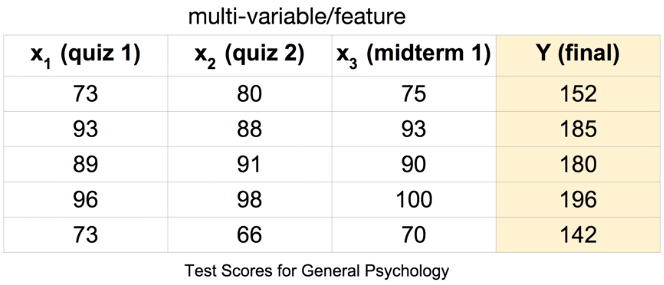
**ML\_Lec 04.**

**Multi-Variable Linear Regression.**

****

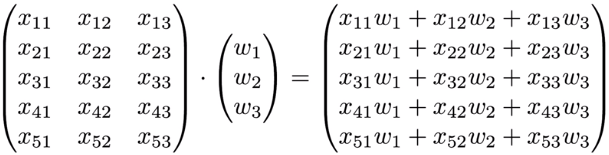
**Cost Function. //손실함수**

**Matrix. //행렬**

**Matrix Multiplication.**

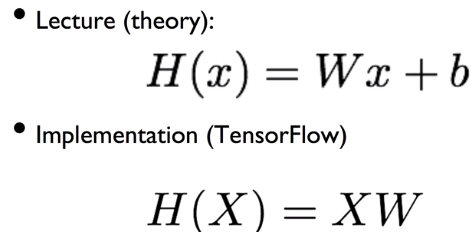
1. ****

Instance

1. ****

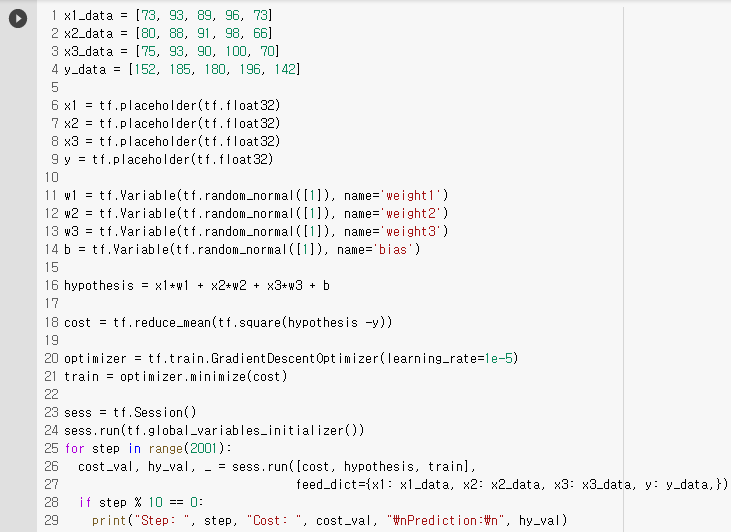
♠ [n, ~~3~~] [~~3~~, m] [n, m]

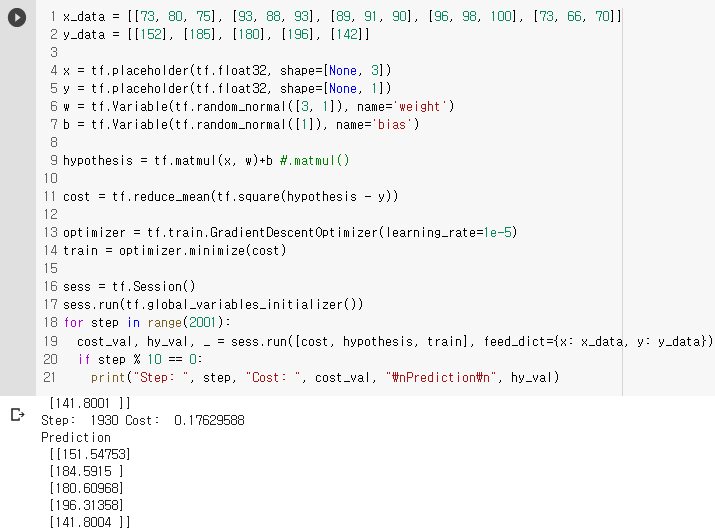
**Wx & XW.**

****

**ML\_Lab 04-1.**

**Test.**





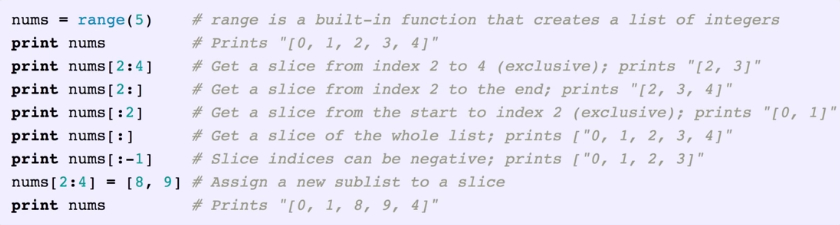
**ML\_Lab 04-2.**

**Loading Data form File.**

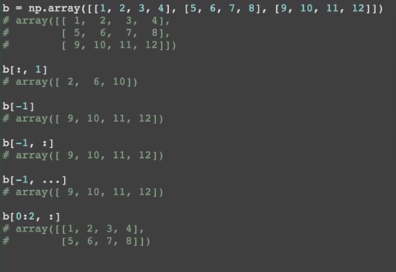
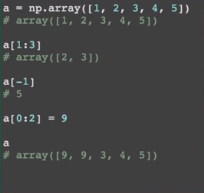




