**머신러닝 4개월 실무 경험자의 강의 다시 듣기**

(현재 모위딥1 : 21/51, 모위딥2 : ~lab6)

**Pytorch 기반**

**추가적으로 공부할 내용**

* **optimizer의 종류 알아보고 특성 공부해보기, optimizer별 pytorch document 보고 사용법 및 parameter들 공부하기**
* **손실함수 종류 알아보고 손실함수 별 특징 정리하기! 손실함수에 대해 수학적 접근 하고 정리해보기**
* **L1, L2 regularization에 대해 더 찾아보기**
* **Maximum likelihood 등 가능도에 대해 찾아보기**

Lec1 ( 기본적인 machine learning의 용어와 개념 설명 )

**Unsupervised learning**: ex) google news grouping/ word clustering

**Supervised learning**: labled된 training dataset으로 학습

* Regression : ex) predicting final exam score
* Binary classification : ex) pass/non-pass
* Multi-label classification : ex) grade(A,B,C,D and F)

Lec2 ( linear regression의 hypothesis와 cost 설명 )

Linear regression은 분류라기 보다는 숫자를 예측하는 느낌

H(X) = Wx + b

Cost function :

제곱을 통해서 차이를 양수화 하고, 차이가 작을 때 보다 클 때 penalty를 많이 주게 됨

Cost(W,b) = (m : data의 개수)

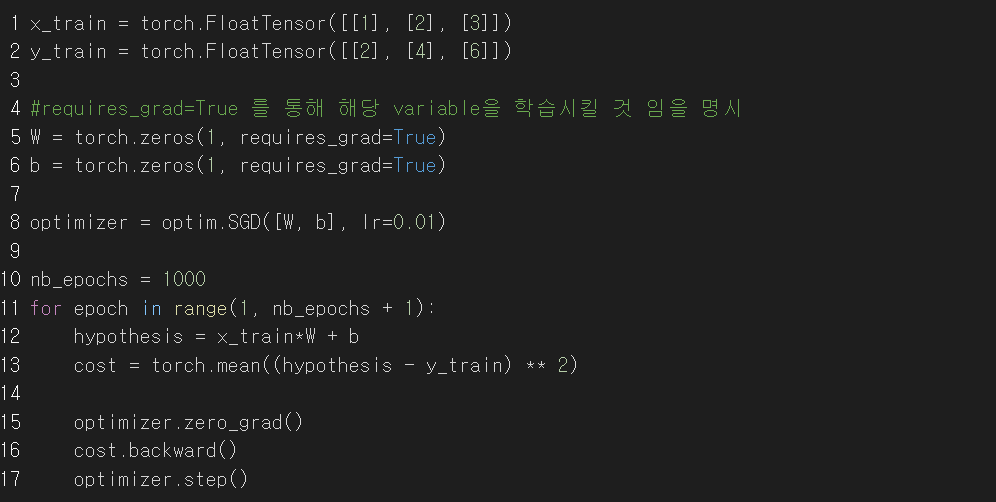
* cost값을 최소화하는 W와 b를 구하자
* MSE(Mean Sqared Error) 평균제곱오차

Lec3 ( linear regression의 cost 최소화 알고리즘의 원리 설명 )

Gradient descent algorithm(경사하강법) <- optimizer

* minimize cost function

1. start with initial guesses
2. each time change the parameters(w, b), select the gradient which reduces cost the most possible
3. repeat & do until converge to a local minimum



Optim.SGD란(stochastic gradient descent)

