메디치소프트 기술연구소

2018.07.26

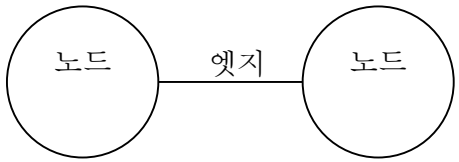
AI 컨설팅 방법론{딥러닝}

버전	작성자	작성일자	비고
V 0.1	한미란	2018. 07.26	

TensorFlow 정의

정의

.Open source software library for numerical computation using data flow graphs.
..python!



.Nodes in the graph represent mathematical operations
.Edges represent the multidimensional data arrays(tensors) communicated between them

. Linux, Max OSX, Windows
(sudo –H) pip install –upgrade tensorflow
(sudo –H) pip install –uograde tensorflow-gpu
https://www.python.org/downloads/release/python-360/

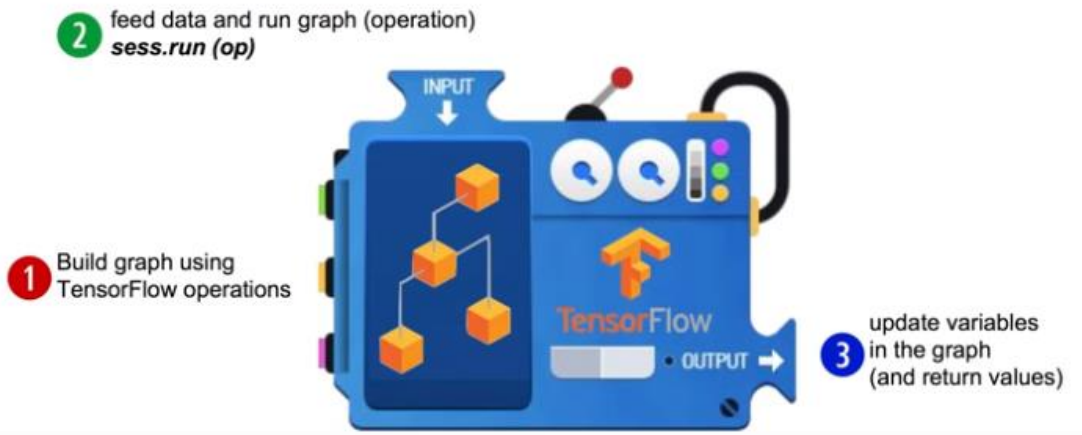
. Google search/Community help
https://www.facebook.com/groups/TensorFlowKR

```
Sungs-MacBook-Pro:hunkim$ python3
Python 3.6.0 (v3.6.0:41df79263a11, Dec 22 2016, 17:23:13)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "help", "copyright", "credits" or "license" for more
information.
>>> import tensorflow as tf
>>> tf.__version__
'1.0.0'
>>>
```

TensorFlow Mechanics(www.mathwarehouse.com)

. Hello World

```
# Create a constant op
# This op is added as a node to the default graph

  hello = tf.constant("Hello, TensorFlow!")
# start a TF session

  sess = tf.Session() #
   run the op and get result

(sess.run(hello))
```

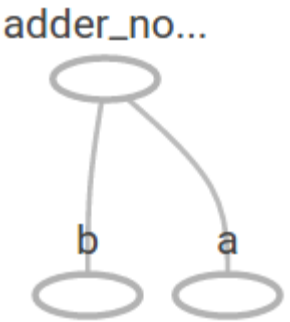


. Computational Graph

```
node1 = tf.constant(3.0, tf.float32)

node2 = tf.constant(4.0) # also tf.float32 implicitly

node3 = tf.add(node1, node2)
```

```
print("node1:", node1, "node2:", node2)
print("node3: ", node3)
```

```
sess = tf.Session()

print("sess.run(node1, node2): ", sess.run([node1, node2]))

print("sess.run(node3): ", sess.run(node3))
```



