CS231n Lceture2, 3 정리

이름: 류창훈

-Lecture2:

-Image Classification-

SVM(서포트 벡터 머신)은 패턴인식, 자료분석을 위한 지도학습 모델이다. 출력층에 사용되는 함수로 소프트맥스(softmax)함수.

컴퓨터는 이미지를 숫자 조합으로 받는다. ex) [0, 255] -> 3채널(RGB)을 뜻함.

그리고 카메라로 어느 한 부분을 비추다가 다른 곳을 비추니 픽셀이 바뀐 것을 확인 가능.

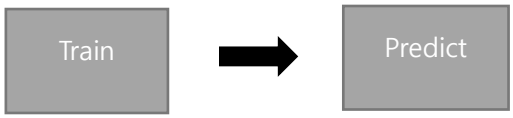
☞ 컴퓨터는 각각의 사물을 서로 다른 픽셀 값(숫자 조합)으로 인식.

데이터-추진(data-driven)접근 방법:

1. 이미지와 레이블(label)로된 데이터셋을 수집.
2. 기계학습(머신러닝)을 사용하여, 클래스분류기를 훈련.
3. 훈련한 것으로 새로운 이미지를 판별.

클래스 분류기:

1. 최근접 이웃(Nearest Neighbor):



-> 기계에서 알아서 train.

-> Predict단계에서 새로운 이미지 가져오기, 유사한 이미지 찾기, 가장 유사한 것 예측

학습(train)시에는 O(1), 예측(Predict)시에는 O(n)의 시간복잡도. 학습은 느리고 예측은 빠른게 좋아서 별로 좋지 못한 분류기이다.

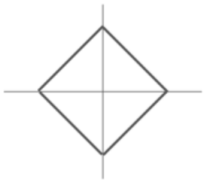
2. K-NearestNeighbors (K-최근접 이웃):

Nearest Neighbors의 취약점을 보완.

k = 1 -> k = 3 -> k = 5 순으로 갈수록 점점 분류 퀄리티가 좋아짐. (물론 때에 따라 다름)

L1 (Manhatan) distance:

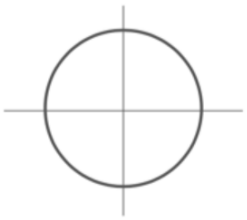
$(a, b) = \sum|a_i - b_i|$



-> 좌표 변경되면 거리 달라짐.

L2(유클리드) distance:

$(a, b) = \sqrt{\sum|a_i - b_i|}$



-> 좌표 변경해도 동일 (더 자연스러움)

L1, L2 둘 중에 뭐가 좋은지 나쁜지는 상황에 따라 다르다.

Setting Hyperparameters:

1st: (train세트 | test세트) -> (train세트 | 검증세트 | test세트) (better)

2nd: cross-validation(교차검증)

-> train세트, test세트 사이에 검증세트를 껴 넣는 것. (fold 사용). 하지만 딥러닝에서는 돈이 많이 들어서 사용을 잘 안함.

K최근접이 이미지에 사용하지 않는 이유:

1. 사진 여러 개 분류 후, L2거리 계산해본 결과. 거리 똑같이 나옴.

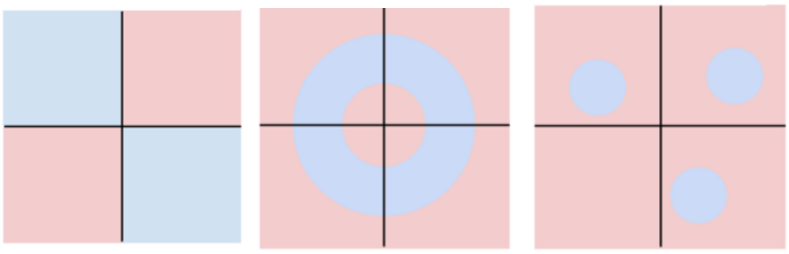
2. 이미지는 너무 많은 데이터로 구성. curse of dimensionality(차원의 저주).

Linear classification(선형 분류) :

사진 -> f(x,W(파라미터,가중치)) -> 분류

f(x,W) = Wx + b

Linear classification(선형 분류)가 어려운 경우:



1. 선을 그릴 방법이 없음(결정영역 decision Region)

2. 선형분류기는 픽셀 단위로 세서, 홀수 짝수 구분 어려움(동등성 parity)

3. 경계를 그릴 수 있는 방법이 모호(멀티모드 multi modes)

-Lecture3:

선형 분류 f(x,W) = Wx + b 할 때, W값은 정의를 하지 않음.

loss function(손실 함수) 정의해서 최적의 파라미터 값 W(최적화)를 해야된다.

전체 데이터를

$\{(x_i, y_i)\}_{i=1}^{N}$ 라고 한다면,

손실은

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

이 L을 최소화 해야 한다.

손실함수:

1. SVM - Hinge Loss

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

Sj는 잘못된 레이블의 스코어, Syi는 제대된 레이블 스코어.

0에서부터 뒤의 식 계산값에서 max로 최댓값을 취하겠다라는 뜻.

질문1:

자동차 점수가 조금 바뀐다면 손실에 어떤 일이?

-> 틀린 점수보다 1이상 더 큰지 그것만 보면 되기 때문에, 손실은 바뀌지 않는다.

질문2:

손실의 최소/최댓값은?

-> 힌지 손실 그래프를 봐보면, 음수로 간다면 손실이 무한대, 그러므로 최소 손실은 0, 최대는 무한대

질문3:

초기화시 W가 너무 작아서 모든 S = 0이라면, 손실은?