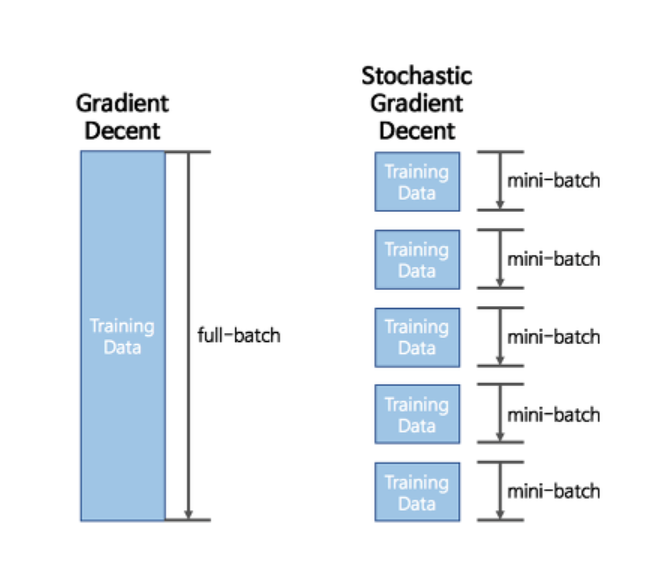
**-> optimizer의 종류 알아보고 특성 공부해보기, optimizer별 pytorch document 보고 사용법 및 parameter들 공부하기**