. Placeholder

```
a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)
adder_node = a + b # + provides a shortcut for tf.add(a, b)
print(sess.run(adder_node, feed_dict={a: 3, b: 4.5}))
print(sess.run(adder_node, feed_dict={a: [1,3], b: [2, 4]}))
```

```
add_and_triple = adder_node * 3.
(sess.run(add_and_triple, feed_dict={a: 3, b:4.5}))
```

. Tensor Ranks, Shapers, and Types

Tensors

```
3 # a rank 0 tensor; this is a scalar with shape []
[1. ,2., 3.] # a rank 1 tensor; this is a vector with shape [3]
[[1., 2., 3.], [4., 5., 6.]] # a rank 2 tensor; a matrix with shape [2, 3]
[[[1., 2., 3.]], [[7., 8., 9.]]] # a rank 3 tensor with shape [2, 1, 3]
```

```
t = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
```

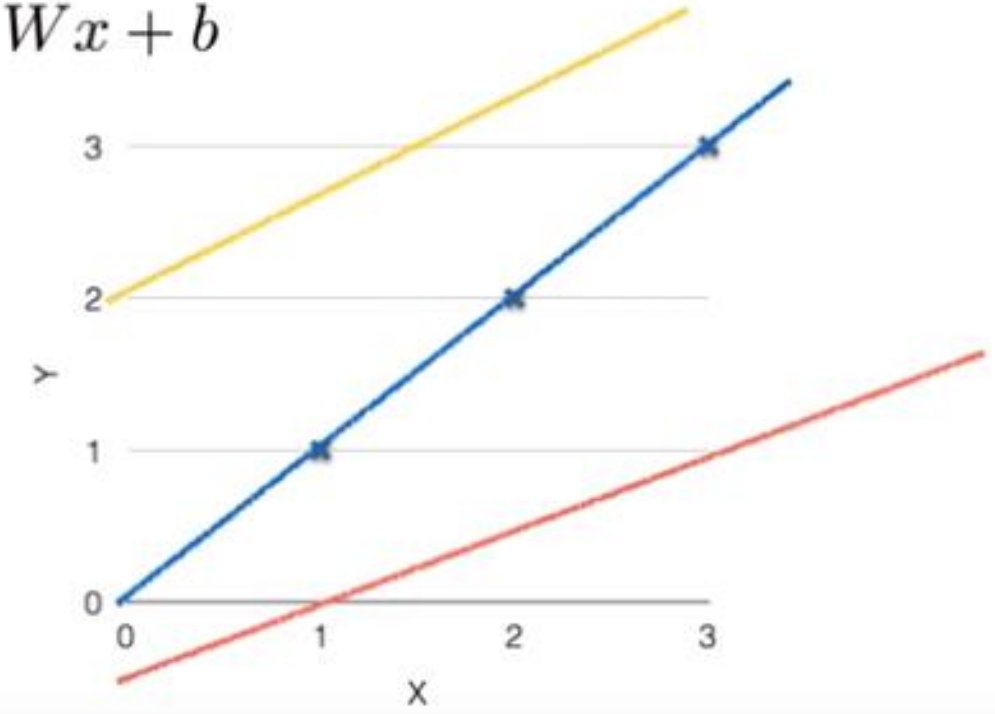
Rank	Math entity	Python example
0	Scalar (magnitude only)	`s = 483`
1	Vector (magnitude and direction)	`v = [1.1, 2.2, 3.3]`
2	Matrix (table of numbers)	`m = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]`
3	3-Tensor (cube of numbers)	`t = [[[2], [4], [6]], [[8], [10], [12]], [[14], [16], [18]]]`
n	n-Tensor (you get the idea)	`....`

Data type	Python type	Description
`DT_FLOAT`	`tf.float32`	32 bits floating point.
`DT_DOUBLE`	`tf.float64`	64 bits floating point.
`DT_INT8`	`tf.int8`	8 bits signed integer.
`DT_INT16`	`tf.int16`	16 bits signed integer.
`DT_INT32`	`tf.int32`	32 bits signed integer.
`DT_INT64`	`tf.int64`	64 bits signed integer.