**Slicing.**

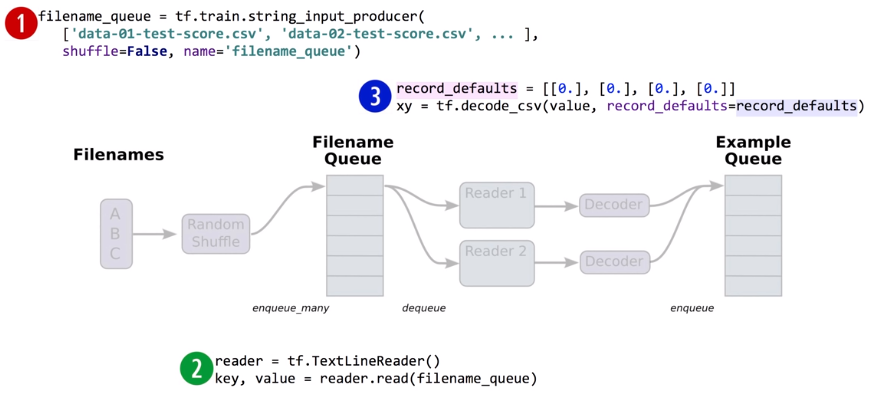


**Indexing, Slicing, Iterating.**

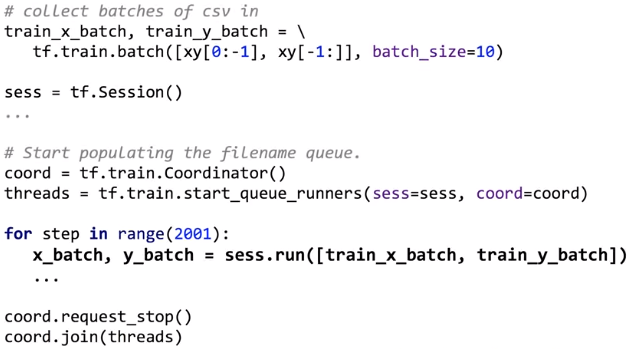
** **

1. Arrays can be indexed, sliced, iterated much like lists and other sequence types in Python
2. As with Python lists, slicing in NumPy can be accomplished with the colon (**:**) syntax
3. Colon instances (**:**) can be replaced with dots (**…**)

**Queue Runners.**



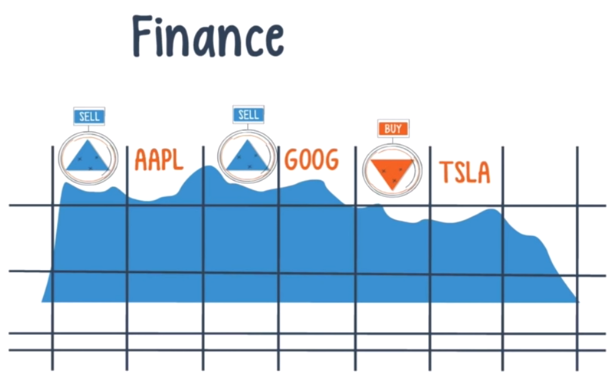
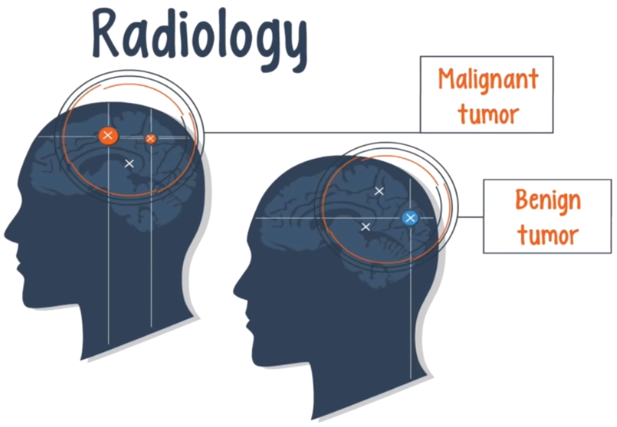
1. 여러 개의 파일을 읽을 때, Filename Queue에 쌓는다. (+Random Shuffle)
2. Reader로 연결한다.
3. Decoder를 거쳐 Example Queue에 쌓는다.
4. 일정 배치만큼 읽어와 학습시킨다.