- Mini-batch 단위로 gradient descent를 함

-확률적(stochastic)은 무작위를 의미 함

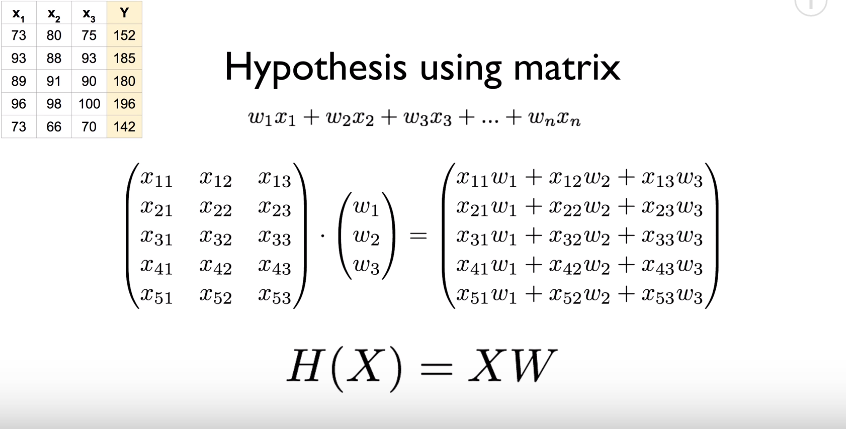
-배치가 너무 커지면 중복 될 가능성도 커서 noise가 심할 수 있음

-뒤죽 박죽이지만 그냥 gradient descent보다 빠르게 찾아나간다



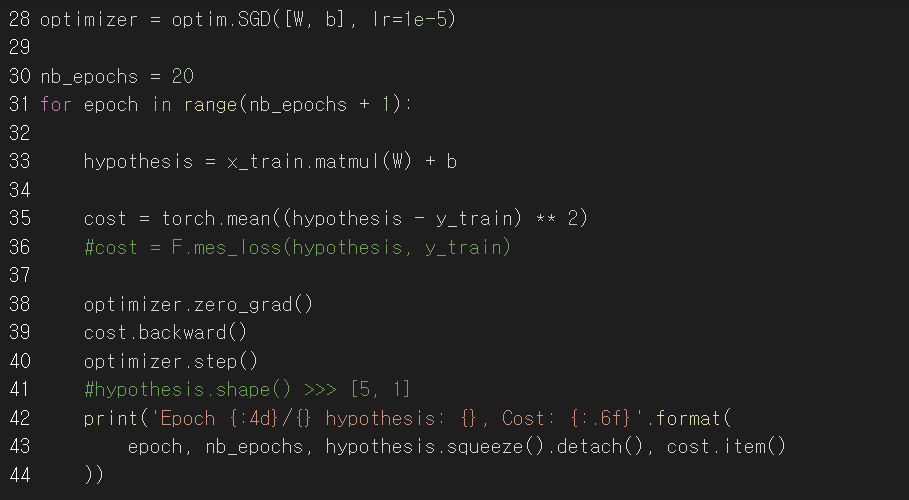
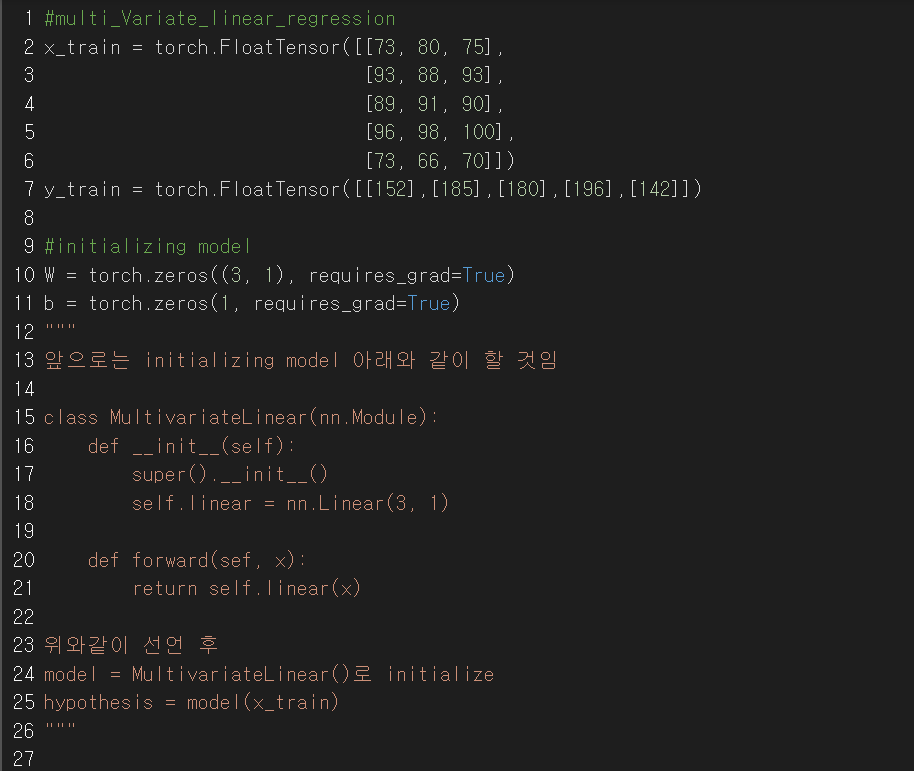
Lec4 ( multi-variable linear regression )

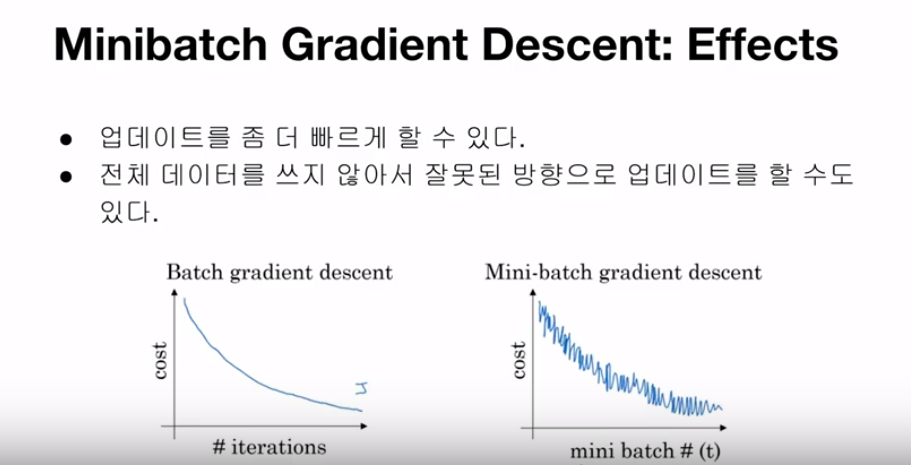
좀 더 복잡한 상황을 위해 matrix multiplication을 활용