Rank	Shape	Dimension number	Example
0	[]	0-D	A 0-D tensor. A scalar.
1	[D0]	1-D	A 1-D tensor with shape [5].
2	[D0, D1]	2-D	A 2-D tensor with shape [3, 4].
3	[D0, D1, D2]	3-D	A 3-D tensor with shape [1, 4, 3].
n	[D0, D1, ... Dn-1]	n-D	A tensor with shape [D0, D1, ... Dn-1].

(Linear) Hypothesis

$$H(x) = Wx + b$$

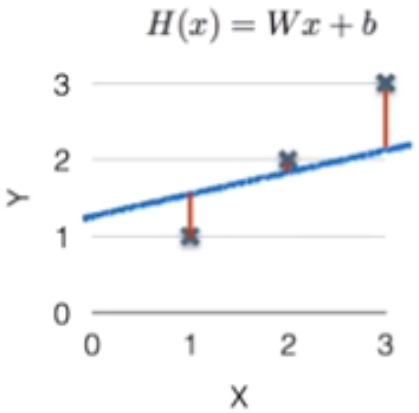


Cost function

- How fit the line to our (training) data

$$H(x) - y$$

$$\frac{(H(x^{(1)}) - y^{(1)})^2 + (H(x^{(2)}) - y^{(2)})^2 + (H(x^{(3)}) - y^{(3)})^2}{3}$$



$$cost = \frac{1}{m}\sum^{m}(H(x^{(i)}) - y^{(i)})^2$$

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

. Linear Regression : how to minimize cost

1. Build graph using TF operations

```
#X and Y data
 x_train = [1.2.3]
 y_train = [1,2,3]
W= tf.Variable(tf.random_noram([1]), name='weight')
b  = tf.Variable(tf.random_noram([1]), name='bias')
# Our yhpothesis XW+b
Hypothesis = x_train*w + b
```

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

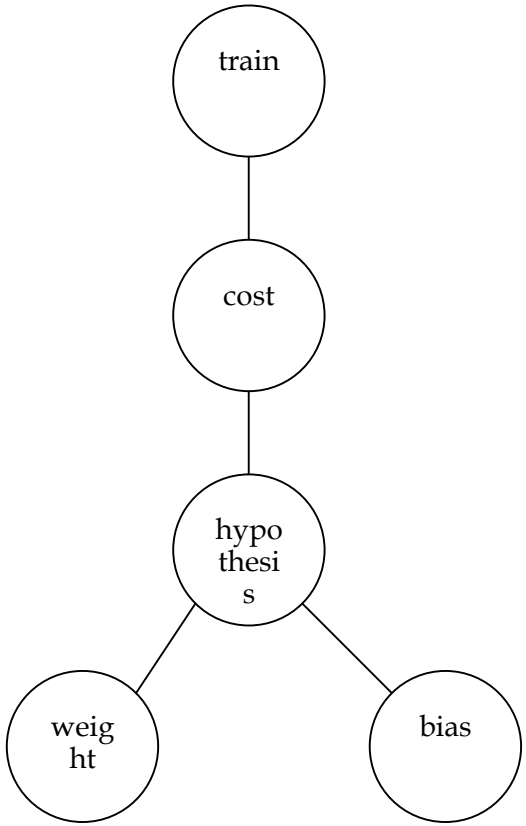
```
#cost/Loss function
Cost = tf.reduce_mean(tf.square(hypothesis – y_train))
```

GradientDescent

```
# Minimize
Optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)
train = optimizer.minimize(cost)
```

2.3 Run/update graph and get results

```
#Launch the graph in a session
sess = tf.Session()
#Initializes global variables in the graph.
 sess.run(tf.global_variables_initializer())
#Fit the line
 for step in range(2001):
        sess.run(train)
         if step % 20 == 0:
            print(step, sess.run(cost), sess.run(W), sess.run(b))
```



