강화학습 스터디 2주차
Lec. 5-7

2020.1.20.

서울대학교 조선해양공학과 생산공학 연구실
유상현

Lec.1-4 Summary

- **Lec1**
  - Machine Learning
  - 지도/비지도/강화학습
- **Lec2-4**
  - Linear Regression
  - HCG (Hypothesis, Cost, Gradient Descent Algorithm -> minimize cost)

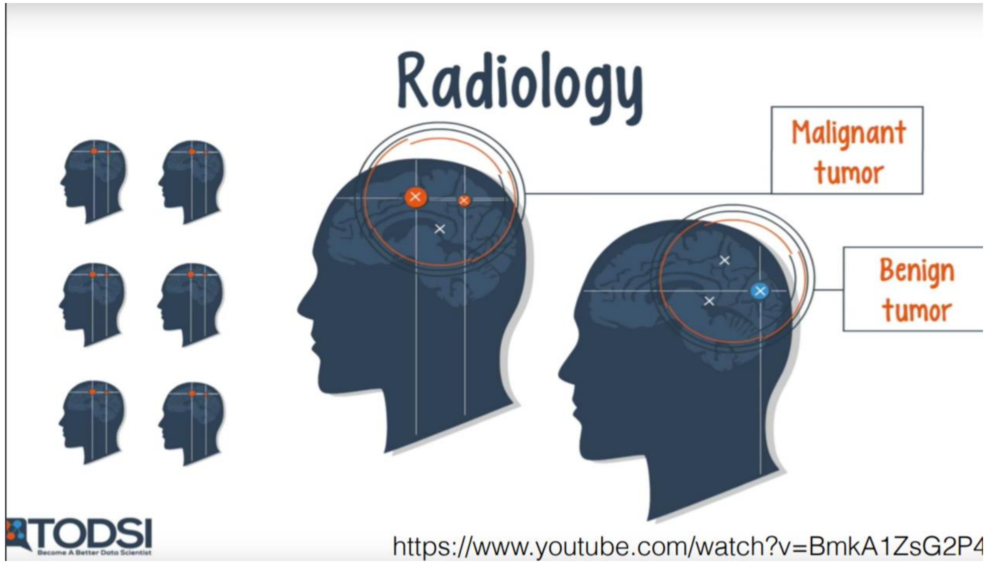
$$W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