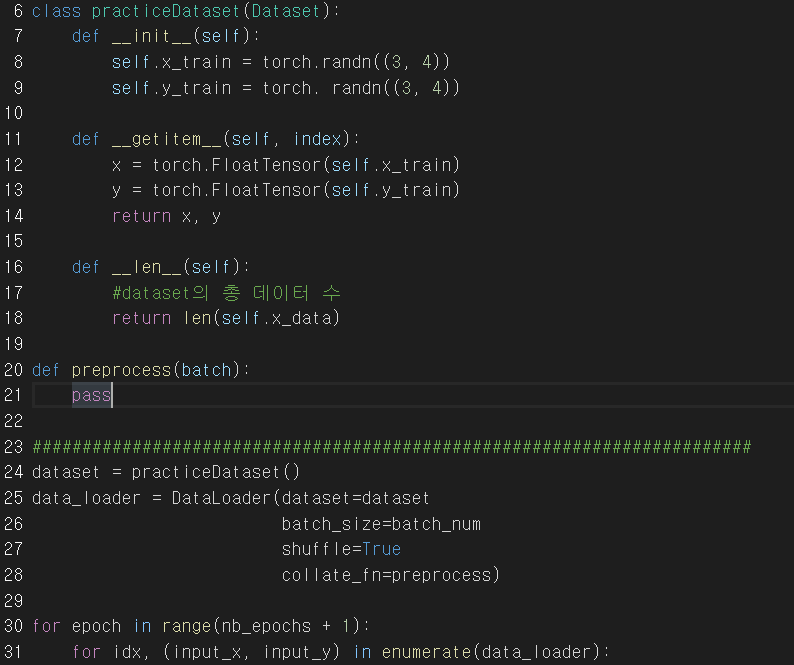
(n, m) (m, k) -> (n, k)

* (n, m) : m features(variables) / n samples(instance) for each feature
* theory에서는 H = Wx와 같이 나타내지만 Matrix 에서는 XW 곱의 순서를 지켜야한다.





How to dataLoad? (basic data loading)



Lec5 ( Logistic Classification, Logistic Regression의 cost 함수 설명 )

* **logistic regression**
* **binary cross entropy**

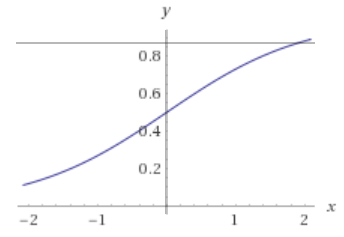
Binary Classification -> spam or ham email/ facebook show or hide

생각해보면 linear regression으로 될 듯하다는 생각

But! 안됨 - hypothesis값이 0 or 1이 나와야 하는데 h(x) = Wx + b의 경우 hypothesis값이 0보다 작거나 1보다 커질 수 있다.

**Logistic Hypothesis**

Sigmoid 함수

( 0 < hypothesis < 1 )

Cost 함수에 대한 이야기

Linear regression의 경우 MeanSquaredError를 쓰게 되는 경우

이런 부드러운 곡선이 생기기 때문에

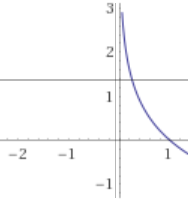
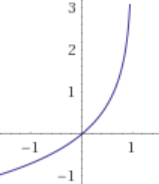
최솟값을 찾아 나가기 쉽다

