ML 개념

ML이란

단일 layer 분류

NN – CNN,RNN

우연서

머신러닝

이해해야 하는 것
- ML과 기존 소프트웨어와의 차이점?
  소프트웨어 설계는 입력을 기반으로 반응하는 것 (explicit programming)
  ML은 명확하게 룰이 없는 경우에 대응하기 위한 방법

-ML의 분류 -
supervised / unsupervised

supervised란
Training set에 label이 있는 것을 의미 →

예) 이미지 분류, 시험점수 예상, 이메일 스팸 필터

Unsupervised
반대로 라벨이 없는것.(라벨을 줄 수 없는 것)
- 구글 뉴스 분류
- 유사한 단어 클러스터링

supervised learning – regression model

Supervised에 대해 조금 더 알아보면

Feature data : X1 x2 x3 x4 의 X 데이터들로 아이템의 특징값들

해당 피쳐들이 나타내는 label : y

라벨값들이 조금씩 다를 수 있음 ➔ all predicting final exam score based on time spent

이제 아래와 같은 애들을 순서대로 알아볼 것이다.

X(hour)	Y(score)
10	90
9	80
3	50
2	30
7	??

regression

X(hour)	Y(p/np)
10	P
9	P
3	F
2	F
7	??

Binary classification

X(hour)	Y(grade)
10	A
9	B
3	D
2	F
7	??

Multi-label classification

선형회귀 (Linear Regression)

목표!
- **가설 - h(x)**
- **Cost함수**
- **Minimize**

앞의 성적 예측 표들을 수행하기 위한 방법

리니어 리그레션의 목표

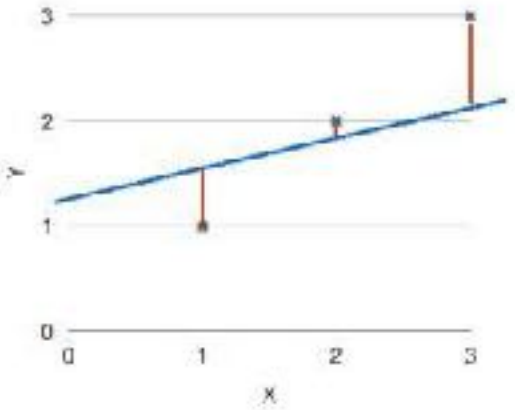
- **데이터들에 맞는 선을 결정해서 최적의 선(그래프)를 찾아내는 것**

X,y값으로 그리면 이렇게 가설을 그릴 수 있음

리니어 리그레션의 방법

step1. 가설을 세운다 **H(x) = Wx+b** → 일차방정식으로 직선
* 가설이란 데이터들이 이선이랑 쭉 비슷하게 있을 것이라는 것

step2. 그래프와 데이터들의 거리가 가장 가까운 그래프를 찾는다.
(이 때 그것을 구하는 방법이 cost 함수와 minimize!!)



$$cost(W,b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

1~m개의 데이터를 가설 그래프와의 결과 값 차이를 제곱해서 모두 더한 다음 m으로 나누는 것. → **따라서 cost 함수가 적으면 데이터에 가장 가까운 그래프라고 할 수 있는 것**

실습

순서

1) 데이터 준비
x_train = [1, 2, 3] y_train = [1, 2, 3]

2) 가설 설정에 필요한 변수 준비 - variable
W = tf.Variable(tf.random_normal([1]), name='weight')
B = tf.Variable(tf.random_normal([1]), name='bias')

3) 가설 설정
Hypothesis = x_train * W + b

4) 최적의 그래프 찾는 식
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)

5) 실행
#Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

6) 학습
```
for step in range(2001):
    sess.run(train)
    if step % 20 == 0:
        print(step, sess.run(cost), sess.run(W), sess.run(b))
```

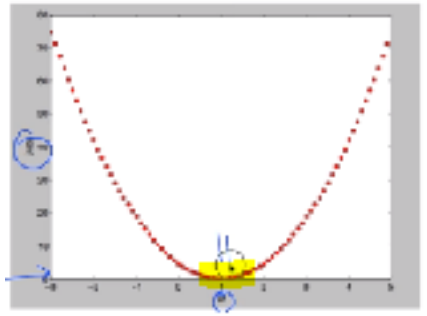
- **Gradient descent 와 minimize란**

: cost 값을 최소화 하는 걸 minimize 라고 하는데 이를 수행하기 위해서 gradient descent algorithm을 사용

최소값을 찾는 방법은 cost 함수의 가장 작은 값을 찾는 것 -> cost 함수의 minimum point를 찾는다. ->해당 미니멈 포인트는 기울기가 가장 작은 부분으로 찾는다 → 미분으로 찾는다(미분이 기울기니까)

How to minimize cost?