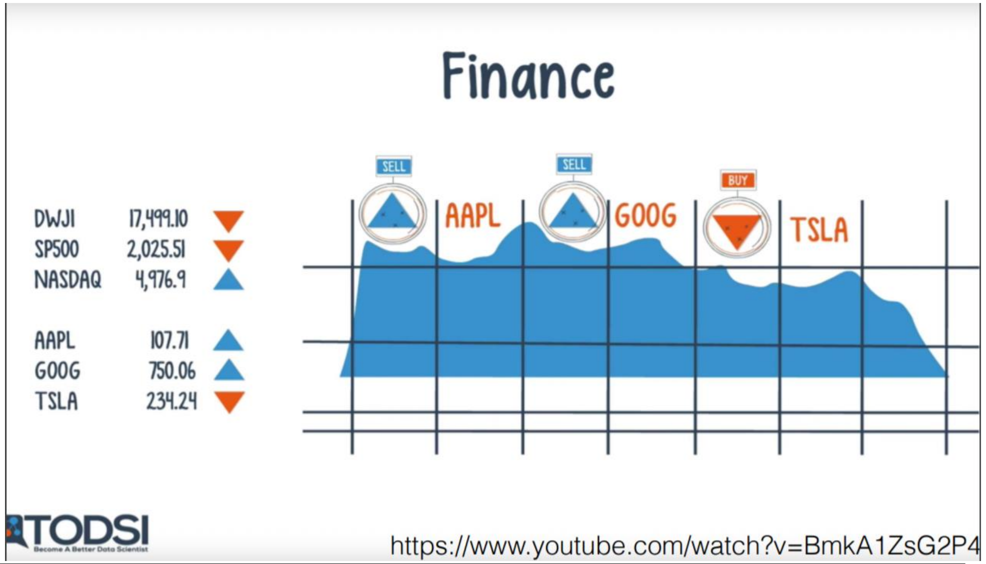
  - Multivariable

◆ Logistic (Regression) Classification

0, 1 encoding

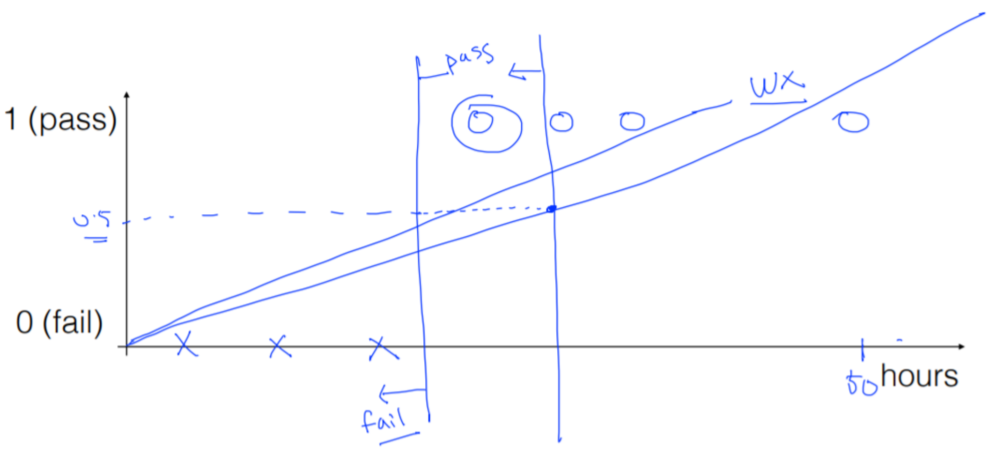
- Spam Detection: Spam (1) or Ham (0)
- Facebook feed: show(1) or hide(0)
- Credit Card Fraudulent Transaction detection: legitimate(0) or fraud (1)

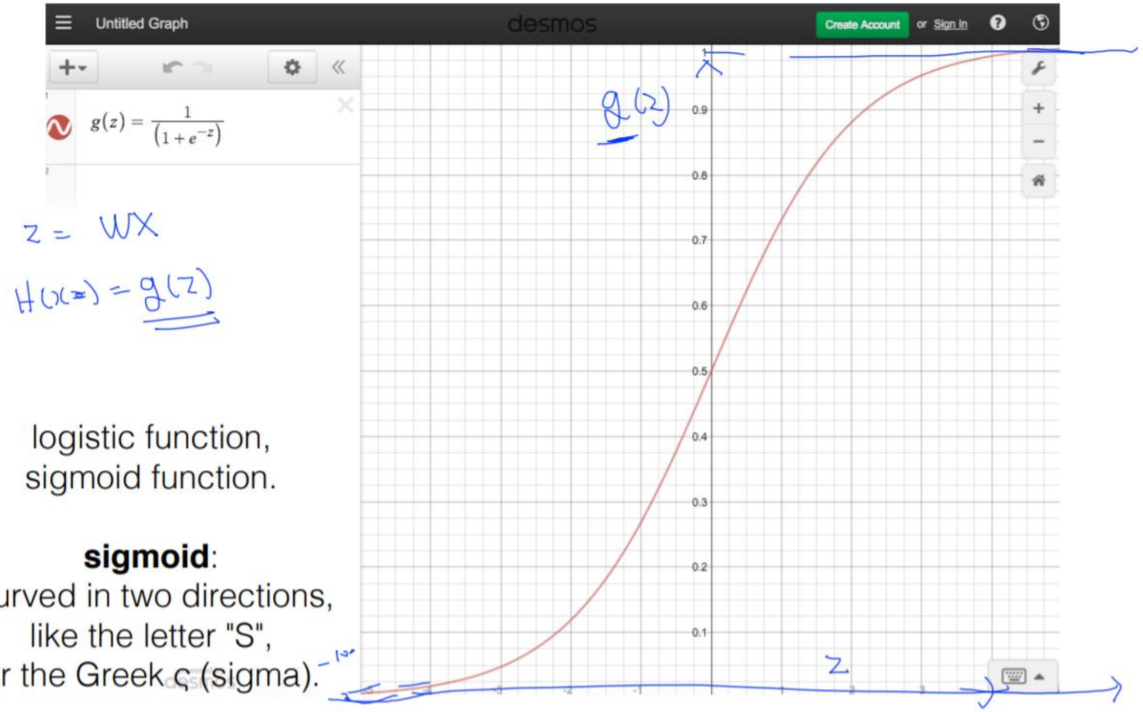




◆ Sigmoid

Linear Regression?



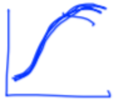


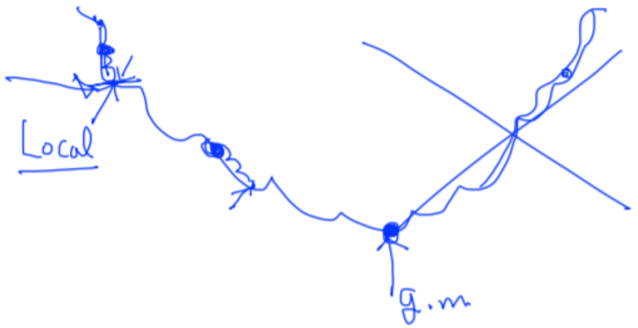
logistic function,
sigmoid function.

sigmoid:
Curved in two directions,
like the letter "S",
or the Greek ς (sigma).

