Loss Functions and Optimization

Overview

1. Loss function
    a. Function to quantify the badness
    b. Tell how good the current classifier is
2. Optimization
    a. Choose the least bad W

Loss Function

1. Multi-class SVM Loss (Hinge Loss)
    a. Compare the scores from correct category with the incorrect categories
        i. correct category score > others: loss = 0
            - comparison with safety margin
        ii. correct category score < others: loss = sum of incorrect scores
    b. Care the relative differences between categories, not the exact scores
    c. Questions
        i. What happens to loss if car scores change a bit?
            - not change b/c car score is already larger than others
            - loss = 0
        ii. What is the min and max possible loss?
            - min is 0 and max is infinity
        iii. At initialization W is small so all s = 0, what is the loss?
            - number of classes - 1 b/c the safety margin is + 1
            - in detail, loop over incorrect classes and the two scores will be about the same so get a loss of one (safety margin)
        iv. What if the sum was over all classes? (including j = y_i)

- loss increases by one

v. What if we used mean instead of sum?

- doesn't change, just rescale the scores

vi. What if we used the square term instead?

- can occur different result

vii. Suppose that we found a W such that L = 0, is this W unique?

- no (e.g., 2W)

d. Linear vs Square

i. Linear

- 클래스 사이의 차이를 그대로 반영함
- 제곱하는 것에 비해서 차이가 적게 반영됨

ii. Square

- 클래스 사이의 차이를 제곱하여 반영함
- 작은 차이도 더 많이 반영됨

2. General Loss Function

a. L = data loss + $\lambda R(W)$

i. data loss

- model predictions should match training data

ii. regularization (R(W))

- model should be simple, so it works on test data
- complex data should overcome the penalty

b. Regularization

i. L2 regularization

- Euclidean norm (square norm)

ii. L1 regularization

- encourage sparsity

iii. Elastic net (L1 + L2)

    iv. Max norm regularization
    v. Dropout
    vi. Fancier
        - batch normalization
        - stochastic depth
  c. Example
    i. x = [1, 1, 1, 1], W1 = [1, 0, 0, 0], W2 = [0.25, 0.25, 0.25, 0.25]
    ii. L2 regularization
        - prefer W2
        - L2는 모든 값들이 고르게 퍼져있을 때 더 단순하다고 판단함
    iii. L1 regularization
        - prefer W1
        - L1은 0이 많을수록 더 단순하다고 판단함

3. Softmax Classifier (Multinomial Logistic Regression)
  a. Scores are the unnormalized log probabilities over the classes
    i. 0 ≤ scores ≤ 1 b/c scores are probabilities
    ii. sum of scores = 1
  b. L = - log of the probability
  c. Questions
    i. What is the min and max possible loss?
        - min is 0 and max is infinity
        - these are theoretical values
    ii. Usually at initialization W is small so all scores = 0, what is the loss?
        - log of C (the number of classes)
    iii. Suppose I take a datapoint and I jiggle a bit (changing its score slightly), what happens to the loss in both cases?
        - Hinge

    - 클래스 사이의 차이값만 중요하고 각각의 값은 중요하지 않으므로 차이가 없음
    - 가장 큰 값이 옳은 클래스라면 그대로 끝이고 이후의 값에 관심을 갖지 않음
- Softmax
    - 옳은 클래스로 분류될 확률이 제일 크도록 끊임없이 계산함
    - 각각의 값이 달라짐에 따라 결과에도 변화가 생기고 옳은 클래스로 분류될 확률이 이미 가장 크다고 해도 이후의 값에 꾸준히 관심을 가짐

Optimization

1. Metaphore
    a. A person walking around the valley
        i. every point corresponds to some setting of the parameter W
        ii. each height is equal to the loss
    b. Find the bottom of this valley
2. Algorithms
    a. Random search
        i. bad algorithm, never use this
    b. Follow the slope
        i. cannot see the direct path at the start, but we can find which way can leads us down the hill
        ii. calculate the gradient
            - gradient is a vector of partial derivatives along each dimension
            - slope in any direction is the dot product of the direction with the gradient
            - direction of steepest descent is the negative gradient
        iii. numerical gradient vs analytic gradient
            - numerical gradient
                - gradient check 하는 용도로 사용됨
                    - check implementation with numerical gradient

        - approximation, slow, easy to write 하다는 특성을 가짐
    - analytic gradient
        - exact, fast, error-prone 하다는 특성을 가짐

c. Gradient descent
    i. initialize W and update weights in the opposite of the gradient function
        - gradient 자체는 증가하는 방향이므로 - gradient 방향으로 업데이트를 해야 원하는 (감소하는) 방향으로 감
    ii. learning rate is a step size
        - important hyperparameter
    iii. other update rules using gradient
        - adam optimizer, gradient descent with momentum

d. Stochastic gradient descent (SGD)
    i. approximate sum using a minibatch of examples
        - b/c full sum can be expensive when N is large
    ii. common size of minibatch is 32, 64, and 128

Image Features

1. Feature representation
    a. Use this technique before deep learning
    b. Procedure
        i. take an image
        ii. extract feature representations
        iii. concatenate the feature vectors
        iv. feed to the linear classifier
    c. Methodologies
        i. color histogram
        ii. histogram of oriented gradients (HoG)
            - 이미지 내 각 지역마다 edge가 얼마나 존재하는지를 계산

        iii. bag or words
            - build code book and encode images
            - 단어가 존재하지 않으므로 이미지 내에서 랜덤하게 패치를 선택하고 패치들을 클러스터링하여 코드북을 생성함

2. Image Features vs ConvNets
    a. Image features
        i. given an image, extract features and obtain scores per classes
        ii. then train the model using feature extraction
    b. ConvNets
        i. given an image, directly learn features from the image and train the model