$$cost(W) = \frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})^2$$

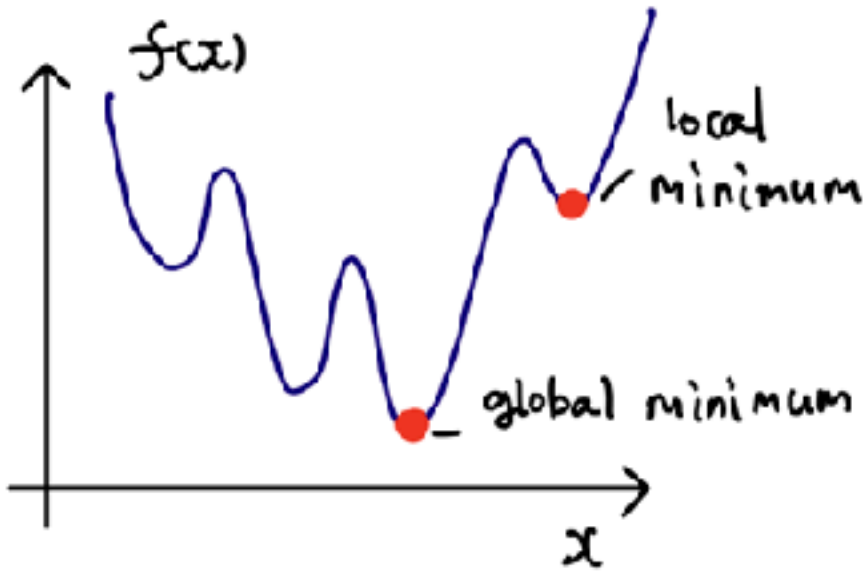


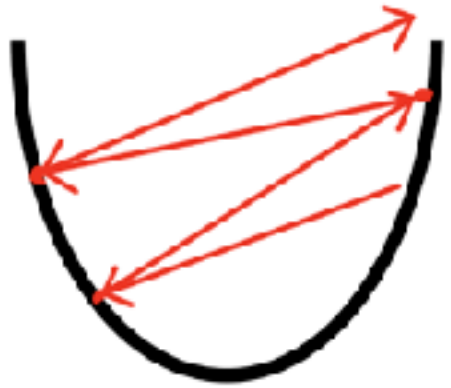
일단 wx로 가설을 간략히 한 다음에 cost 함수에 넣음

$$W := W - \alpha\frac{\partial}{\partial W}cost(W)$$

W값에 +- lr*(미분분한 기울기)
기울기가 0이면 해당 W로 픽스됨

추가 용어
- Convex model :위 그래프처럼 cost 함수가 엎어진 바가지 모양인 것들
- Local minima :국소 지역성으로 cost 함수의 진정한 최소값으로 가기 전에 기울기가 0 인 포인트를 만나서 진짜 cost의 최소값을 찾지 못하는 현상





large learning rate

small learning rate

Multi-variable linear regression

값이 여러 개인 경우에는 어떻게 할까

X1(test 1)	X2(test 2)	X3(test 3)	Y(final)
10	10	10	90
9	9	9	80
3	3	3	50
2	2	2	30

$$H(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + b$$

$$cost(W, b) = \frac{1}{m} \sum_{I=1}^{m} (H(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) - y^{(i)})^2$$

연산만 늘어날 뿐 방법은 동일
데이터 처리 방법만 조금 달라짐

None(제한없음)

```
x_data = [[73., 80., 75.],   y_data = [[152.],
          [93., 88., 93.],            [185.],
          [89., 91., 90.],            [180.],
          [96., 98., 100.],           [196.],
          [73., 66., 70.]]            [142.]]
```

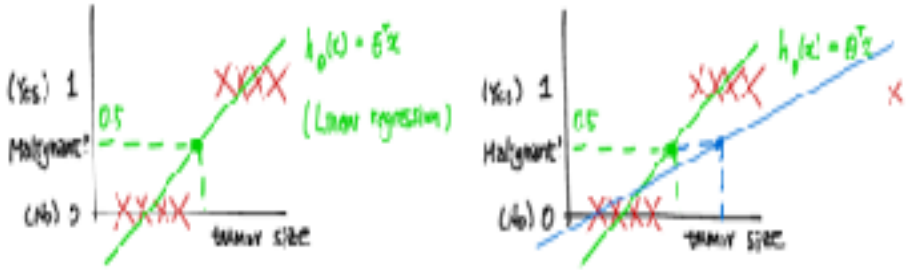
```
# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([3, 1]),
name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis - 행렬곱으로 동일하게 처리
hypothesis = tf.matmul(X, W) + b
```

Logistic regression Classification

(논리회귀)

목표

Binary Classification : 둘 중 하나의 카테고리에 속하게 하는것
like spam 필터, 또는 주식시장에서 팔까 말까, 암진단.



리니어 리그레션으로도 가능한데 가장 근접한 선을 그은 다음 y 값을 기준으로 통과 미통과를 나눔

하지만 x 값이 겁나 큰 게 들어와서 학습을 하다보면

그럼 우리가 세워놨던 0.5로 보았을 때 값을 비정상적으로 나눌 수 있음

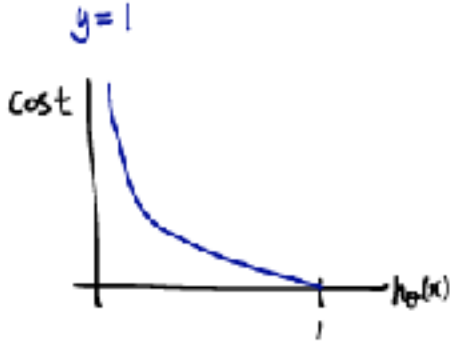
이 때 왜 0.5는 바꿀 수 없는가?
만약 이 그래프가 종양을 구분하는 것이라고 할때 0.5보다 크면 종양인데 값 구분을 위해 0.5보다 작다고 해줄 수는 없음

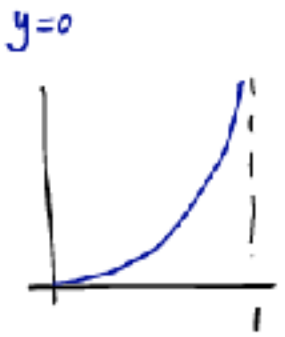
Sigmoid를 Hypothesis function에 사용하자

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

⇒ 문제점
*Sigmoid*함수를 *hypothesis function*으로 사용시 *cost function*에 값을 넣었을 때 *convex* 모델이 성립하지 않음

Log함수를 Cost function으로 사용하자(뒷장)