◆ Cost Function

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





understanding cost function

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

$1+e^{-z}$

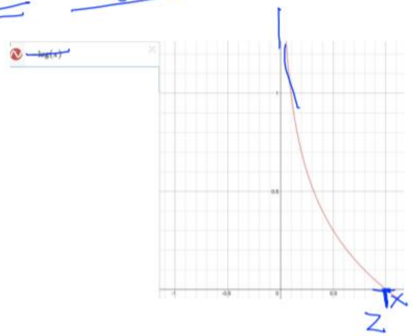
log

Cost $\boxed{y=1}$

$H(x)=1 \rightarrow cost(1)=0$

$H(x)=0 \rightarrow cost=\infty$

cost = $g(z) = -\log(z)$

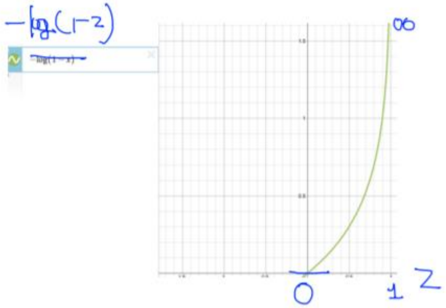


$y=0$

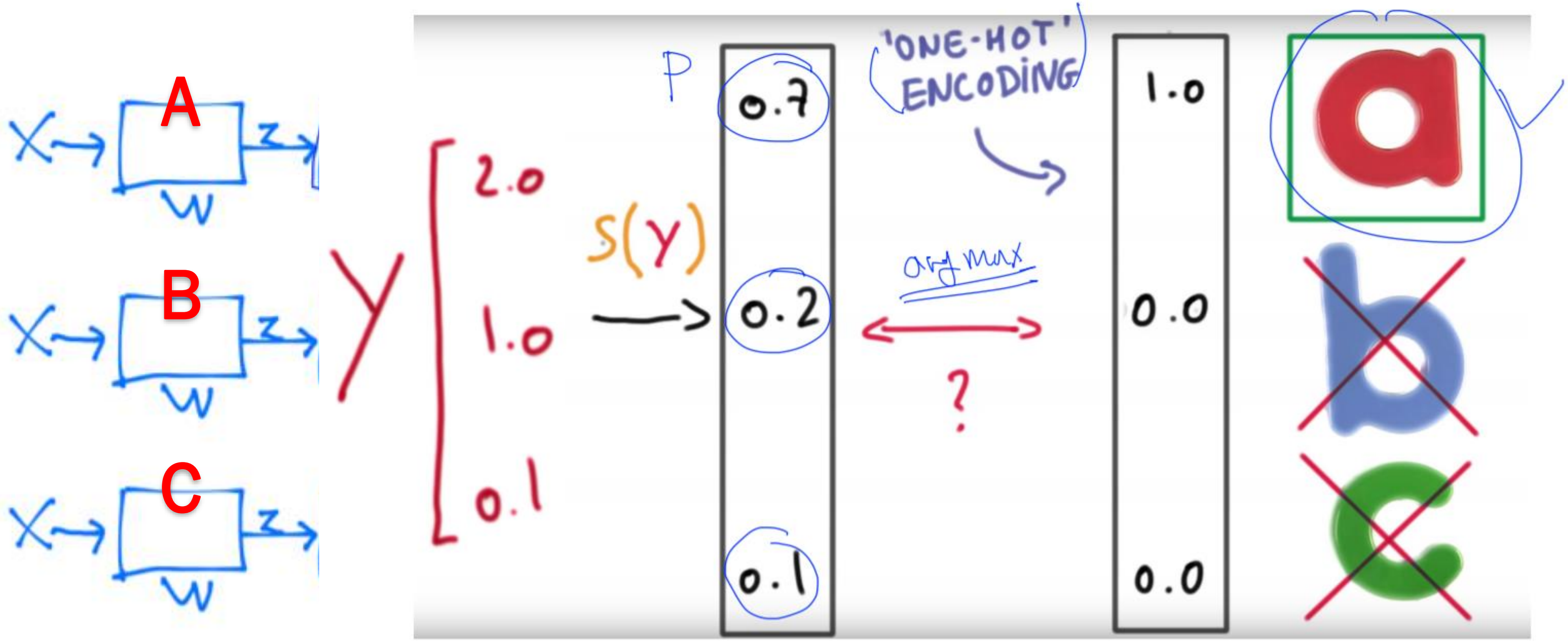
$H(x)=0$, $cost=0$

$H(x)=1$, $cost=\infty$

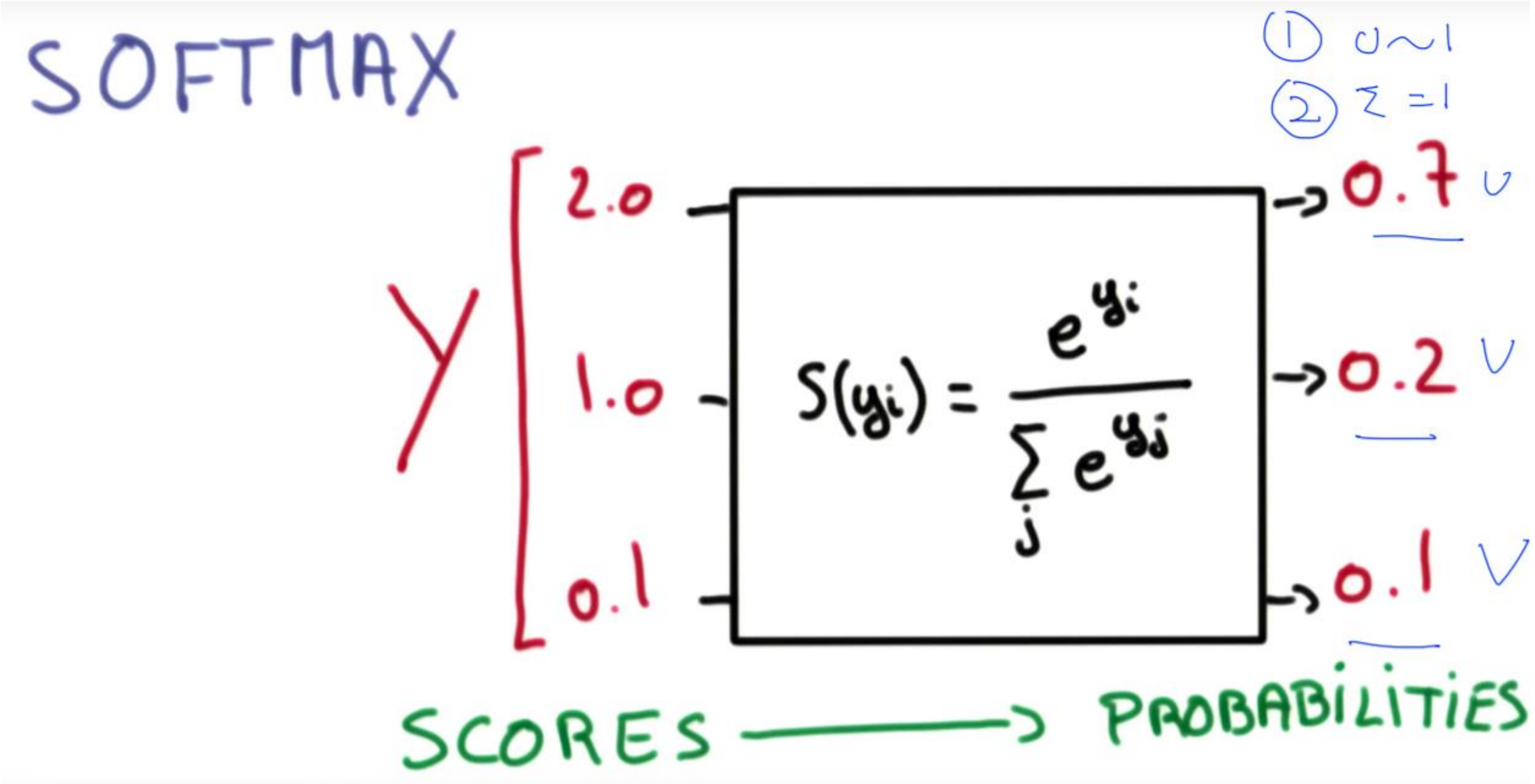
$-\log(1-z)$



◆ Softmax Classification (Multinomial Classification)



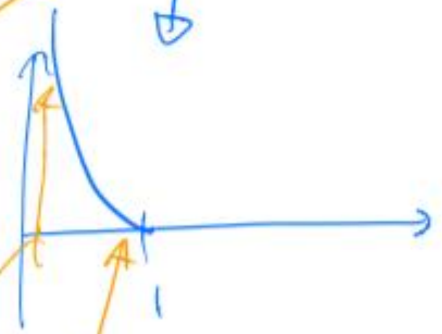
◆ Softmax Function



◆ Cost Function – Cross Entropy

$S(Y)$ $\begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$ $L=Y$ $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

$$D(S, L) = -\sum_i L_i \log(S_i)$$



실제 정답인 1로 쉽게로 내게 다음일

모델이 답을 설했으면 확률이 높은데에다 Cost ↓.

모델이 답을 틀렸으면 확률 낮은 데에다 Cost ↑.

* Logistic Regression의 Cost는 Label이 2개인 Cross Entropy.

○ 전체 Cost function.

$$\mathcal{L} = \underbrace{\frac{1}{N}\sum_i}_{\text{mean}} \underbrace{D(S(wx_i+b), L_i)}_{\text{Cross Entropy.}}$$