**ML\_Lec 05-1.**

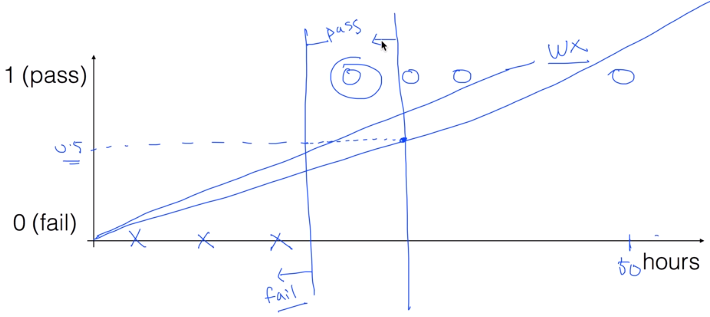
**Binary Classification.**

♠ 0,1 Encoding

1. Spam Detection: Spam(1) or Ham(0)
2. Facebook Feed: Show(1) or Hide(0)
3. Credit Card Fraudulent Transaction: Legitimate(0) or Fraud(1)

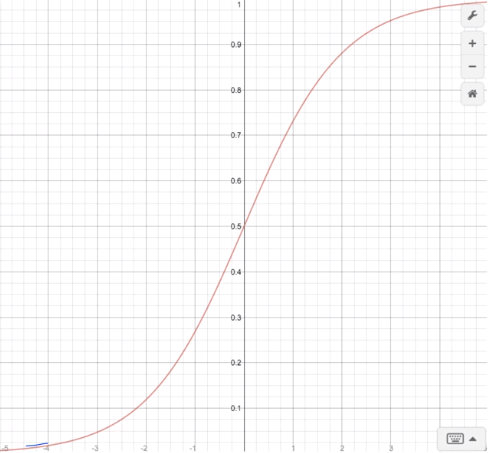
**Pass(1)/Fail(0) Based on Study Hours. (If Linear Regression.)**

****

♠ 문제점

1. 수치가 커지면(50hours) 선이 기울어진다.
2. 선이 기울어지면 기준값(0.5)의 위치가 바뀐다.
3. 기준값의 위치가 바뀌면 기존 결과의 판단이 달라진다.
4. 1보다 큰 값, 0보다 작은 값이 나올 수 있다.

**Logistic Regression. //로지스틱 회귀**

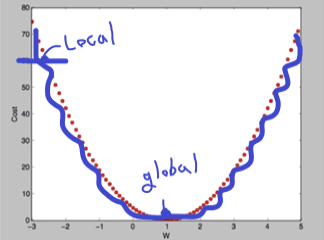
****

♠Logistic Function,Sigmoid Function

Sigmoid: Curved in two directions, like the letter “S”, or the Greek ς(Sigma) //들여쓰기 예외

**ML\_Lec 05-2.**

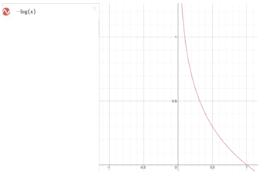
**Cost Function.**



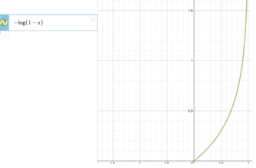
1. 문제점

Local Cost에서 끝나 Global Cost로 갈 수 없다.