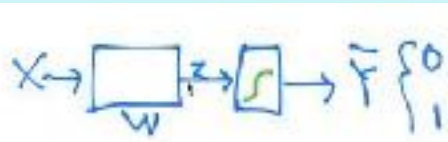




결론

$$\text{cost}\left(h_\theta(x), y\right) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Logistic regression Classification – cost function

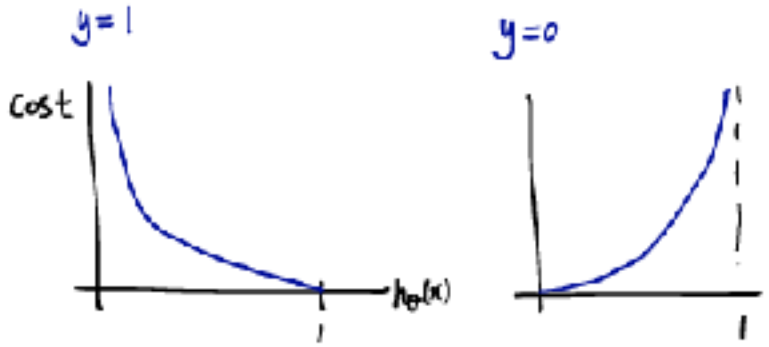


- 가설 - h(x)
- Cost함수
- Minimize

Log 함수를 cost function으로 사용하는 이유

1. 자연상수로 이루어진 가설함수의 제곱을 convex - model로 만들기 위해
   자연상수의 역이 로그함수이기 때문에 로그함수를 사용하면 컨벡스 모델로 변환 이가능

2. binary classification 에 문제없이 작동함



	라벨값이 1	라벨값이 0
예측값이 1	정답 o cost(1) = 0	정답 x cost(1) = 1
예측값이 0	정답 x cost(0) = 1	정답 o cost(0) = 0

$$C(H(x), y) = \begin{cases} -log(H(x)) & : y = 1 \\ -log(1 - H(x)) & : y = 0 \end{cases}$$

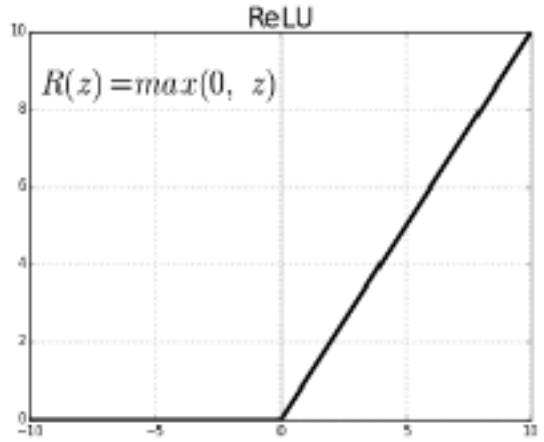
$$C(H(x), y) = ylog(H(x)) - (1 - y)log(1 - H(x))$$

Binary Classification에서 Cost 함수의 의미

실제의 값과 예측한 값이 같거나 비슷하면 → 0

실제 값과 예측한 값이 같거나 비슷하지 않으면 →1

```
# cost/loss function
cost = -tf.reduce_mean(Y * tf.log(hypothesis) + (1 - Y) *tf.log(1 - hypothesis))
```