하지만 logistic함수의 경우 MSE를 쓰게 되는 경우

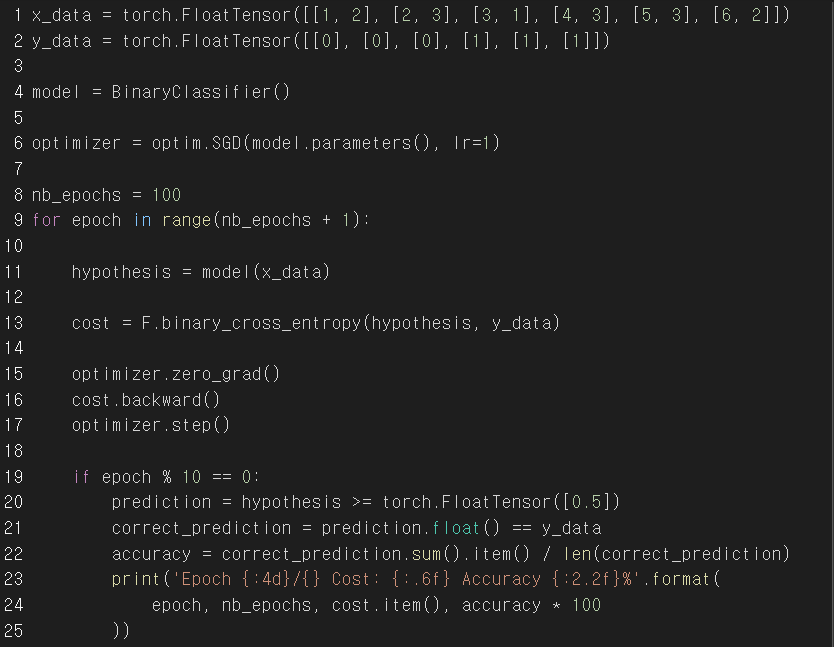
이러한 이상한 모양이 생기기 때문에

Global minimum을 찾지 못하고 local minimum을 찾는 경우가 생긴다.

Logistic에 알맞은 cost함수

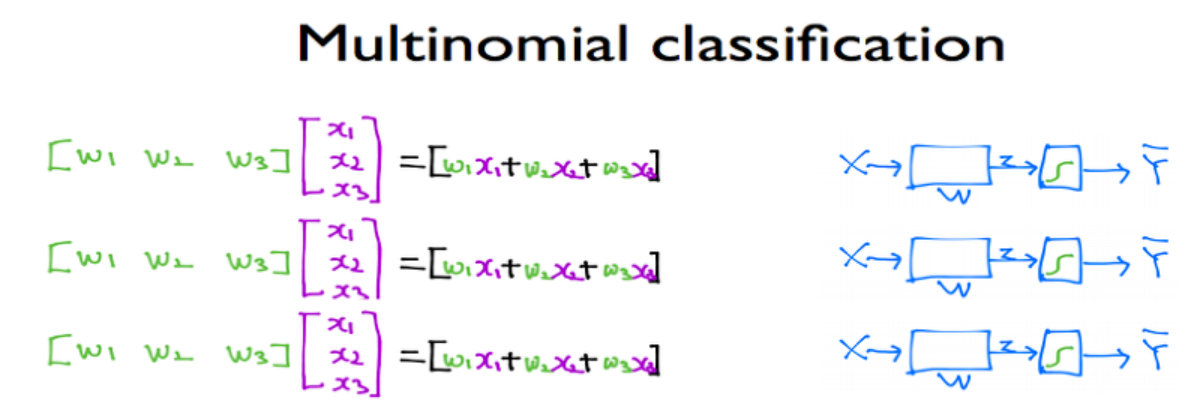
If condition 없이 하나의 식으로 표현하면



Lec6 ( softmax regression )

