에 음의 무한대 score을 만족해야 최적의 해이다.
  - 더욱 최적을 만들고자 한다.(양/음의 무한대에는 끝이 없으므로)

- Data에 피팅하는 것과 모델을 심플하게 만드는 것 사이에 trade-off가 존재한다.

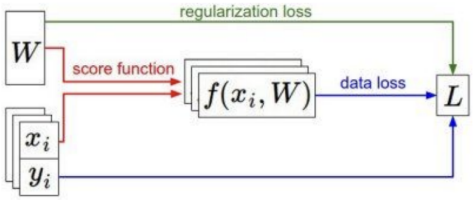
Softmax

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

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



Optimization

- slope(loss func)를 내려간다고 가정할 때, 반복적으로 값(parameter vector)을 옮겨보고,
  더 최적인 방향으로 움직이기
- Random search: 무작위 수를 사용하여 최적값 찾기
- 기울기 = 미분값
- 미분값(기울기, Gradient)을 통해, 어느 방향이 loss를 낮출 수 있는 방향인지 알아낼 수 있다.(Descent)
- **Gradient Descent**
  - loss function의 기울기가 감소하는 방향으로 weight를 움직이는 것
  - Step size/Learning rate: hyper-parameter. gradient 계산 후, 그 방향으로 얼마나 움질일지에 대한 것을 결정

SGD(Stochastic Gradient Descent)

- SGD: 모든 데이터의 loss의 평균을 내서 하나의 step을 이동한다.
  - 문제) 데이터 수가 매우 많을 수 있다. → 다음 step을 가기위해, 많은 연산 필요 → 매우 느림
- (Random) Mini-batch: 한 step을 이동하기 위해, 모든 데이터에서 N개의 샘플만을 사용한다.
  - N: 32 or 64 or 128 등

- raw 이미지 픽셀값을 입력으로 받아서 선형분류하는 것은 매우 성능이 좋지 않다.
- 따라서, 이미지의 특징을 뽑아낸 후, 특징 벡터를 입력 값으로 쓰는 방법이 있다.
  - ex) color histogram, HoG, Bag of Words