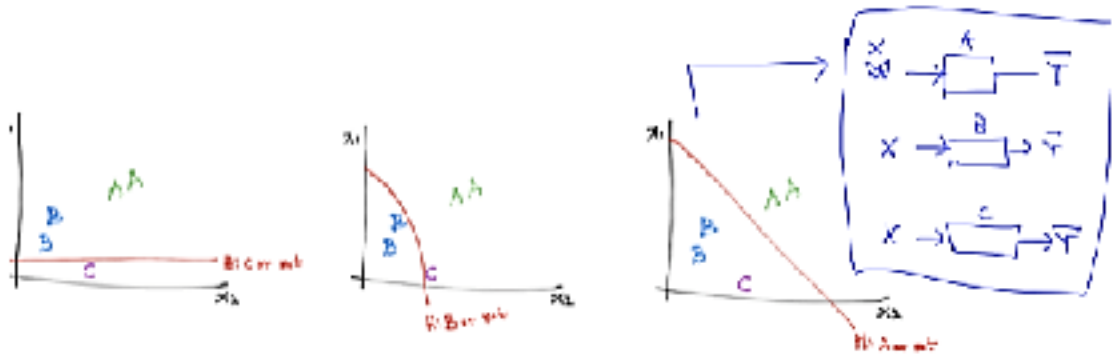

ReLU
$R(z) = max(0,\ z)$
10
8
6
4
2
0
−10
−5
0
5
10

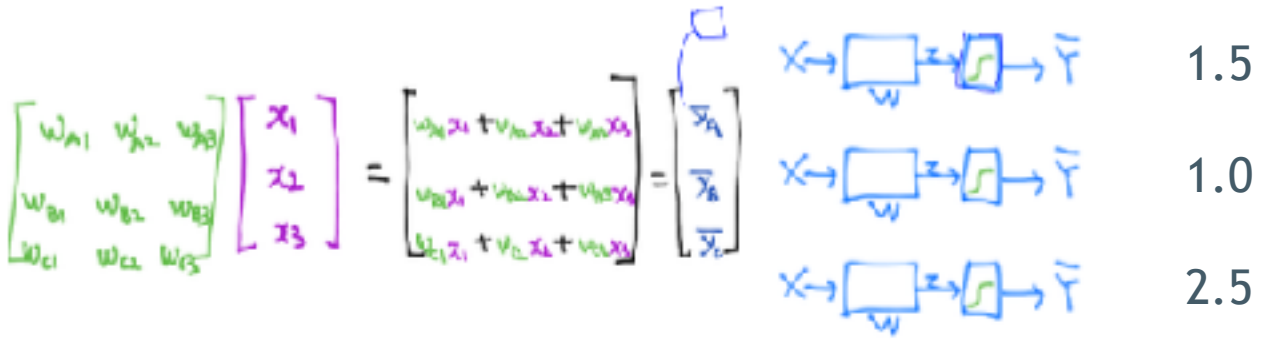
multinomial classification

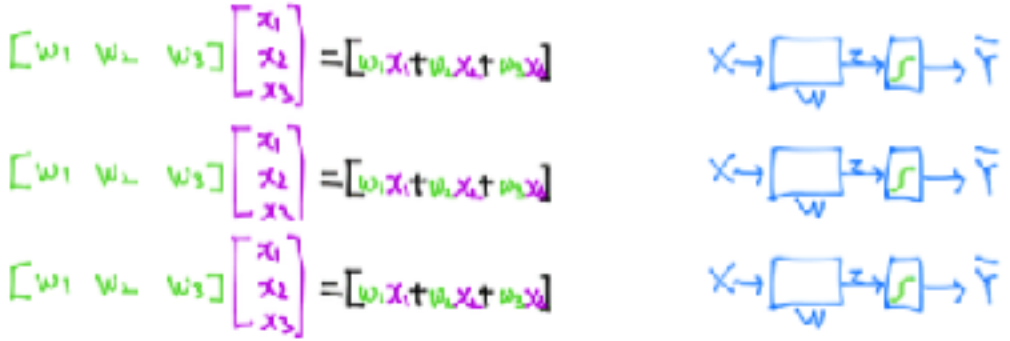
세 개 이상의 분류를 수행하기

IDEA. 여러 개의 binary classification 이 필요

1. A or not A
2. B or not B
3. C or not C







*즉 각각 A일 확률 B일 확률 C일 확률을 구하게 되는 것

multinomial classification -

SOFTMAX FUNCTION

출력값을 0~1 사이의 값으로 정규화하여 총 합이 1 이되게 하는 함수

ONE HOT ENCODING

출력값을 0~1 사이의 값으로 정규화하여 총 합이 1 이되게 하는 함수

CROSS ENTROPY WITH LOGIT

Softmax functio과 one hot edncoding 연산을 포함해서 tf 에서 loss function 을제공

구현

```
# tf.nn.softmax computes softmax activations
# softmax = exp(logits) / reduce_sum(exp(logits), dim)
logits = tf.matmul(X, W) + b
hypothesis = tf.nn.softmax(logits)

# Cross entropy cost/loss
cost_i = tf.nn.softmax_cross_entropy_with_logits(logits=logits,
                                                 labels=Y_one_hot)
cost = tf.reduce_mean(cost_i)
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)

prediction = tf.argmax(hypothesis, 1)
correct_prediction = tf.equal(prediction, tf.argmax(Y_one_hot, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

엔트로피 with logit :
https://taeoh-kim.github.io/blog/cross-entropy%EC%9D%98-%EC%A0%95%ED%99%95%ED%95%9%ED%99%95%EB%A5%A0%EC%A0%81-%EC%9D%98%EB%AF%B8/

학습의 정확도를 확인하는법

데이터 분리

Training data / Test data 로 분리

Test Data 는 y(label value)가 없이 생성된 모델에 넣었을때의 결과값으로 정확도를 측정하는 것.

용어정리

DATA 데이터 처리	Hypothesis function 가설함수 - 모델	Cost function 비용함수 – 모델최적화1	Minimize W,B 최적화
• Trainging data /test data • normalization	• wx+b • Activation function : Sigmoid, tanh, Lrue • Softmax	• 거리 계산 • Log (sigmoid to convex) • One hot encoding • 엔트로피	• gradient decent • convex model • learning rate

- XOR문제풀기
- Backpropagation
- weight setting
- Droupout
- 앙상블
- CNN
- RNN

- Powered by aws gpu, google cloud

Deep Learning

xor문제풀기 , Backpropagation , weight setting, droupout, 앙상블, cnn, rnn