Training Set Index

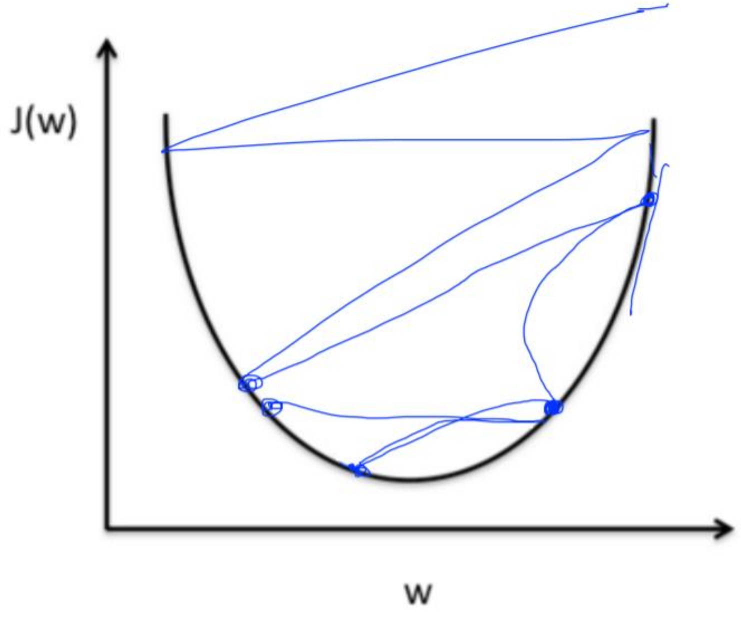
→ G.D.A.

Application & Tips:

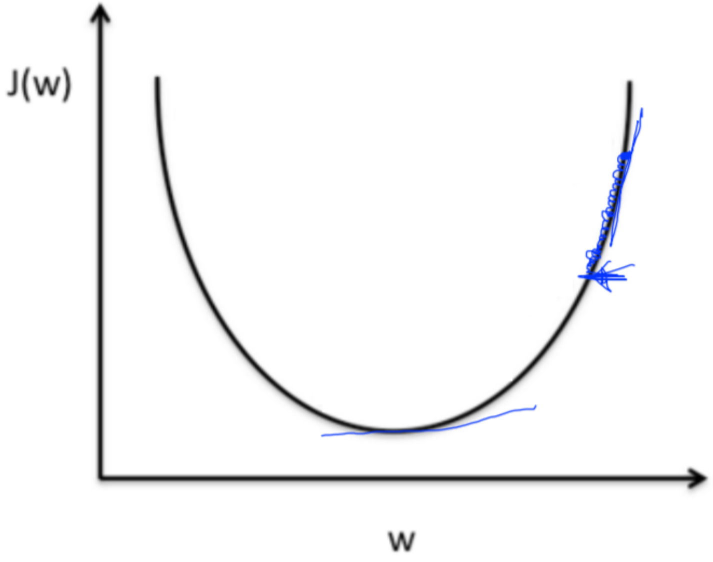
- Learning rate, data preprocessing, overfitting

◆ Learning Rate

Large learning rate: overshooting



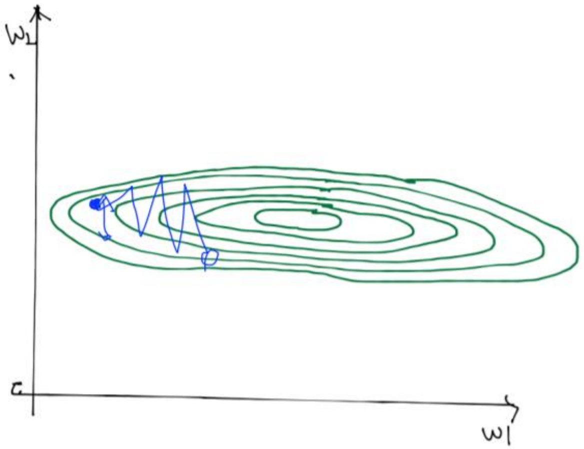
Small learning rate:
takes too long, stops at local minimum



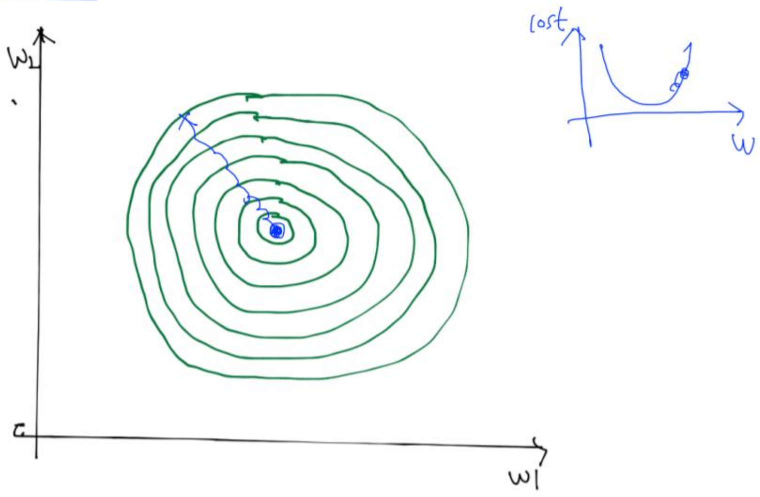
data preprocessing

Data (X) preprocessing for gradient descent

x1	x2	y
1	9000	A
2	-5000	A
4	-2000	B
6	8000	B
9	9000	C



Data (X) preprocessing for gradient descent



◆ Overfitting

Solutions for overfitting

- More training data!
- Reduce the number of features
- **Regularization**

☆

◆ Overfitting

- Let's not have too big numbers in the weight

LOSS

regularization strength

$$\mathcal{L} = \frac{1}{N}\sum_{i} D(S(wx_i+b), L_i) + \boxed{\lambda \sum W^2}$$