-> 버그, 코드를 확인해 봐야 한다.

질문4:

모든 클래스에 대한 합은?

-> 모두 1씩 증가해서 최종 Loss도 1씩 증가.

질문5:

합 대신 평균을 사용하면?

-> 당연히 똑같다.

질문6:

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)^2$$

로 바뀌면 어떻게 되는가:

-> 비선형적으로 바꿔서 다른 함수로 된다.

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

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

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

->위 식에 버그가 있는데, 2배를 해줬을 시에, 똑같이 손실 값 0이 나오므로, 그냥 같다.

정규화(regularization)

만약 어떤 데이터에 대해서 그래프를 그렸을 때, 그 그래프가 구불구불 하다면, 좋지 않음.

직선 그래프(단순한 그래프)를 예측하는게 더 좋다.

그래서 이를 위해서 정규화가 필요.

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

빨간 네모 제외한 것이 대표 손실 함수. 빨간 박스가 정규화 손실. 이 두개가 충돌하면서, 최적의 값을 찾는 것.(Data에 가장 최적화가 된 값을 찾는 것)

L2 regularization은 W의 값을 전부 고려하는 것을 좋아한다.

ex) [1,0,0,0]과 [0.25, 0.25, 0.25, 0.25] 중에서 두번째를 더 선호.

소프트맥스 분류(Softmax Classifier) = 다항 로지스틱 회귀:

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

-log함수를 최소화 하는게 목적.

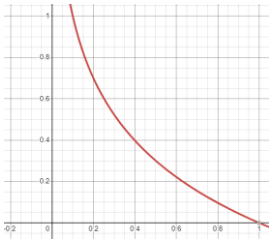
$$L_i = -\log\left(\frac{e^{sy_i}}{\sum_j e^{s_j}}\right)$$

이게 최종 식.

-> 모든 확률을 다 더하면 '1'.

질문1: 가능한 손실 L_i의 최대/최솟값은?

->



y축 값이 음수로 가면 안되기 때문에, 최대값은 무한대, 최솟값은 0

질문2: 초기화에서 W가 매우 작아 모든 S = 0이면, 손실은 얼마인가?

-> 처음부터 점수는 0으로 나올 것. 그러므로 e^0은 1. 그러므로 각각 점수는 1.

총 합은 3이므로, 확률은 -log(1/3)이 결론.

소프트맥스 vs SVM

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

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

SVM값은 뒤에 마진 값 1 때문에, 크게 변화가 없다.

반대로 소프트맥스 함수는 모든 레이블을 다 보기 때문에, 매우 민감하다 볼 수 있다.

최적화(Optimization):

최적화의 나쁜 예시:

1. 랜덤탐색(Random search)

2. 경사 따라가기(Follow the slope)

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

그냥 미분. 기울기 구하려면 맞다.

근데 근사치일 뿐이고, 평가시 매우 느리다.

-> 우리가 구하고자 하는 건 손실(loss)인데, 이건 W(가중치, 파라미터)의 함수일 뿐이다.

수치적 경사(Numerical gradient): 대략적, 느림, 작성하기 쉬움

분석적 경사(Analytic gradient): 정확, 빠름, 오류 발생 가능성 높음

-> 처음에는 분석적 경사를 쓰고, 계산 검토를 할 때는 수치적 경사를 사용한다(gradient check).

Mini-batch Gradient Descent:

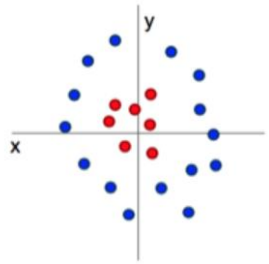
현실에서 실질적으로 항상 사용하는 기법. Full-batch Gradient Descent와는 달리, train세트 중에서 일부만 뽑아서 써서, 좀 더 효율적.

미니배치 사이즈는 환경에 맞춰서 설정. 보통은 (32, 64, 128)정도로 설정.

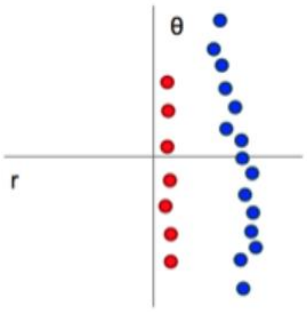
이렇게 하면 노이즈가 쫌 생기는데, 장기적으로 보면 Loss가 서서히 내려간다.

이미지 피쳐(Image Features):

원래의 방식-> 선형 분류기를 썼었는데, 바로 쓰지 않고, '픽셀 추출'(1st) 그리고 선형 분류기로 입력 후 이렇게 나온 '여러 계산결과를 하나로 합치고'(2nd) 그 다음 선형 분류기(Linear Classifier)(3rd)



이랬던 것을,



이렇게 바꿔서.

예시)

1. 색 히스토그램(Color Histogram)

->각각의 색(bin)을 빈도수를 측정해서, 그래프로 카운트.

2. Histogram of Oriented Gradients(HoG)

->각 픽셀을 나누고, 그 곳에 몇 개의 bin이 있는지(얼마전까지 사용되었다.)

3. Bag of words

-> 이미지의 작은 지점을 벡터로 기술, 이걸 다 모아서 사전화. 이 사전화 한 것 내에서 테스트할 이미지와 가장 유사한 것을 찾기(K-means 방식 사용). 이걸 Linear Classifier 적용.

원래 전통적인 방식은, Feature를 추출하고, 이걸 Linear Classifier 적용해서 결과값 얻는 방식.

딥러닝에서는 Feature를 따로 추출하지 않고, 이미지를 던져주면, 함수가 결과값을 알아서 출력하는 방식.