Section 7

- Learning Rate: Gradient Decent 에서 Learning Rate(알파) 를 설정, Decent의 Step의 크기

적절하지 못한 Learning Rate는 다음의 문제를 일으킨다.

+ Overshooting: 학습이 이뤄지지 않을 정도로 learning rate가 높음

+ Small learning rate: 학습이 너무 오래걸림

위의 각 상황은 Cost function Step별 값을 관찰하며 세부 조정한다.

- Data Preprocessing

데이터 값에 큰 차이가 있을 경우 Normalize를 해야함

(데이터의 차가 커서 Overshooting됨)

1) Zero-centered data: 변수의 평균값이 [0]이 되도록한다.

2) Normalized data: 데이터를 Scaling하여 소수의 변수가 매우 큰 경우 이를 분산 기준으로 Scaling함으로써 예외 케이스로 인한 Traing Error를 방지한다.

Ex> standardization

result = 각 차원에 대해, (x - 평균) / 분산

- Overfitting

: 테스트 데이터와 너무 딱 맞아서(Specific) 실물 데이터(General)와는 차이가 나는 경우

해결책

1) Training data가 많으면 해결될 수 있음

2) Regularization을 사용:

Loss = Cost 함수 + 람다(=regularization 상수)*시그마(w^2)

ex) tf.reduce_sum(tf.square(W))

=> 위와 같은 식을 통해 학습 시에 Cost를 전체적인 평균치에 가깝게

기존 Overfitting 된 데이터보다

더 크게 조정함으로써

General Solution에 가깝게 한다.

- Performance evaluation

Training Set으로 평가하면 제대로 된 학습 평가를 진행할 수 없다.

학습 평가를 위해서 다음으로 데이터를 구분하여 시스템을 설계한다.

1) Training Set: 데이터의 70% 가량을 할당하여 학습 데이터로 활요한다.

(이 중, 적은 데이터를 학습 시에 자체 평가를 하기 위한 Validation Set으로 사용한다.)

2) Test Set: 학습 이후 평가를 위한 데이터로 사용한다.

- Online learning

기존 학습된 있는 데이터에 추가로 계속해서 학습

* MNIST 는 이미지 프로세싱에 쓰이는 손글씨 데이터

Tensorflow의 세션 저장 기능으로 학습 데이터를 저장가능

:http://stackoverflow.com/questions/33759623/tensorflow-how-to-restore-a-previously-saved-model-python

- Accuracy

95~99%면 굉장히 정확

(90대 이상이면 일반 ML에서는 굉장히 정확한 편, Deep Neural Net기준으로는 정확하지 않음)

-

Section 8: Deep Learning

- Activation Function

Activation function (Sig(w*x) + b)을 통해 1보다 크면 값을 전달 / 아니면 전달하지 않음

이러한 Logistic Regression Units이 모여 Multi Layered Networks를 구성

- XOR 문제의 해결

Perceptron: MLP(Multi Layered Perceptron)를 쓰면 XOR 해결가능하지만 학습이 불가능하다고 함 (bias)

- Backpropagation: 뒤에서 앞으로 전달하며 문제를 해결 -> Perceptron 저술의 이슈를 해결

Layer가 많아지면 오히려 정확도가 떨어짐

- Convolutional Neural Network

이미지 인식에 쓰이는 방식