메디치소프트 기술연구소

2018.07.26

How to minimize cost

MEDICISOFT

. Hypothesis and Cost

$$H(x) = Wx + b$$

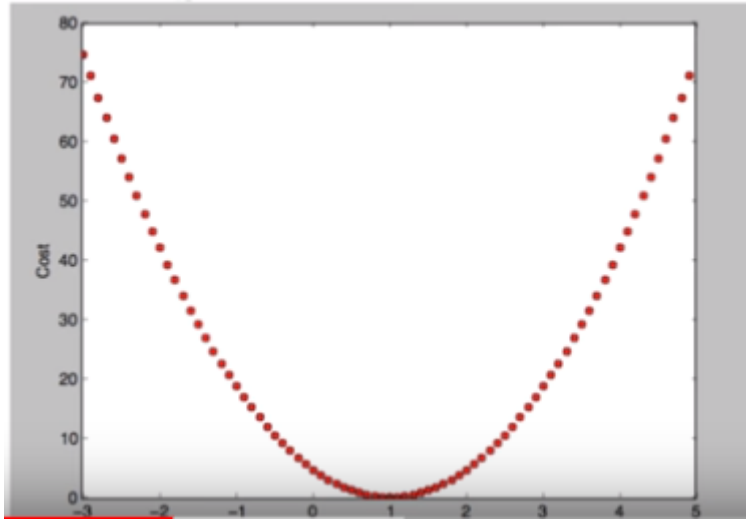
$$cost(W, b) = \frac{1}{m}\sum_{i=1}^{m}(H(x^{(i)}) - y^{(i)})^2$$

x	y
1	1
2	2
3	3

- **W =1 , cost(W) =0**

$$\frac{1}{3}((1*1-1)^2 + (1*2-2)^2 + (1*3-3)^2)$$

- **W =0 , cost(W) =4.67**
- **W =2 , cost(W) =4.67**



. Gradient descent algorithm

. Formal definition

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

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

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

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

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

$$W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$$

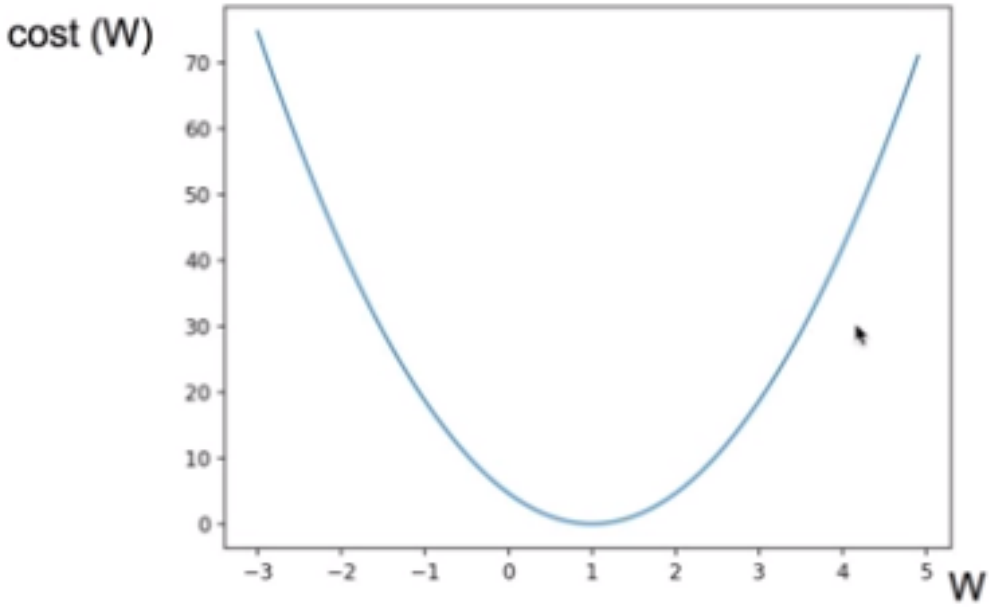
Simplified Hypothesis

```python
import tensorflow as tf
import matplotlib.pyplot as plt
tf.set_random_seed(777)  # for reproducibility
X = [1, 2, 3]
Y = [1, 2, 3]
W = tf.placeholder(tf.float32)
# Our hypothesis for linear model X * W
hypothesis = X * W
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Launch the graph in a session.
sess = tf.Session()
# Variables for plotting cost function
W_history = []
cost_history = []
for i in range(-30, 50):
    curr_W = i * 0.1
    curr_cost = sess.run(cost, feed_dict={W: curr_W})
    W_history.append(curr_W)
    cost_history.append(curr_cost)
# Show the cost function
plt.plot(W_history, cost_history)
plt.show()
```