* **softmax**
* **cross entropy**

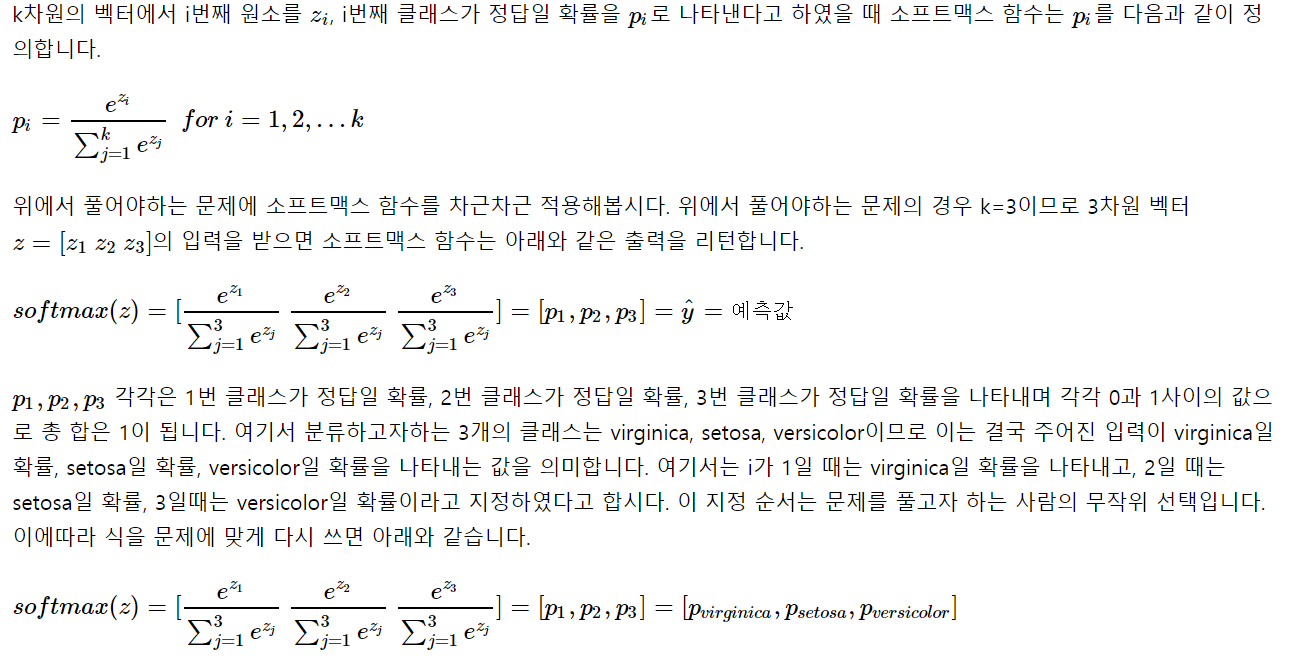
softmax는 데이터를 2개 이상의 그룹으로 나누기 위해 binary classification을 확장한 모델이다.



weight들은 각각 A, B, C 그룹을 나누기 위한 각 그룹별 가중치

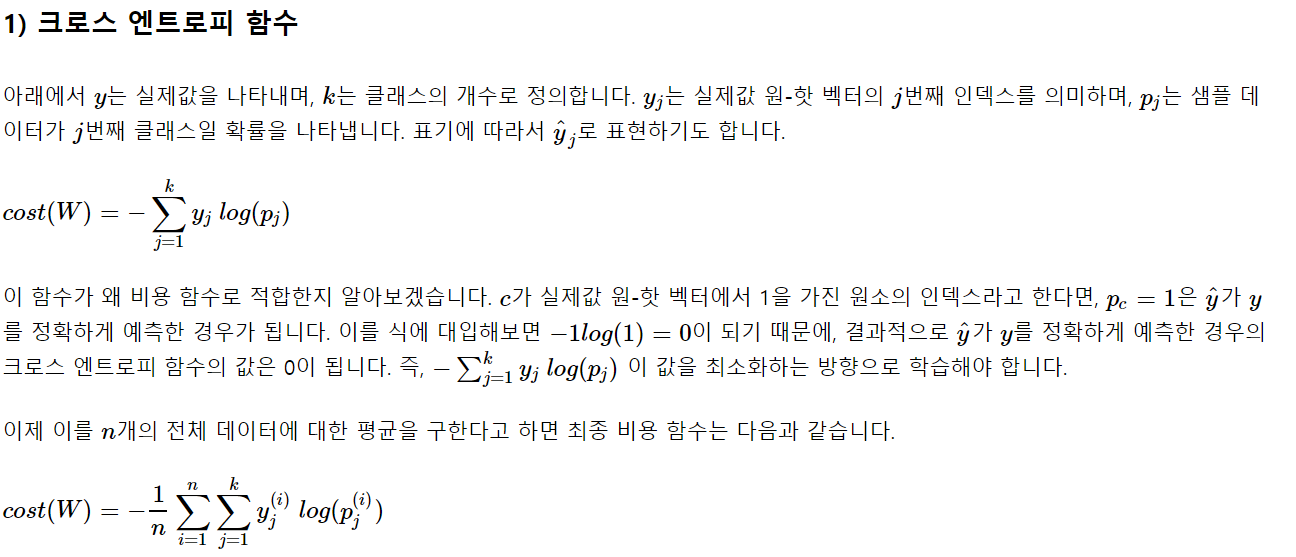
logistic regression을 사용하기 전 값에 대해서 logistic을 적용하여 0~1 사이 값으로 만들어 줘야한다. 이를 위해 softmax가 사용된다.