+extra tips

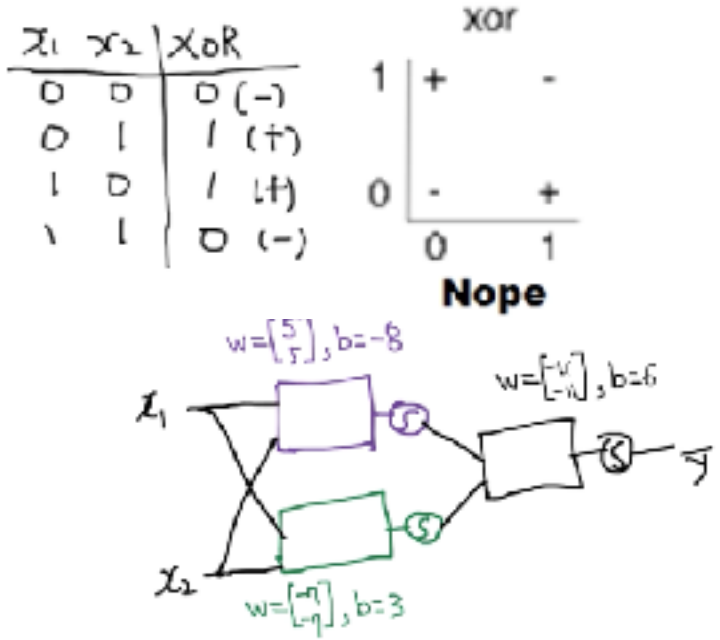
Powered by aws gpu , google cloud

XOR NN – 가능한 W가 있는걸 확인

Issue. XOR 의 경우 단일 리니어로 풀 수 없음

Idea. 여러 리니어 리그레션을 연결해서 풀어보자

- 지금까지는 단일 리니어리그레션을 여러 개를 사용해도 이어서 사용하지는 않았지만 이제 이어서 사용함



x1과 x2는 (0, 0), (0, 1), (1, 0), (1, 1)
W = (5, 5), b = -8
(0*5 + 0*5) + -8 = 0 - 8 = -8 ==> sigmoid(-8) = 0
(0*5 + 1*5) + -8 = 5 - 8 = -3 ==> sigmoid(-3) = 0
(1*5 + 0*5) + -8 = 5 - 8 = -3 ==> sigmoid(-3) = 0
(1*5 + 1*5) + -8 = 10 - 8 = 2 ==> sigmoid(2) = 1

W = (-7, -7), b = 3
(0*-7 + 0*-7) + 3 = 0 + 3 = 3 ==> sigmoid(3) = 1
(0*-7 + 1*-7) + 3 = -7 + 3 = -4 ==> sigmoid(3) = 0
(1*-7 + 0*-7) + 3 = -7 + 3 = -4 ==> sigmoid(3) = 0
(1*-7 + 1*-7) + 3 = -14 + 3 = -11 ==> sigmoid(3) = 0

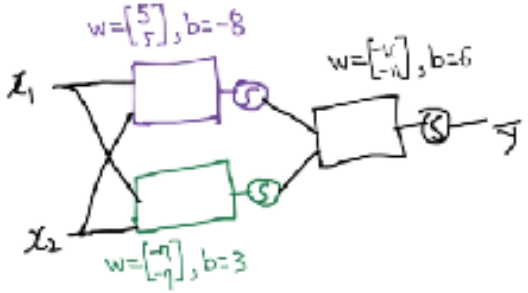
마지막 logistic regression에 전달될 x1과 x2는 이전 결과의 조합이므로 (0, 1), (0, 0), (0, 0), (1, 0)
W = (-11, -11), b = 6
(0*-11 + 1*-11) + 6 = -11 + 6 = -5 ==> sigmoid(-5) = 0
(0*-11 + 0*-11) + 6 = 0 + 6 = 6 ==> sigmoid(6) = 1
(0*-11 + 0*-11) + 6 = 0 + 6 = 6 ==> sigmoid(6) = 1
(1*-11 + 0*-11) + 6 = -11 + 6 = -5 ==> sigmoid(-5) = 0

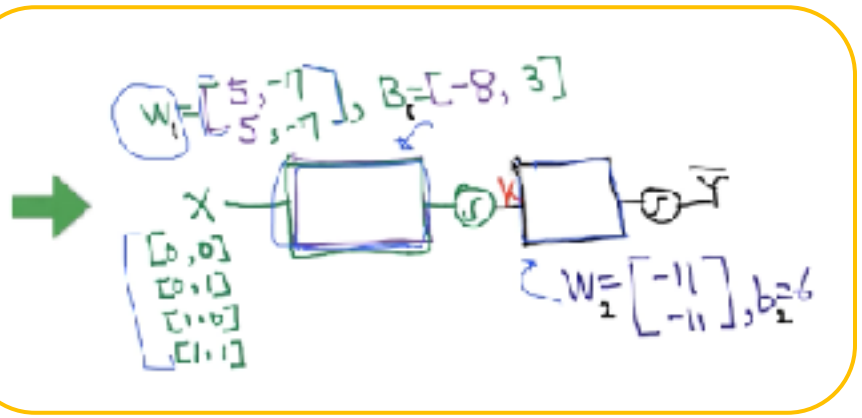
XOR NN

Issue. XOR 의 경우 단일 리니어로 풀 수 없음

Idea. 여러 리니어 리그레션을 연결해서 풀어보자

- 지금까지는 단일 리니어리그레션을 여러 개를 사용해도 이어서 사용하지는 않았지만 이제 이어서 사용함





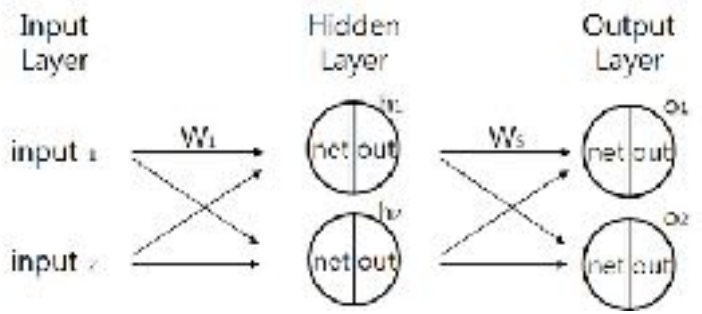
레이어를 쌓는 방법

```
with tf.name_scope('layer1') as scope:
    W1 = tf.Variable(tf.random_normal([2, 2]), name='weight1')
    b1 = tf.Variable(tf.random_normal([2]), name='bias1') # output2ro
    layer1 = tf.sigmoid(tf.matmul(X, W1) + b1) # bias는 아웃풋 갯수

    w1_hist = tf.summary.histogram("weights1", W1)
    b1_hist = tf.summary.histogram("biases1", b1)
    layer1_hist = tf.summary.histogram("layer1", layer1)


with tf.name_scope("layer2") as scope:
    W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
    b2 = tf.Variable(tf.random_normal([1]), name='bias2')
    hypothesis = tf.sigmoid(tf.matmul(layer1, W2)   b2)