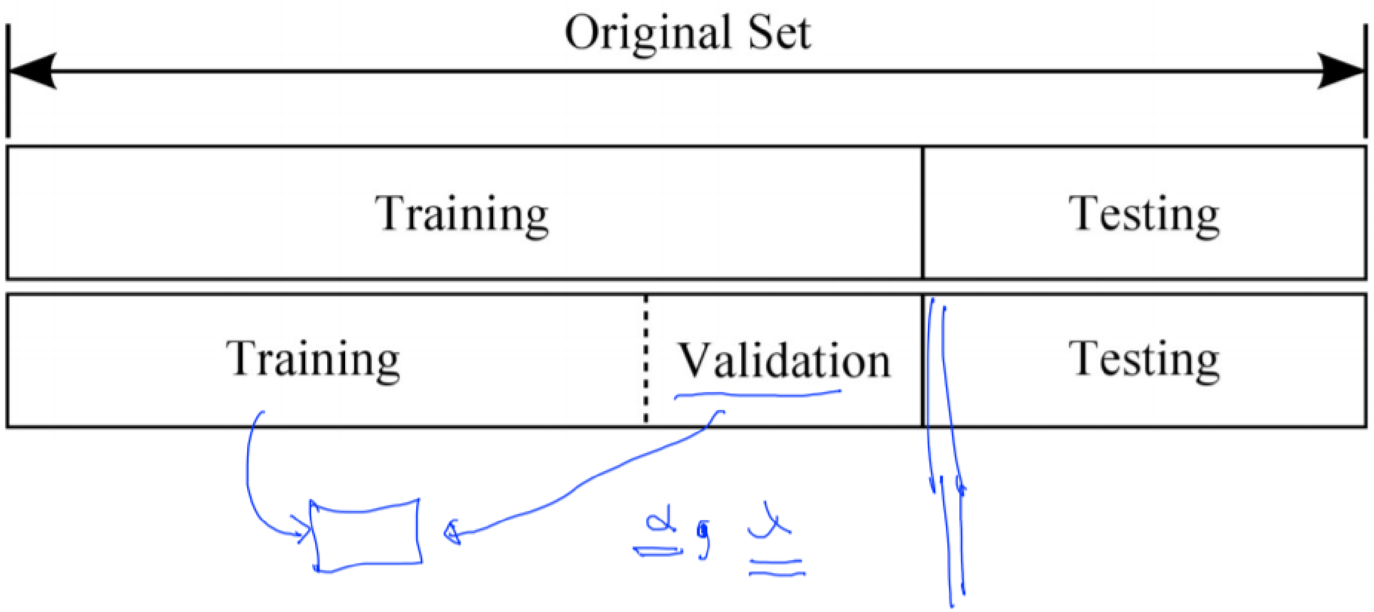
TRAINING SET

O X

0.001

◆ Training / Test

Training, validation and test sets



Lab

◆ 전체 프로세스

데이터 세팅

Hypothesis

↓

Cost Define

↓

Minimize Cost

① Build Graph

⇓

Session Open

↓

계산 실행

② Feed Data & Run Graph

⇓

결과출력

③ Update & Return

```python
xy = np.loadtxt('data-03-diabetes.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]
```

Data Load

① Build Graph

```python
# placeholders for a tensor that will be always fed.
X = tf.placeholder(tf.float32, shape=[None, 8])
Y = tf.placeholder(tf.float32, shape=[None, 1])

W = tf.Variable(tf.random_normal([8, 1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')

# Hypothesis using sigmoid: tf.div(1., 1. + tf.exp(tf.matmul(X, W)))
hypothesis = tf.sigmoid(tf.matmul(X, W) + b)
```

Hypothesis

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

Cost define

```python
train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(cost)
```

Min. Cost

```python
# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))
```

결과 출력, 정확도 계산

```python
# Launch graph
with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    feed = {X: x_data, Y: y_data}
    for step in range(10001):
        sess.run(train, feed_dict=feed)
        if step % 200 == 0:
            print(step, sess.run(cost, feed_dict=feed))
```

② Feed Data & Run Graph

```python
    # Accuracy report
    h, c, a = sess.run([hypothesis, predicted, accuracy], feed_dict=feed)
    print("\nHypothesis: ", h, "\nCorrect (Y): ", c, "\nAccuracy: ", a)
```

③ 결과출력.