$$H(x) = Wx$$

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

Gradient descent



$$W := W - \alpha \frac{1}{m} \sum_{i=1}^{m} (W x^{(i)} - y^{(i)}) x^{(i)}$$

```
# Minimize: Gradient Descent using derivative:
W -= learning_rate * derivative
learning_rate = 0.1
gradient = tf.reduce_mean((W * X - Y) * X)
descent = W - learning_rate * gradient
update = W.assign(descent)
```

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

Simplified Hypothesis

```
x_data = [1, 2, 3]
y_data = [1, 2, 3]
# Try to find values for W and b to compute y_data = W * x_data + b
# We know that W should be 1 and b should be 0
# But let's use TensorFlow to figure it out
W = tf.Variable(tf.random_normal([1]), name='weight')
X = tf.placeholder(tf.float32)
Y = tf.placeholder(tf.float32)
# Our hypothesis for linear model X * W
hypothesis = X * W
# cost/loss function
cost = tf.reduce_mean(tf.square(hypothesis - Y))
# Minimize: Gradient Descent using derivative: W -= learning_rate * derivative
learning_rate = 0.1
gradient = tf.reduce_mean((W * X - Y) * X)
descent = W - learning_rate * gradient
update = W.assign(descent)
# Launch the graph in a session.
sess = tf.Session()
# Initializes global variables in the graph.
sess.run(tf.global_variables_initializer())

for step in range(21):
    sess.run(update, feed_dict={X: x_data, Y: y_data})
    print(step, sess.run(cost, feed_dict={X: x_data, Y: y_data}), sess.run(W))
```

```
# Minimize: Gradient Descent Magic
optimizer =
  tf.train.GradientDescentOptimizer(learning_rate=0.1)
train = optimizer.minimize(cost)
```

- Hypothesis

$$H(x) = Wx + b$$

- Cost function

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

- Gradient descent algorithm

multi-variable linear regression (*new)

multi-variable/feature

x_1 (quiz 1)	x_2 (quiz 2)	x_3 (midterm 1)	Y (final)
73	80	75	152
93	88	93	185
89	91	90	180
96	98	100	196
73	66	70	142

$$H(x) = Wx + b$$

$$H(x_1, x_2, x_3) = w_1x_1 + w_2x_2 + w_3x_3 + b$$

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

multi-variable linear regression (*new)

Matrix multiplication

$$\begin{Bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{Bmatrix} \times \begin{Bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{Bmatrix} = \begin{Bmatrix} \quad \end{Bmatrix}$$

$$\begin{pmatrix} x_1 & x_2 & x_3 \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_1 w_1 + x_2 w_2 + x_3 w_3 \end{pmatrix}$$

$$H(X) = XW$$

multi-variable linear regression (*new)

$$w_1x_1 + w_2x_2 + w_3x_3 + ... + w_nx_n$$

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \\ x_{41} & x_{42} & x_{43} \\ x_{51} & x_{52} & x_{53} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = \begin{pmatrix} x_{11}w_1 + x_{12}w_2 + x_{13}w_3 \\ x_{21}w_1 + x_{22}w_2 + x_{23}w_3 \\ x_{31}w_1 + x_{32}w_2 + x_{33}w_3 \\ x_{41}w_1 + x_{42}w_2 + x_{43}w_3 \\ x_{51}w_1 + x_{52}w_2 + x_{53}w_3 \end{pmatrix}$$

$$H(X) = XW$$

[5 ,3] [3, 1] [? ?]