    w2_hist = tf.summary.histogram("weights2", W2)
    b2_hist = tf.summary.histogram("biases2", b2)
    hypothesis_hist = tf.summary.histogram("hypothesis", hypothesis)
```

lab-09-5-linear_back_prop.py

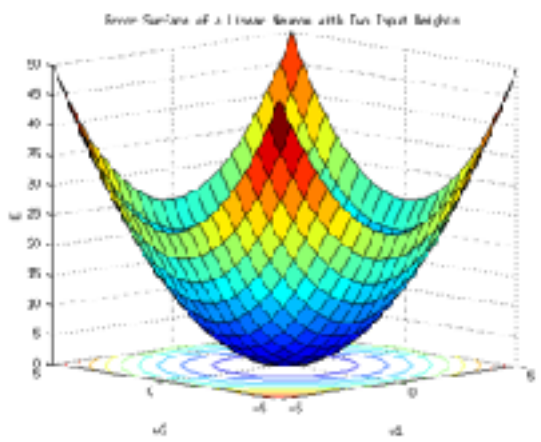
Back-propagation

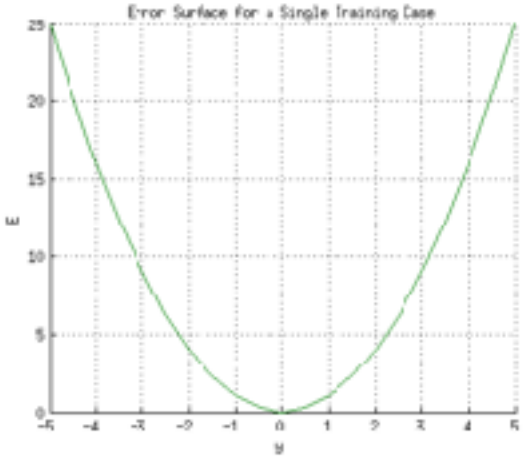
: w,b 들이 결과에 어느 정도의 영향 주는지 알아내는 방법



Issue. Layer가 깊어지면서 해당 결과값에 대해 다시 그 w,b가 얼마만큼의 영향력을 가지는지 알 수 없음
Idea. 미분을 통해서 영향력을 계산해 낼 수 있음
+ 물론 이부분도 구현은 프로그램이 해주지만 중요한 부분이니까 알아두기

F = wx+b , g=wx, f=g+b





*즉 각각 A일 확률 B일 확률 C일 확률을 구하게 되는 것

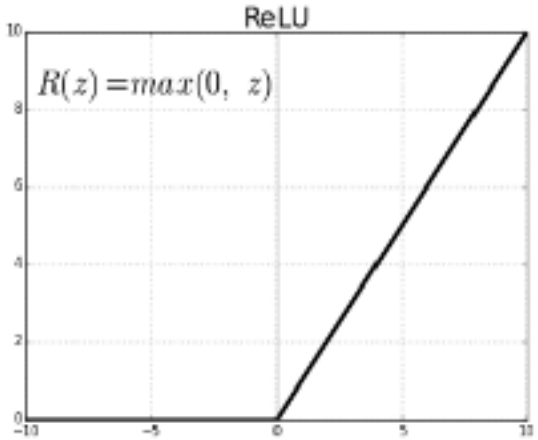
성능을 높이는 방법

① 초기값을 잘 주기

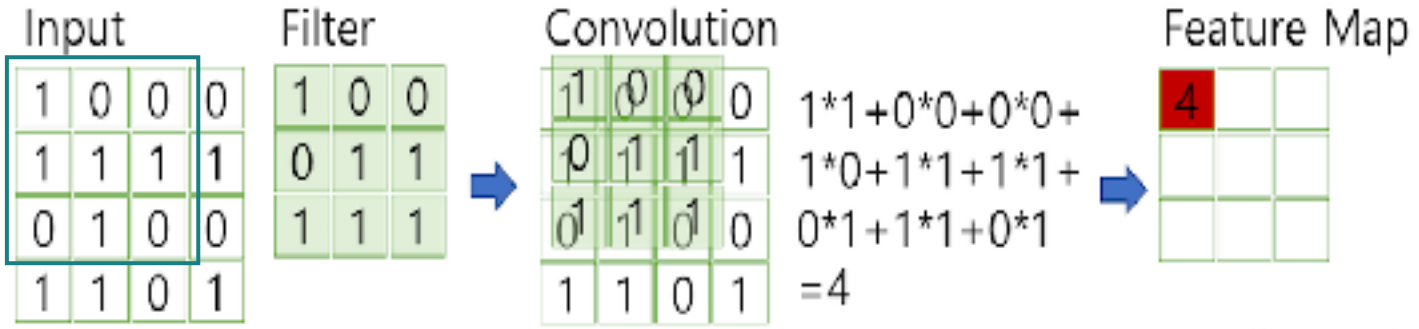
② 오버피팅을 막기 : 더 많은 데이터, 정규화, dropout

③ Ensemble

④ Vanishing Gradient 문제 막기 : relu사용



드디어 CNN



