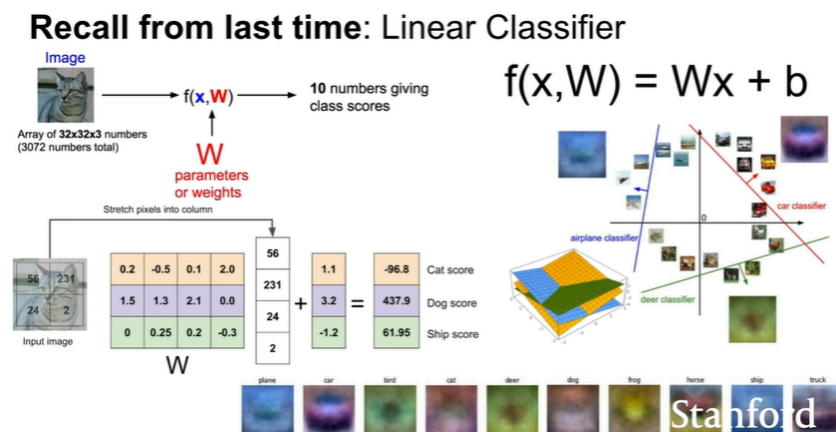


Week3_CS231n_3

Loss Functions and Optimization

지난 강의 복습

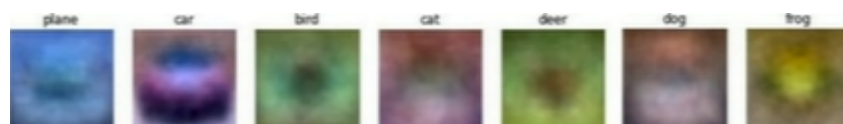


지난 강의에서 Neural network 의 첫 번째 구성 블록 중 하나인 Linear classifier 에 대해 배웠다.

Linear classifier는 parametric classifier라고도 하는데, training단계에서 training data의 정보가 parameter matrix인 W 로 summarize된다.

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

- 이미지를 넣으면 이미지의 pixel을 stretch 해서 하나의 column 벡터인 x 로 바꾼다.
- W 인 **parameter matrix**가 이 x 값을 각 class에 대한 점수인 column vector로 전환시킨다.
- 결과값에서 높은 score일 수록 해당 input이 해당 class에 속할 가능성이 높다는 것을 의미한다.



CIFAR-10

⇒ **Linear classification**은 class당 학습된 **template**에서 위와 같은 해석을 얻게 된다.

이미지의 모든 pixel과 10개의 classes마다 W 에 대응하는 entry가 존재하고, 해당 entry는 그 pixel이 해당 class에 얼마나 영향을 주는지를 나타낸다.

즉, W 의 각 행들이 각 class의 template을 의미하게 된다.

만약 이 행들은 재배열(unravel)해서 이미지의 픽셀에 대한 가중치에 대응시키면, 해당 행 값들을 image로 바꿔 class에 학습된 template를 위와 같이 시각화 할 수 있다.

Recall from last time: Linear Classifier



airplane	-3.45	-0.51	3.42
automobile	-8.87	6.04	4.64
bird	0.09	5.31	2.65
cat	2.9	-4.22	5.1
deer	4.48	-4.19	2.64
dog	8.02	3.58	5.55
frog	3.78	4.49	-4.34
horse	1.06	-4.37	-1.5
ship	-0.36	-2.09	-4.79
truck	-0.72	-2.93	6.14

TODO:

1. Define a **loss function** that quantifies our unhappiness with the scores across the training data.
2. Come up with a way of efficiently finding the parameters that minimize the loss function. (**optimization**)

W 값을 선택하는 방법, W 가 좋은지 안좋은지 판단하는 방법

1. Loss function : W 가 얼마나 안좋은지를 알려줌
2. Optimization : loss function을 최소화시켜 W 를 좋은 쪽으로 찾아가게 하는 것

Loss Function

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



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

A **loss function** tells how good our current classifier is

Given a dataset of examples

$$\{(x_i, y_i)\}_{i=1}^N$$

Where x_i is image and y_i is (integer) label

Loss over the dataset is a sum of loss over examples:

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

위 자료를 보면 cat, car는 제대로 분류가 되었지만 frog는 완전 잘못 분류된 것을 알 수 있다.

$$(x_i, y_i)_{i=1}^N$$

⇒ N : dataset의 data수

x : image의 pixel value

y : x에 대한 prediction값 ⇒ label, class갯수 범위의 정수값

$f(x, W)$ 로 구한 값과 y값을 loss function인 $L()$ 에 대입하여 loss를 구하고, 전체 dataset에 대한 loss의 평균을 L이라 한다.

Multiclass SVM loss (= Hinge loss)

SVM(support vector machine)

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

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

s_j : 정답이 아닌 클래스의 score

s_{y_i} : 정답 클래스의 스코어

1 : safety margin

⇒ 만약 s_j (정답이 아닌 클래스의 score)값에 safety margin인 1을 더한 값이 s_{y_i} (정답인 클래스)의 값보다 크면 이때의 $s_j + 1 - s_{y_i}$ 값이 loss가 된다.

그 반대일 경우 loss가 존재하지 않는다는 의미의 0이 된다.

⇒ 정답인 score가 정답이 아닌 score보다 1 이상 크면 loss가 존재하지 않는다.

cat	3.2	1.3	2.2	<div>the SVM loss has the form:</div> $L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + 1)$ <div>Loss over full dataset is average:</div> $L = \frac{1}{N} \sum_{i=1}^N L_i$ $L = (2.9 + 0 + 12.9)/3$ $= 5.27$
car	5.1	4.9	2.5	
frog	-1.7	2.0	-3.1	
Losses:	2.9	0	12.9	

Loss를 계산하면 현재의 classifier는 5.27만큼 성능이 나쁘다는 것을 알 수 있다.

최종 loss = 5.27

SVM loss에 대한 Q1 : What happens to loss if car scores change a bit?

⇒ loss의 값은 변하지 않는다. 즉 score가 몇 점인지에 대한 정확한 값보다는 margin값을 기준으로 정답 클래스가 다른 클래스보다 충분히 큰지가 중요

SVM loss에 대한 Q2 : What is the min/max possible loss?

⇒ Min = 0 , Max = infinite

hinge loss 그래프 개형 보면 판단할 수 있음

SVM loss에 대한 Q3 : =At initialization W is small so all s ~ 0, What is the loss?

⇒ # of classes - 1

loss가 제대로 나오는지 판단하기 위한 디버깅용으로 많이 사용.

SVM loss에 대한 Q4 : What if the sum was over all classes?(including j = y_i)

⇒ loss를 구할 때 정답 클래스의 값을 포함시킬 경우 이전 loss값 + 1이됨

⇒ loss의 min값이 1이됨

SVM loss에 대한 Q5 : What if we used mean instead of sum?

⇒ 별 차이없음.

SVM loss에 대한 Q6 : max연산에 square연산을 추가할 경우 loss의 변화

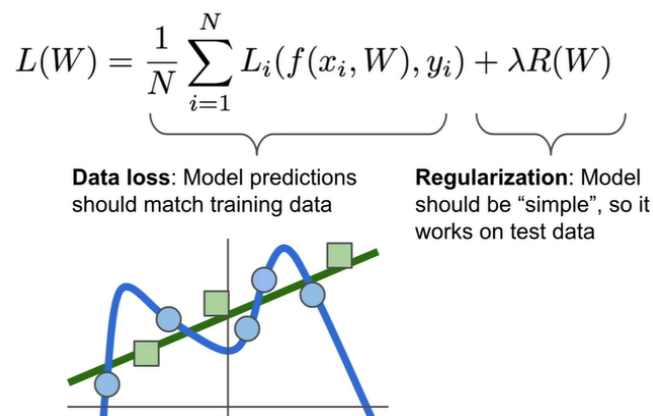
⇒ non linear이므로 값이 달라짐. 매우매우 안좋다 라는 값을 낼 수 있음.