이 중에서 가장 큰 값이 채택되므로 argmax를 하는 것과 전체적 과정이 비슷하다고 보면 된다.



<<https://wikidocs.net/35476>>

Cross entropy



<<https://wikidocs.net/35476>>

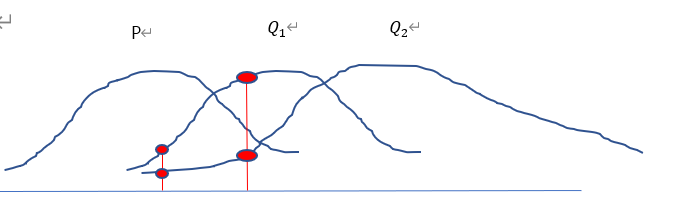
Entropy란 불확실성을 의미한다. 고로 cross entropy값이 높다는 것은 불확실성이 높다는 것을 의미하는 것 이며, 이 값을 줄여나가야 한다.

값은 실제 관찰 값으로 one-hot encoding 되어있기 때문에 1이나 0의 값이다.

* **손실함수 종류 알아보고 손실함수 별 특징 정리하기! 손실함수에 대해 수학적 접근 하고 정리해보기**

**Softmax**

**Cross Entropy**



H(P, Q1) < H(P, Q2)

Cross entropy 값이 작아지도록 해야 함 그래야 P와 Q가 가까워진다

**Lec7 (Learning rate, data preprocessing, overfitting 그리고 일반화regularization)**

**Learning rate**

Gradient descent를 할 때 중요한 것이 learning rate(보통 처음 0.01)

cost값이 줄어들지 않고 발산하게 된다면 learning rate를 너무 크게 잡은 것

cost 값이 줄어드는 데 takes too long or stops at local minimum 하다면 learning rate 너무 작게 잡은 것

**Data preprocessing**

Data분포가 좋지 않다면 한쪽으로 왜곡된 등고선이 나타날 수 있음

이렇게 되면 learning rate를 잘 잡더라도 cost 값이 바깥으로 빠져나갈 수 있기 때문에

Zero-centered data(0중심 분포)로 바꾸거나 normalized data(특정 범위안의 분포)로 바꿔줘야 한다.

Feature에 대해 어떤 feature는 큰 값을, 어떤 feature는 작은 값을 갖고 있다면 큰 값에 조금 더 집중하여 학습 할 수 있기 때문에 normalized 해주는게 좋다.