Issue. 이미지를 처리해야 하는데 픽셀값으로 처리해야함.

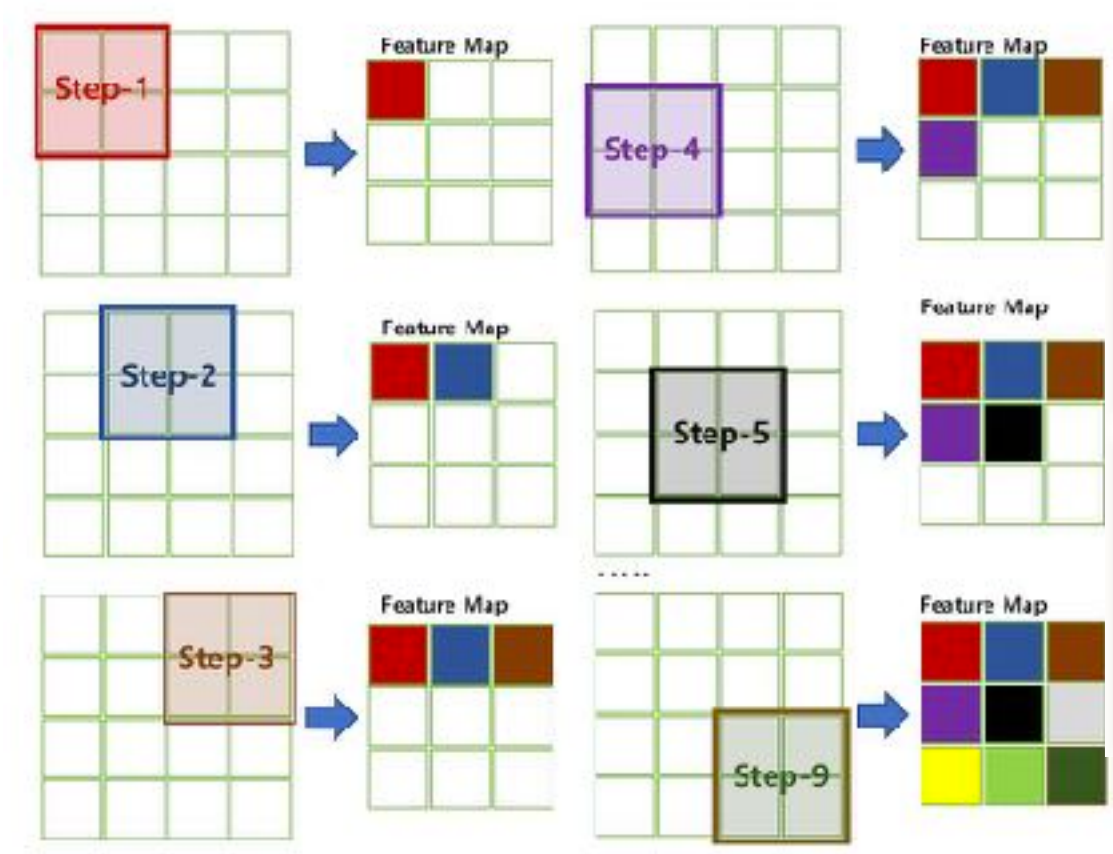
Idea. 이미지를 뭉탱이로 잘라서 X1 x2~~~~xn의 데이터를 레이어에 다 넣어버린다

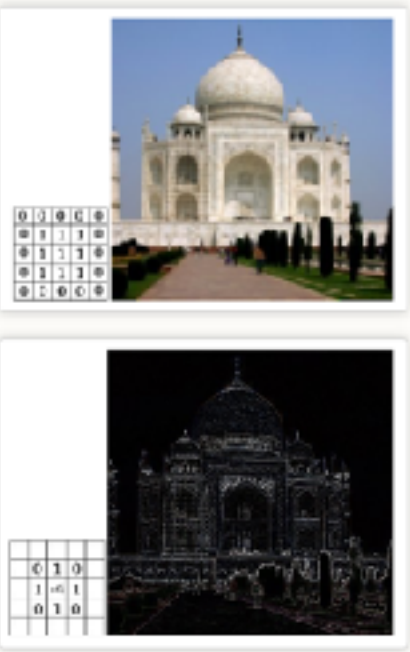
알아야 하는것

- **Conv layer 만들기** : 32 by 32 image 에서 1 filter 당 1장의 feature map 을 뽑아냄
  - Stride : 움직이는 칸 수
  - Filter : input image 에서 값을 뽑아내는 사이즈
  - **Padding 처리(위에 포함)** : conv layer 로 뽑아내는 feature map이 작아지지 않도록 처리.

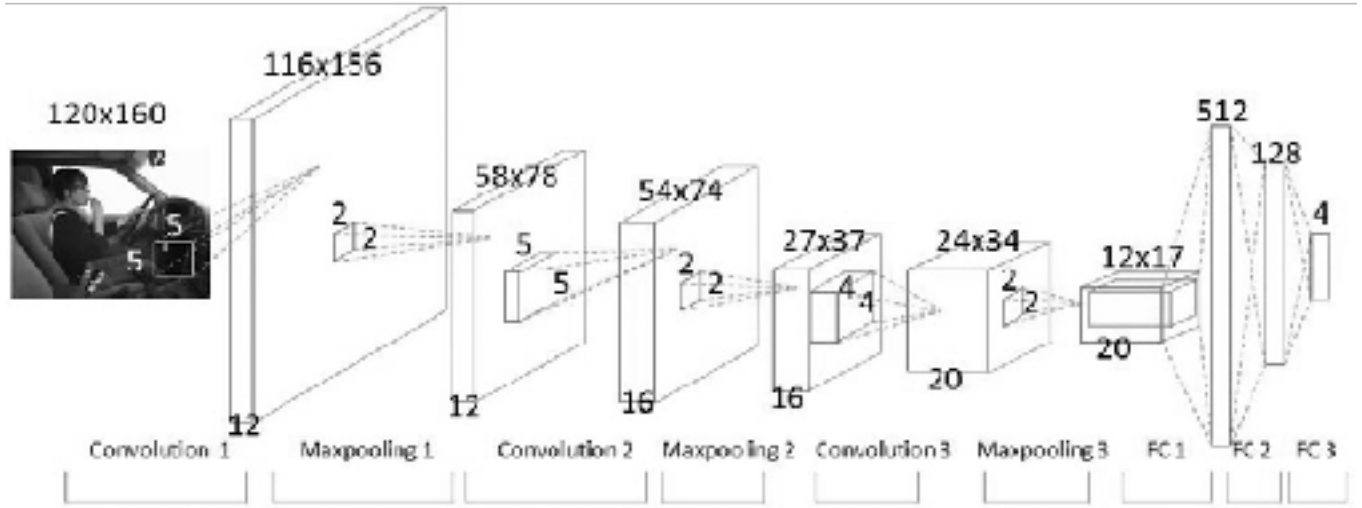
- **Pooling layer** : conv layer 다음에 풀링 처리를 하는데 이는 피처 맵의 크기를 줄어주는 것 – classification model 에서는 주로 max pooling 사용(해상도줄이기)

* 위 세개를 쌓아서





cnn



* Conv layer + maxpooling => 를 쌓아서 만든다.

그럼 RNN은?

multinomial classification = softmax classification (논리회귀)

Idea. 여러 개의 binary classification 이 필요하다
1. A or not A
2. B or not B