// 때문에 사용

cost

H(x), W

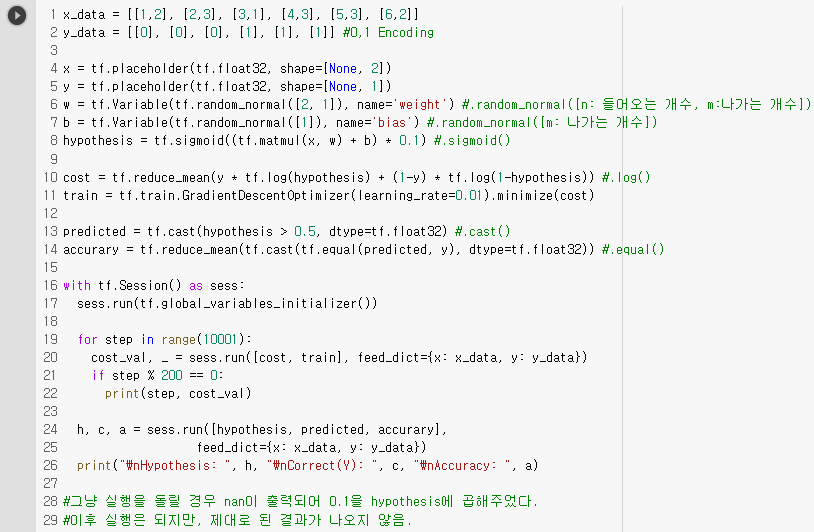
cost

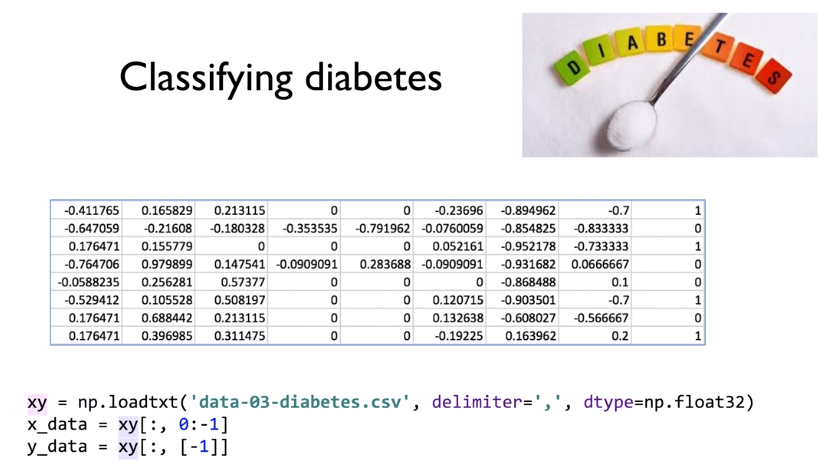
H(x), W

**Gradient Descent Algorithm.**

**ML\_Lab 05.**

**Test.**

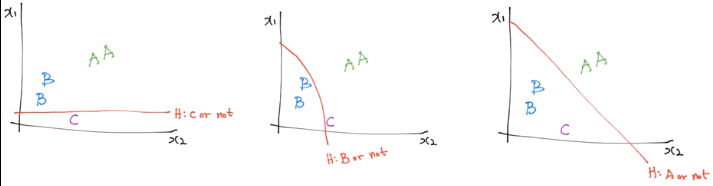




**ML\_Lec 06-1.**

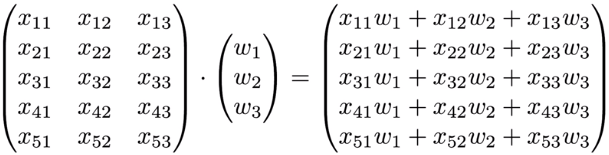
**Multinomial Classification.**

Hyper Plane으로 각각의 경우를 구분하고, Matrix Multiplication을 이용한다. //n차원-1의 선형함수