- 예측 값 계산, 정확도 계산할 때 tf.cast
- 한 data type에서 다른 data type으로 바꿔주는 함수

$$W := W - \alpha \frac{\partial}{\partial W}$$

datatype 바꾸기

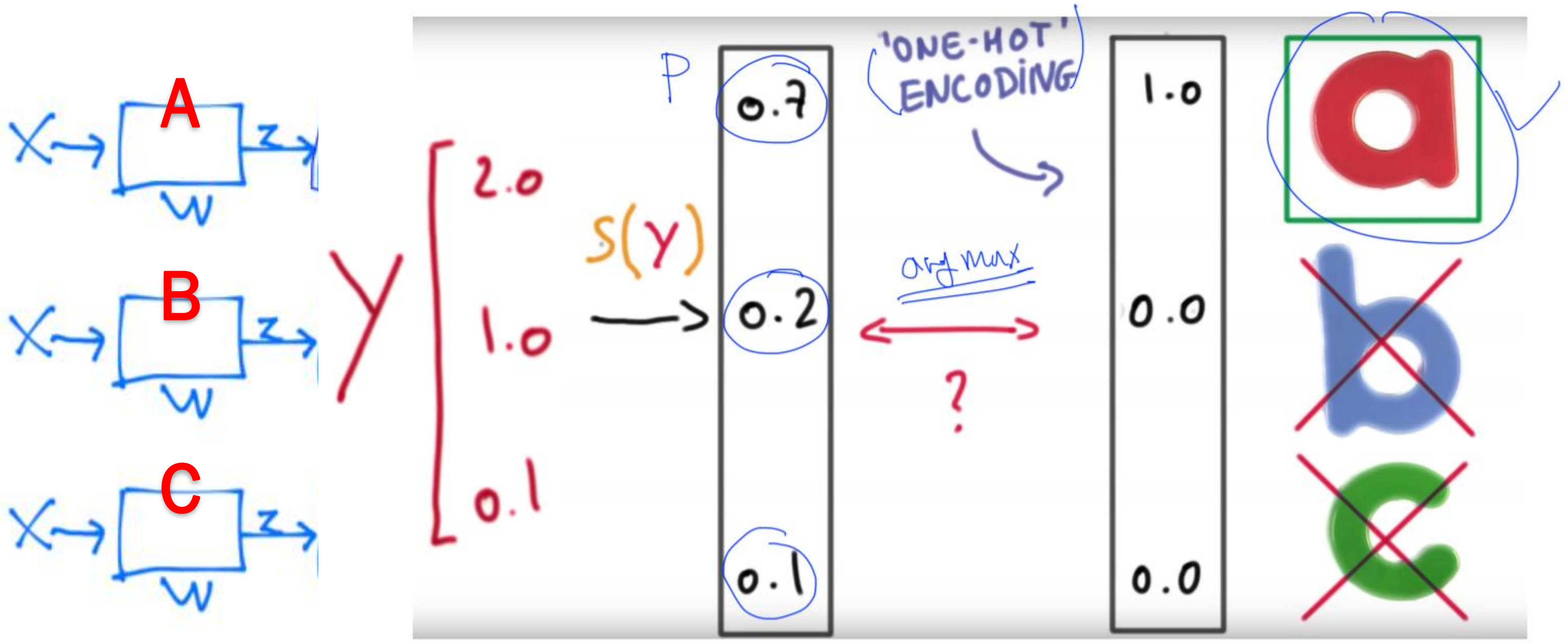
Boolean

e.g. 0 or 1 in float32

```
# Accuracy computation
# True if hypothesis>0.5 else False
predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, Y), dtype=tf.float32))
```

mean

◆ Softmax Classification (Multinomial Classification)



◆ Softmax, Classification

```
hypothesis = tf.nn.softmax(tf.matmul(X,W)+b)
```

Cost function: cross entropy

LOSS

$$\mathcal{L} = \frac{1}{N}\sum_i D(S(WX_i+b), L_i)$$

TRAINING SET

STEP

$$-\alpha \Delta\mathcal{L}(w_1, w_2)$$

DERIVATIVE

```
# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize(cost)
```

softmax_cross_entropy_with_logits

```
logits = tf.matmul(X, W) + b
hypothesis = tf.nn.softmax(logits)
```

1

```
# Cross entropy cost/loss
cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hypothesis), axis=1))
```

2

```
# 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)
```

Test & one-hot encoding

```
hypothesis = tf.nn.softmax(tf.matmul(X,W)+b)
```

```
all = sess.run(hypothesis, feed_dict={X: [[1, 11, 7, 9],
                                          [1, 3, 4, 3],
                                          [1, 1, 0, 1]]})

print(all, sess.run(tf.arg_max(all, 1)))
```