- Lecture (theory):

$$H(x) = Wx + b$$

- Implementation (TensorFlow)

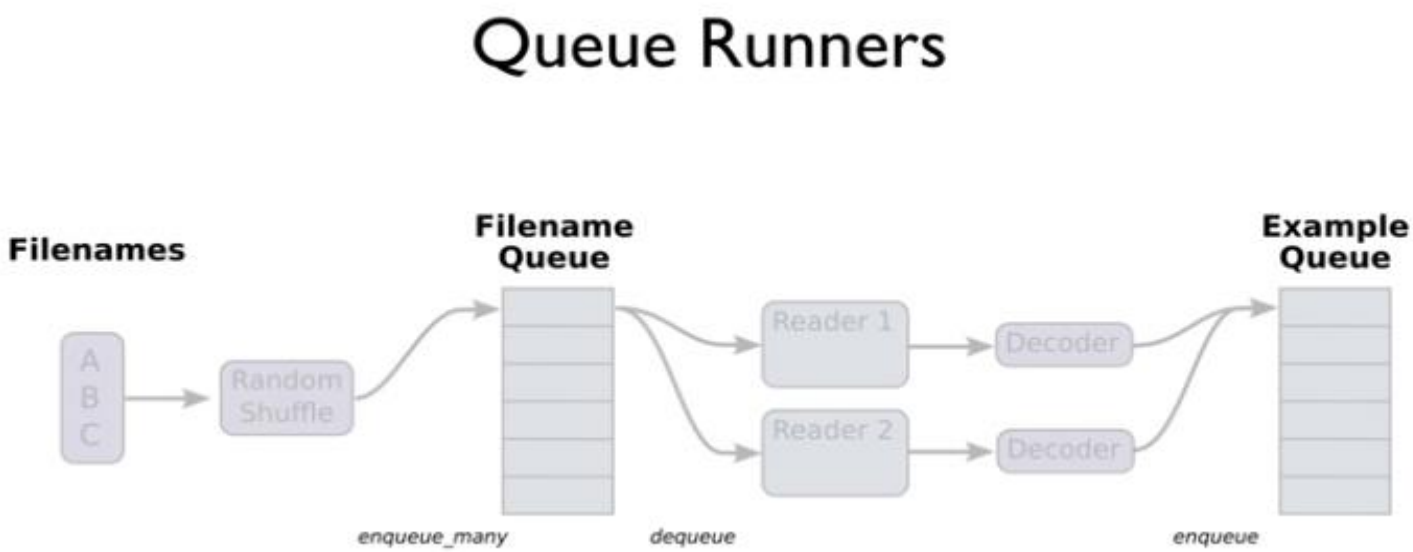
$$H(X) = XW$$

Multivariable 예제 1, 2 실습

Loading data from file

```
import numpy as np

xy = np.loadtxt('test-score.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]y_data = xy[:, [-1]]

print(x_data.shape, x_data, len(x_data))
print(y_data.shape, y_data, len(x_data))
```