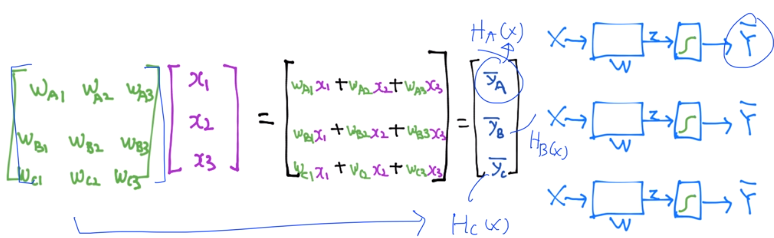


1. X → 「C」 → Y
2. X → 「B」 → Y
3. X → 「A」 → Y

++++++++++++++++++++

****

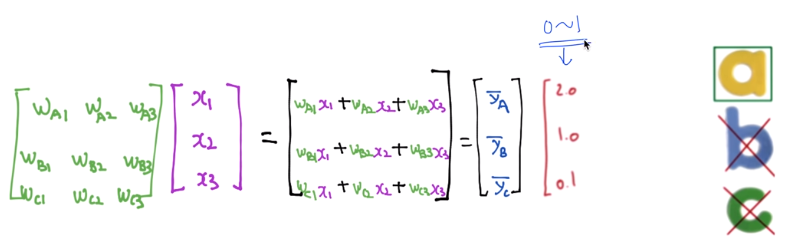
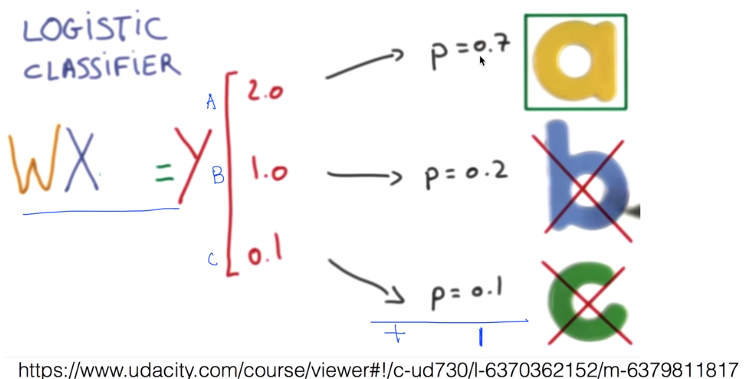
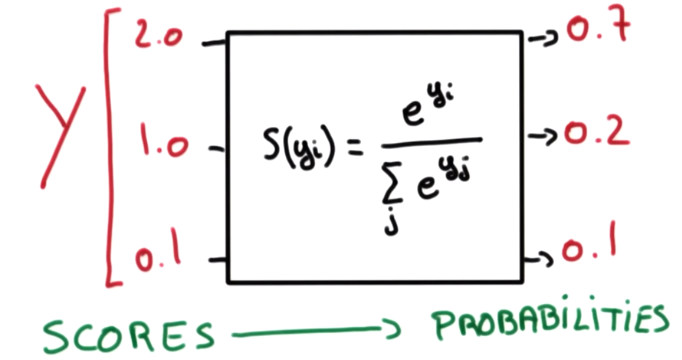
∥∥∥∥∥∥∥∥∥∥∥∥∥∥∥∥∥∥∥∥



**ML\_Lec 06-2.**

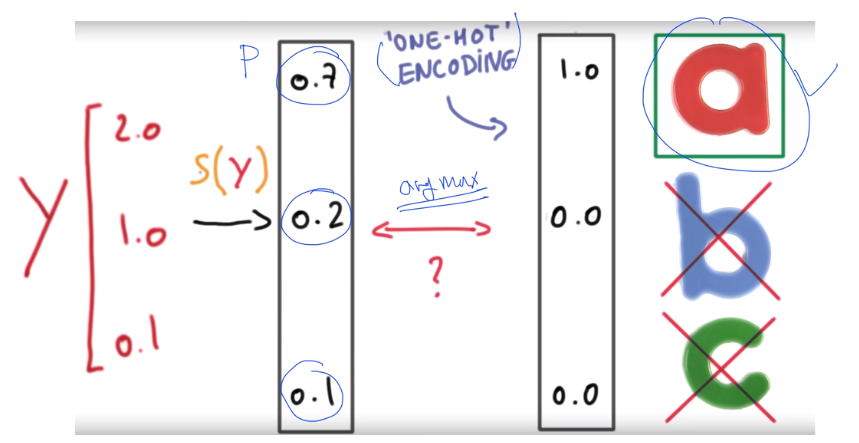
**Softmax Function.**

Softmax Function을 이용하여 값을 확률과 같이 바꾸고, One Hot Encoding을 이용한다.