* Normalization

X\_std[ : , 0] = (x[ : , 0] – x[ : , 0].mean()) / x[ : , 0].std()

**Overfitting**

* Our mdel is very good with training data set
* Not good at test dataset or in real use

Solutions

* More training data
* Reduce the number of features(중복된 것 줄이고 features 줄이기 특징이 너무 많으면 overfitting 날 우려가 있다.)
* Regularization

**Regularization**

**-weight decay**

일반적인 방법: cost함수 뒤에 더해줌(L2 regularization으로 weight에 대한 패널티 부여)

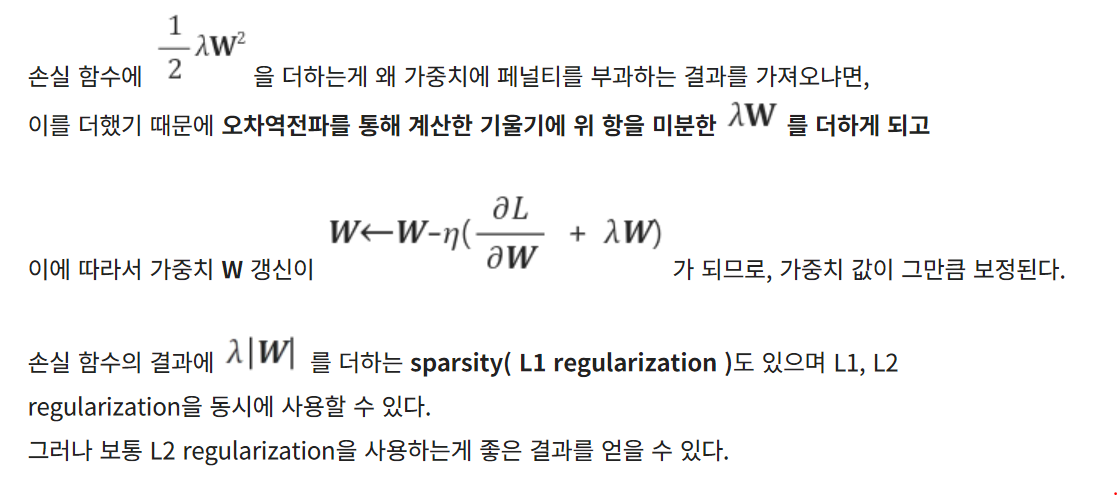
람다 값이 0 이면 regularization 쓰지 않겠다, 1이면 regularization을 중요시 여기겠다는 의미

cost값을 단순히 줄이는 방식으로 학습을 하게 되면 특정 가중치들이 커지면서 오히려 결과가 나쁘게 되는 경우가 있다

이를 해결하는 것이 L2 regularization이며 특정 가중치가 비이상적으로 커지는 것을 막는 weight decay 효과를 볼 수 있게 된다

Local noise인 outier들의 영향을 최소하는 하는 것

* **L1, L2 regularization에 대해 더 찾아보기**



Reference

<https://umbum.dev/222>

**- Early stopping**

Validation loss가 더 이상 낮아지지 않을 때 멈춰준다.

**-reduce network**

**-dropout**

**-batch normalization**

**이들을 적용한 basic approach to train dnn**

1. Make a neural network architecture
2. Train and check that model is over-fitted

* If it is not, increase the model size(deeper and wider)
* If it is, add regularization, such as drop-out, batch-normalization

1. Repeat from step 2

**Training/ Testing dataset**

Original set을 training set과 testing set으로 나눈다

종종 **Training set**을 training set과 validation set(development set)으로 나눠서 training 하는 도중에 학습이 잘 되고 있는지 판단 할 수 있도록 한다.(이 과정을 통해 overfitting을 막을 수 있다)

빨간색이 validation loss 파란색이 train loss

초록색 지점부터 간격이 넓어지므로 overfit이 나는 중

따라서 초록색 지점 epoch에서의 parameter를 선택하면 overfitting을 방지할 수 있다.

**Maximum Likelihood Estimation(MLE)**

Likelihood(가능도)가 최대가 되는 값을 찾는 것 = observation을 가장 잘 확률 분포 함수의 parameter 값을 찾아나가는 과정

숙명적으로 overfitting이 따르게 된다.

**Lec**