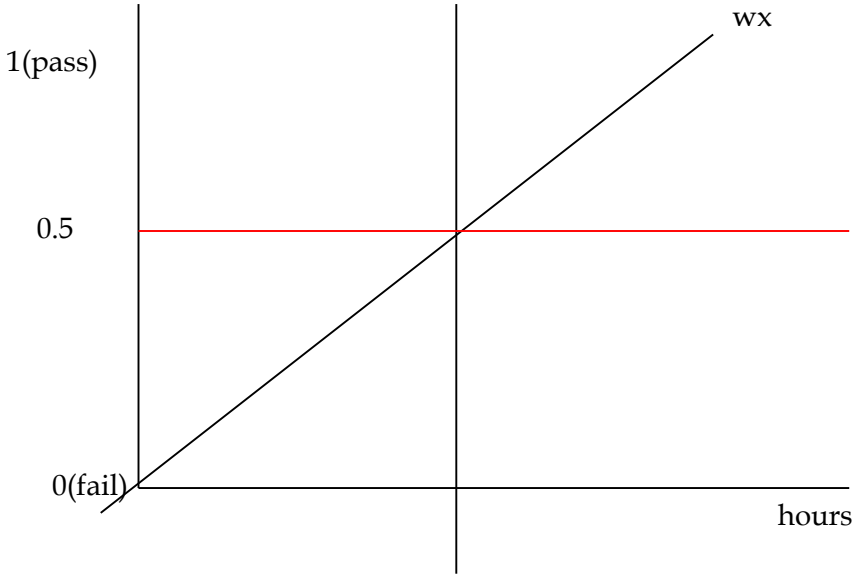
```
nums = range(5)    # range is a built-in function that creates a list of integers
print nums         # Prints "[0, 1, 2, 3, 4]"
print nums[2:4]    # Get a slice from index 2 to 4 (exclusive); prints "[2, 3]"
print nums[2:]     # Get a slice from index 2 to the end; prints "[2, 3, 4]"
print nums[:2]     # Get a slice from the start to index 2 (exclusive); prints "[0, 1]"
print nums[:]      # Get a slice of the whole list; prints ["0, 1, 2, 3, 4]"
print nums[:-1]    # Slice indices can be negative; prints ["0, 1, 2, 3]"
nums[2:4] = [8, 9] # Assign a new sublist to a slice
print nums         # Prints "[0, 1, 8, 9, 4]"
```




```
filename_queue = tf.train.string_input_producer(
    ['data-01-test-score.csv'], shuffle=False, name='filename_queue')

reader = tf.TextLineReader()
key, value = reader.read(filename_queue)

record_defaults = [[0.], [0.], [0.], [0.]]
xy = tf.decode_csv(value, record_defaults=record_defaults)
```

Logistic regression

스팸 : spam or ham
페이스북: show or hide
사기 결제: legitimate/fraud



0에서 1사이에 나와야 하나
X = [1,2,3,4,10] w=0.5 일경우

X가 100 y=50

$$g(z) = \frac{1}{(1+e^{-z})}$$

Sigmoid: S자 곡선 (logistic function)

$$H(X) = \frac{1}{1+e^{-W^T X}}$$

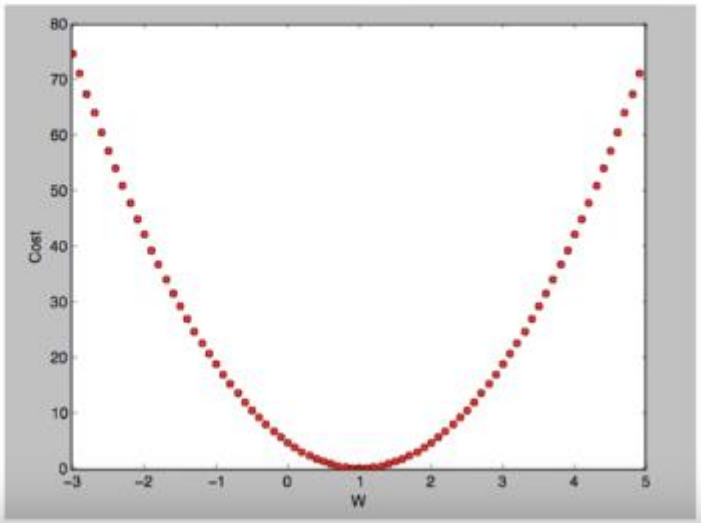
- We know Y is 0 or 1

$$H(x) = Wx + b$$

- Hypothesis can give values large than 1 or less than 0

Logistic regression

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



$$cost(W) = \frac{1}{m}\sum \quad c(H(x), y)$$

$$H(X) = \frac{1}{1 + e^{-W^T X}} \longrightarrow$$

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

$$\longrightarrow c(H(x), y) = ylog(H(x)) - (1 - y)log(1 - H(x))$$

Minimize cost- Gradient decent algorithm

$$cost(W) = -\frac{1}{m}\sum ylog(H(x)) + (1-y)log(1-H(x))$$

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

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

# Minimize
a = tf.Variable(0.1) # Learning rate, alpha
optimizer = tf.train.GradientDescentOptimizer(a)
train = optimizer.minimize(cost)
```