1. 
2. 
3. 

Logits

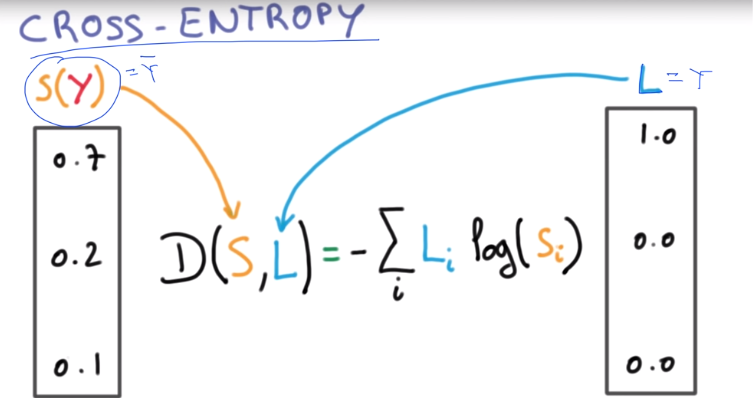
Softmax

1. 0~1의 값
2. 확률(p)
3. ****

♠ One Hot Encoding

제일 큰 값(확률)을 1로 하고 나머지는 0으로 한다. //들여쓰기 예외

**Cross-Entropy Cost Function.**

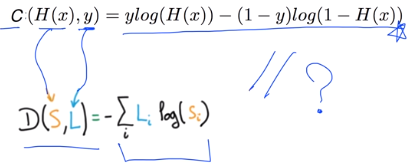


= //: 0~1

1. L = = B (Real)
2. S = = B (T),
3. S = = A (F),

**Logistic Cost & Cross Entropy.**

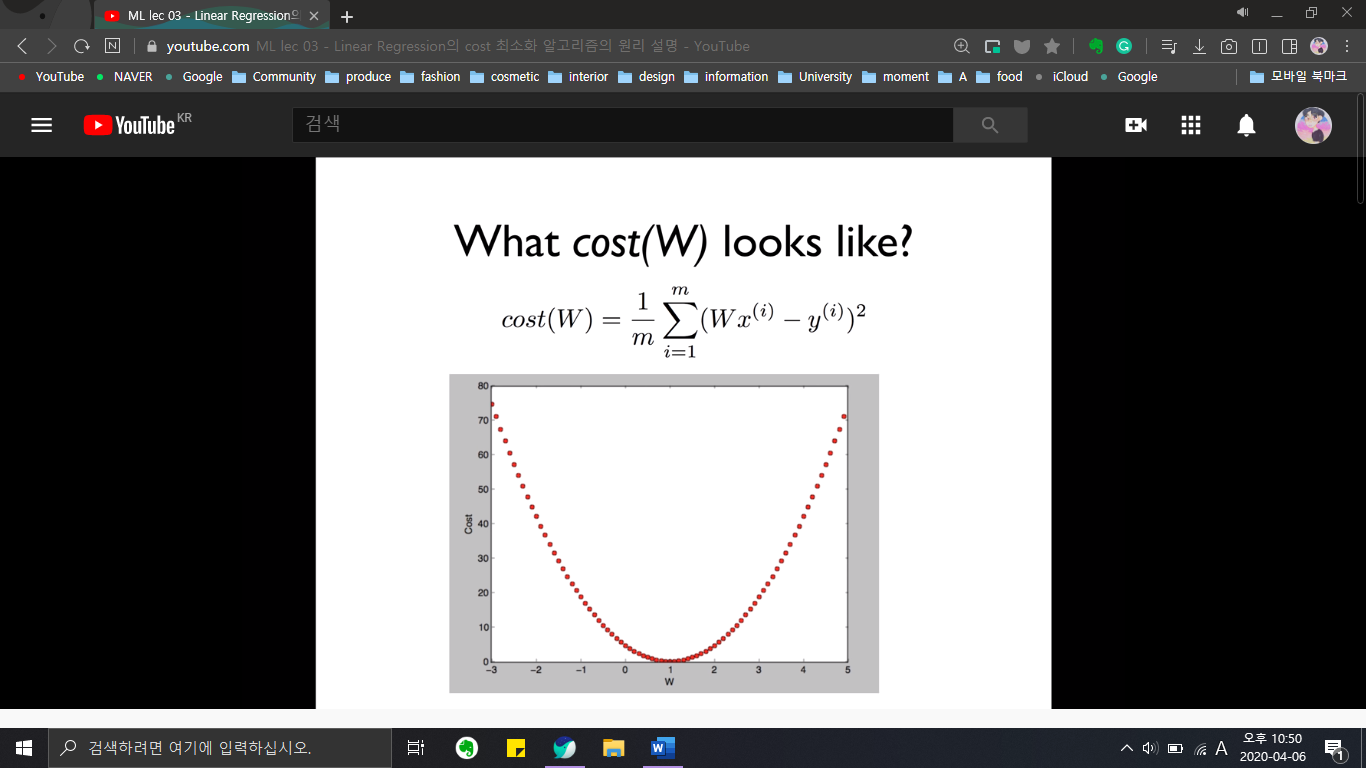
-



**Cost Function.**

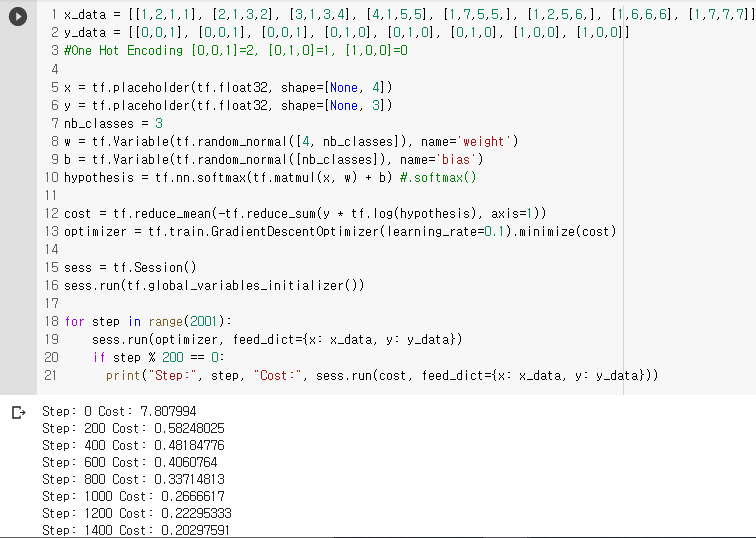
**Gradient Descent Algorithm.**

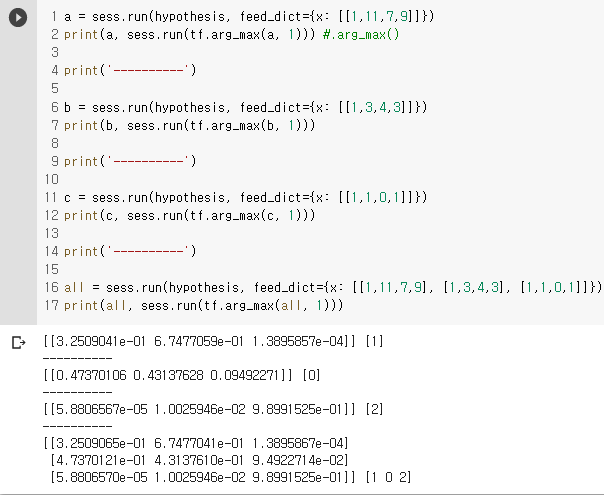
//자세한 미분은 다루지 않음.