```
# Lab 4 Multi-variable linear regression
import tensorflow as tf
tf.set_random_seed(777)  # for reproducibility

x_data = [[73., 80., 75.],
          [93., 88., 93.],
          [89., 91., 90.],
          [96., 98., 100.],
          [73., 66., 70.]]
y_data = [[152.],
          [185.],
          [180.],
          [196.],
          [142.]]
```

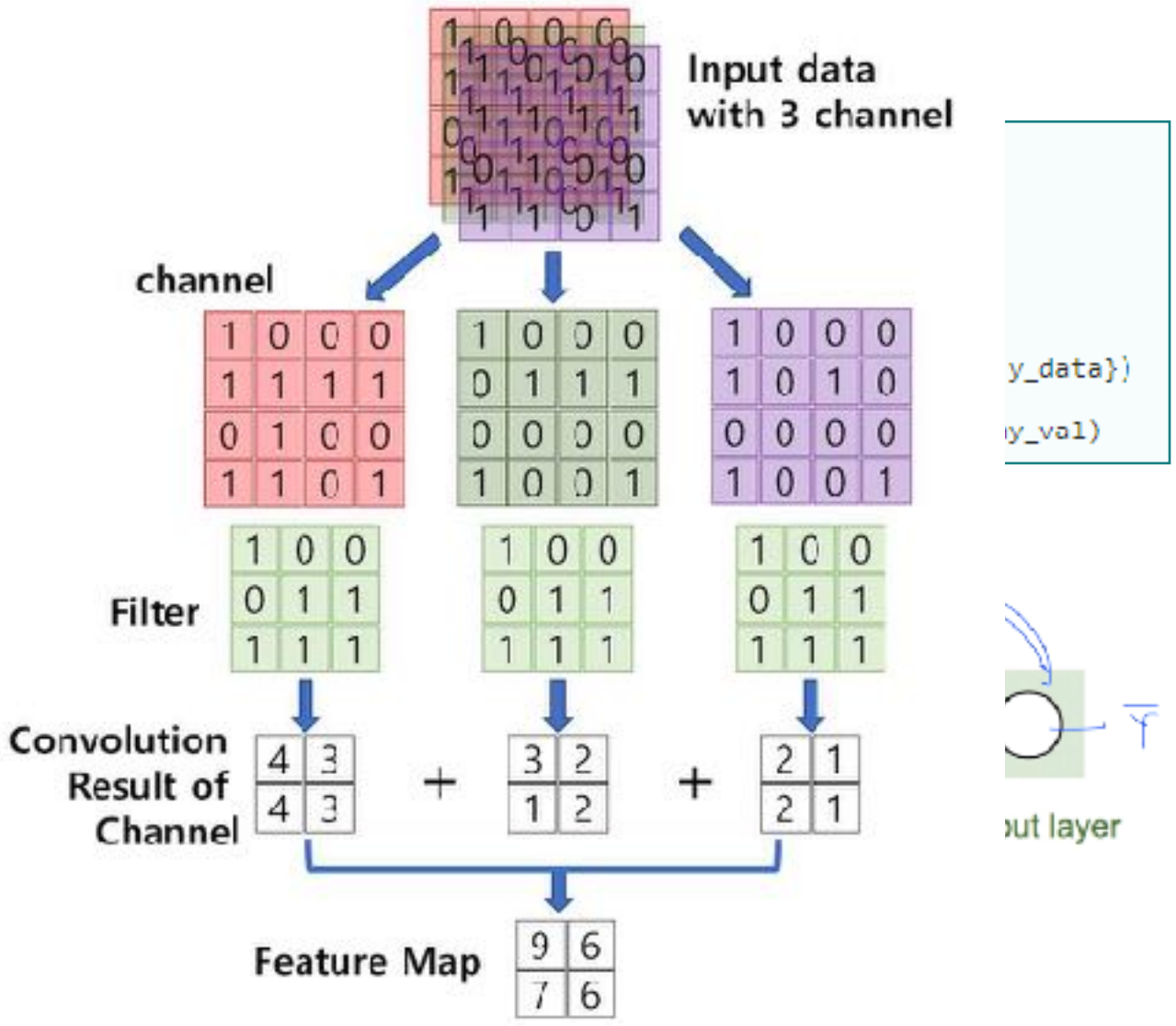
데이터 준비 -> 파일로 분리

데이터 크기에 따라 변수 준비

```
# placeholders for a tensor that will be always fed
X = tf.placeholder(tf.float32, shape=[None, 3])
Y = tf.placeholder(tf.float32, shape=[None, 1])

3 1 3 3 3 3
3 1 3 3 3 3
3 1 3 3 3 3
3 1 3 3 3 3
3 1 3 3 3 3
3 1 3 3 3 3

om_normal([3, 1]), name='wei
om_normal([1]), name='bias')
```

, 미니마이즈 함수 준비

```
, W) + b

unction
f.square(hypothesis - Y))

adientDescentOptimizer(learn
train = optimizer.minimize(cost)
```



multi_variable_matmul_linear_regression.py