Last Q : $L=0$ 인 W 를 찾았다고 했을 때, 이 W 는 유일한가?

⇒ No, $2W$ is also has $L = 0$

n 배를 할 경우에도 score값의 차이가 1 이상이 나올 경우 0으로 값이 동일하기 때문이다. 즉 W 는 여러개가 될 수 있다.

Overfitting



이전까지 training 단계를 통해 W 를 만들었지만 더 관심을 가져야 할 곳은 test 단계이다. 그러나 W 는 train dataset에 맞춰서 만들어졌기에 W 는 test data set에 맞지 않는다.

위 그림에서 train dataset에 맞춰서 prediction 을 그렸지만 초록색인 test값과는 맞지 않는 것을 알 수 있다. ⇒ **overfitting**

Regularization


Regularization을 통해 모델을 simple하게 하여 testset에 맞는 model을 찾아준다.

- L2 regularization : Squared norm of W
- L1 regularization : L1 norm
- Elastic net(L1 + L2)

-

Softmax Classifier(Multinomial Logistic Regression)

Softmax Classifier (Multinomial Logistic Regression)



cat	3.2
car	5.1
frog	-1.7

scores = unnormalized log probabilities of the classes.

$P(Y = k|X = x_i) = \frac{e^{s_k}}{\sum_j e^{s_j}}$

where $s = f(x_i; W)$

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

확률값을 이용해서 Loss를 구한다.

모든 score에 exponential을 취한 값을 기준으로 확률을 구한다.

이후 구한 값에 -log()를 취한다. (평균구한값이 1에 가까울수록 loss가 0이됨)

Q1. What is the min/max x possible loss L_i ?

⇒ Min = 0, Max = infinite

Q2. Usually at initialization W is small so all s ~ 0. What is the loss?

⇒ $-\log(1/\text{\#of classes})$: 디버깅에 사용

Softmax vs SVM

- SVM은 margin을 통해 max값 계산
- Softmax는 확률계산
- SVM은 datapoint가 약간 변해도 둔감
- Softmax는 확률 계산이기 때문에 datapoint가 약간 변해도 바로 확률에 영향

Optimization

⇒ Optimization은 W값으로 setting된 거대한 valley를 내려가는 것과 같다.

Loss값이 0인 부분을 찾아 가야하는데 loss function, regularization, f()모두 복잡한 연산

- Random search ⇒ 가장 나쁜 방법
- Follow the slope(Gradient descent)

⇒ analytic gradient 이용 (numerical gradient 는 디버깅툴로 사용)

- Stochastic Gradient Descent(SGD)

⇒ N의 값이 클수록 Gradient descent의 연산이 매우 느려짐.

minibatch를 이용해 데이터를 잘라서 사용하는 SGD

Aside : Image Features

CNN연구 이전의 Image의 특징을 이용하는 방법

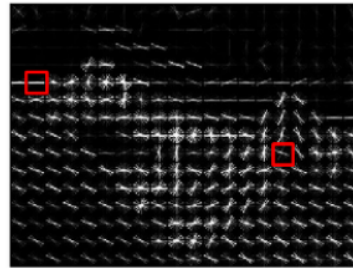
- Motivation
 - Linear function으로 특징을 구분하기 어려운 상황을 극좌표계로 변환 시 linear하게 변환 가능해지는 상황에서 motivation을 받음

특징 추출법

- Color Histogram : image에서 어떤 color hue값이 많이 나오는지를 count해서 feature 추출
- HoG(Histogram of Oriented Gradients)
: 이미지를 8x8 pixel로 자르고 해당 값에 어떤 각도가 많은지를 히스토그램으로 추출



Divide image into 8x8 pixel regions
Within each region quantize edge
direction into 9 bins



Example: 320x240 image gets divided
into 40x30 bins; in each bin there are
9 numbers so feature vector has $30 \times 40 \times 9 = 10,800$ numbers

- Bag of Words

: 이미지를 잘라낸 후 클러스터링(각도, 색깔)

새로운 이미지가 들어오면 똑같이 클러스터링해서 특징 비교

자연어처리에서 사용하는 방법