**ML\_Lab 06-1.**

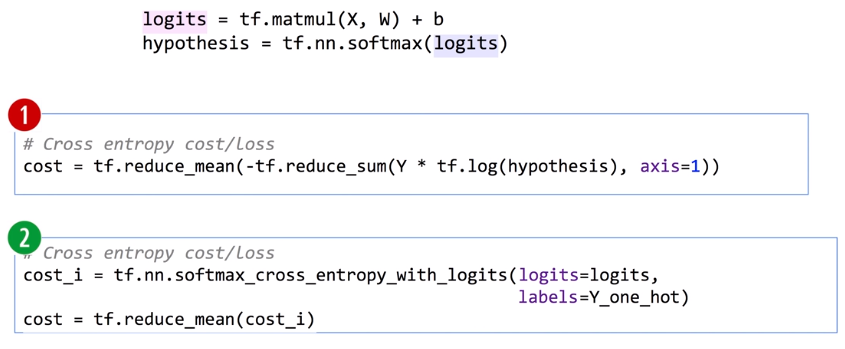
**Test.**





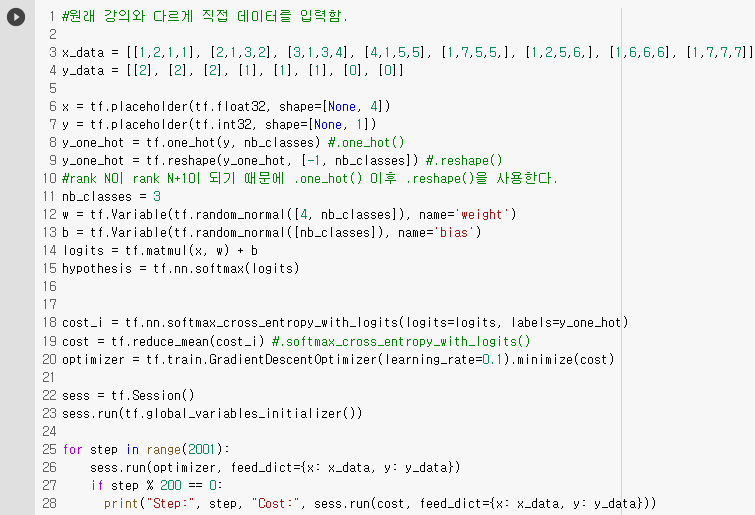
**ML\_Lab 06-2.**

**Softmax\_cross\_entropy\_with\_logits.**

****

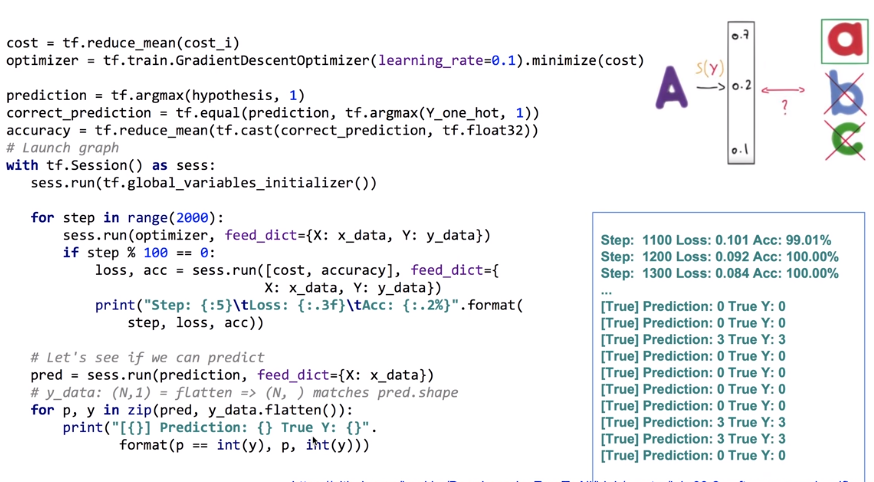
1. ML\_Lab 06-1.에서의 코스트
2. .softmax\_cross\_entropy\_with\_logits()을 사용한 코스트

**Test.**



**Animal Classification.**

****

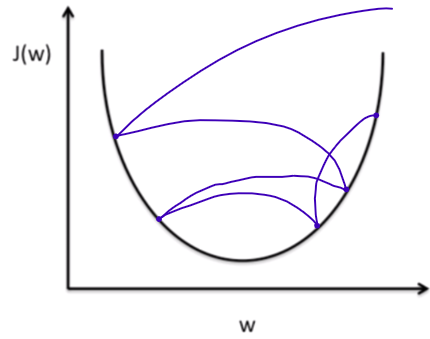
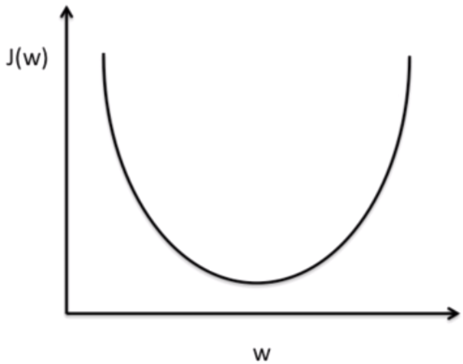
****

**ML\_Lec 07-1.**

About Machine Learning Tips.

**Learning Rate. //학습률**

1. Large Learning Rate: Overshooting.



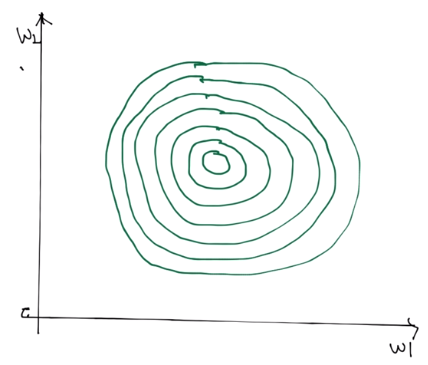
1. Small Learning Rate: Takes too long & Stops at local minimum. //강조 예외
2. Try several Learning Rates
3. Observe the Cost Function.
4. Check it goes down in a reasonable rate.

**Data(X) Preprocessing for Gradient Descent. //데이터 전처리.**

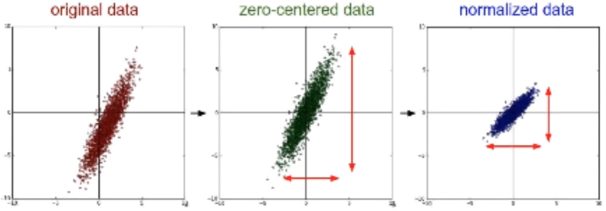
1. 값의 차이가 심하게 날 경우



1. 정상적인 경우



1. Data(X) Preprocessing

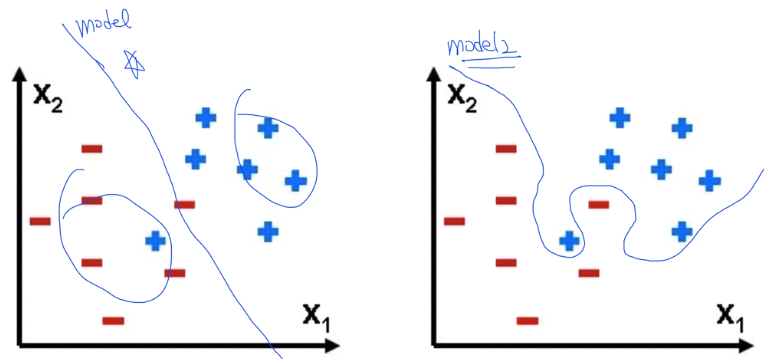


♠ Standardization //표준화

//들여쓰기 예외

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

**Overfitting. //과적합**

****

1. Our model is very good with Training Data Set. (with memorization)
2. Not good at test dataset or in real use.
3. Solutions for Overfitting
4. More Training Data
5. Reduce the number of features
6. Regularization

**Regularization. //일반화, 정규화**

Let’s not have too big numbers in the weight.

1. No Regularization
2. High Regularization
3. Little Regularization
4. Code: 12reg = 0.001 \* tf.reduce\_sum(tf.square(W))

**ML\_Lec 07-2.**

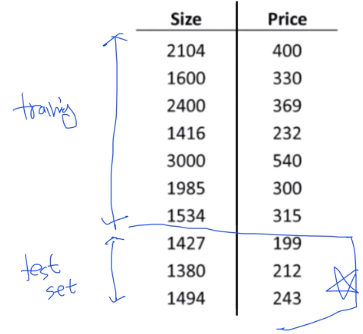
**Evaluation using Training Set.**

모든 데이터를 학습시키는 것은 좋은 방법이 아니다.



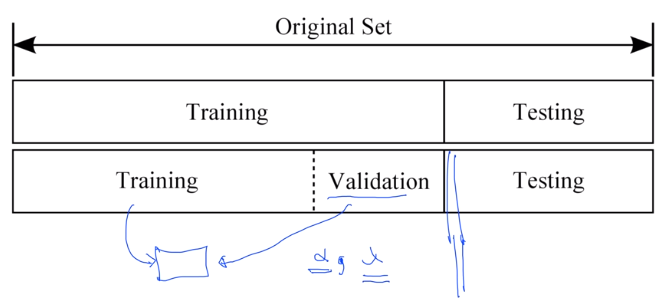
1. 100% Correct (Accuracy)
2. Can memorize

**Training Sets & Test Sets.**



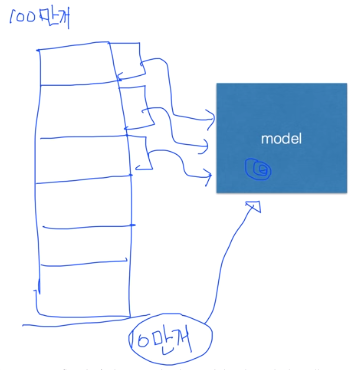
**Training, Validation and Test Sets.**

(Learning Rate) & (Regularization Strength)의 값을 조정한다.

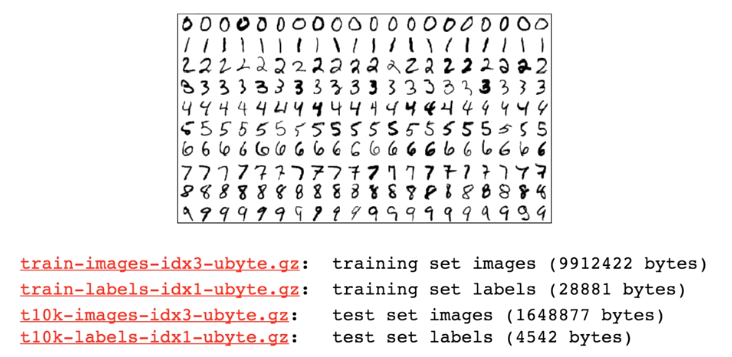


**Online Learning.**

이미 100만개가 입력되어 있을 때, 추가로 10만개의 데이터만 추가하면 된다.



**MNIST Dataset.**



**Memo**

.constant() //상수

.Session() //tensor에 데이터를 넣어 흐르게 함.

.run() //실행

.add() //더하기

.placeholeder(), feed\_dict={a:a\_data} //변수, 값을 나중에 할당.

.Variable() //변수, 자동으로 업데이트.

.random\_normal(Shapes) //랜덤 값 반환

.reduce\_mean() //평균

.square() //제곱

.GradientDescentOptimizer() //미니 배치 확률적 경사하강법(SGD) 구현.

.minimize() //최소화

.global\_variables\_initializer() //.Variable()를 초기화.

.append() //append

.plot() //plot

.show() //show

.reduce\_sum() //총합

.assign() //.Variable()의 값 변경.

.compute\_gradients() //compute\_gradients

.apply\_gradients() //apply\_gradients

.matmul() //matmul

.loadtext() //text불러오기.

.set\_random\_seed() //랜덤 값 시드, 다른 환경에서도 같다.

.string\_input\_producer() //Queue, text를 Filename Queue에 쌓기.

.TextLineReader() //Queue, text를 Reader로 연결.

.read() //Queue, text읽기.

.decode\_csv() //Queue, text decode

.batch() //Queue, text batch

.Coordinator() //Queue, Coordinator생성.

.start\_queue\_runners() //Queue, Queue를 Thread로 시작.

.request\_stop() //Queue, 중지

.join() //Queue, 대기

.sigmoid() //S자 곡선

.log() //로그

.cast() //새로운 자료형

.equal() //값이 같은지

.softmax() //softmax

.arg\_max() //arg\_max

.one\_hot() //one\_hot

.reshape() //reshape

.softmax\_cross\_entropy\_with\_logits() //softmax\_cross\_entropy\_with\_logits

.format() //format

.flatten() //flatten

//Lab으로 연결 https://colab.research.google.com/drive/1gaTpEufmhoK2CsEsNyfDDtyynQ\_HRpSu

//14폰트, 12폰트, 10폰트

//1. 1) a. \*♠

//0.71 1.34