최댓값 찾기

axis

```
[[ 1.38904958e-03  9.98601854e-01  9.06129117e-06]
[ 9.31192040e-01  6.29020557e-02  5.90589503e-03]
[ 1.27327668e-08  3.34112905e-04  9.99665856e-01]]

[1 0 2]
```

tf.one_hot and reshape

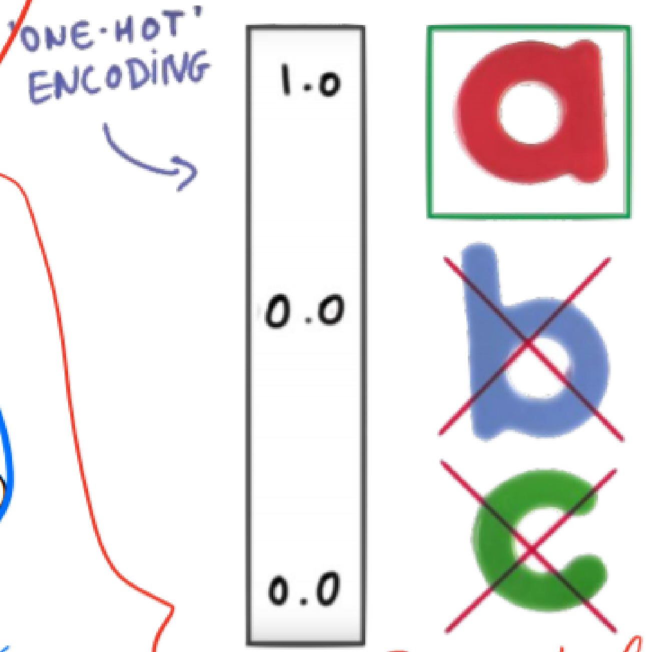
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0
1	0	0	1	0	0	0	1	1	1	0	0	4	1	1	1	0
0	0	1	0	0	1	0	1	1	0	0	1	0	1	1	0	3
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
1	0	0	1	0	0	0	1	1	1	0	0	4	0	1	0	0
1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	0
0	1	1	0	1	0	0	0	1	1	0	0	2	1	1	0	1
0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	3
0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	4	0	0	0	6
0	0	1	0	0	1	1	0	0	0	0	0	6	0	0	0	6
0	1	1	0	1	0	1	0	1	1	0	0	2	1	0	0	1
1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	0

```
Y = tf.placeholder(tf.int32, [None, 1])  # 0 ~ 6, shape=(?, 1)
Y_one_hot = tf.one_hot(Y, nb_classes)  # one hot shape=(?, 1, 7)
Y_one_hot = tf.reshape(Y_one_hot, [-1, nb_classes]) # shape=(?, 7)
```

(0010000)

```
# Predicting animal type based on various features
xy = np.loadtxt('data-04-zoo.csv', delimiter=',', dtype=np.float32)
x_data = xy[:, 0:-1]
y_data = xy[:, [-1]]

nb_classes = 7  # 0 ~ 6

X = tf.placeholder(tf.float32, [None, 16])
Y = tf.placeholder(tf.int32, [None, 1])  # 0 ~ 6

Y_one_hot = tf.one_hot(Y, nb_classes)  # one hot
Y_one_hot = tf.reshape(Y_one_hot, [-1, nb_classes])

W = tf.Variable(tf.random_normal([16, nb_classes]), name='weight')
b = tf.Variable(tf.random_normal([nb_classes]), name='bias')

# 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)
```

Load Data



Hypothesis

cost

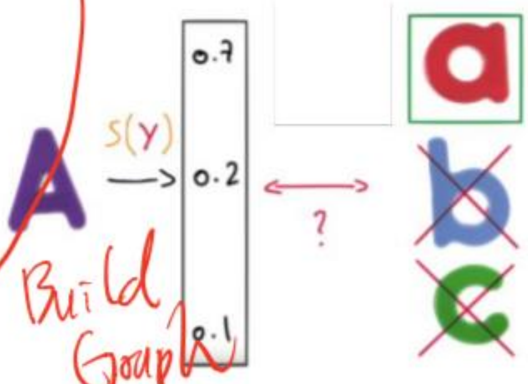
Min.

① Build Graph

```python
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))
```

결과계산

Build Graph




```python
# Launch graph
with tf.Session() as sess:
   sess.run(tf.global_variables_initializer())

   for step in range(2000):
       sess.run(optimizer, feed_dict={X: x_data, Y: y_data})
       if step % 100 == 0:
           loss, acc = sess.run([cost, accuracy], feed_dict={
                                X: x_data, Y: y_data})
           print("Step: {:5}\tLoss: {:.3f}\tAcc: {:.2%}".format(
               step, loss, acc))
```

② Feed Data & Run Graph

```python
   # Let's see if we can predict
   pred = sess.run(prediction, feed_dict={X: x_data})
   # y_data: (N,1) = flatten => (N, ) matches pred.shape
   for p, y in zip(pred, y_data.flatten()):
       print("[{}] Prediction: {} True Y: {}".format(p == int(y), p, int(y)))
```

차원축소

③ 결과출력

◆ Application & Tips:

- Learning rate, data preprocessing, overfitting

Normalized inputs (min-max scale)

```
xy = np.array([[828.659973, 833.450012, 908100, 828.349976, 831.659973],
               [823.02002, 828.070007, 1828100, 821.655029, 828.070007],
               [819.929993, 824.400024, 1438100, 818.97998, 824.159973],
               [816, 820.958984, 1008100, 815.48999, 819.23999],
               [819.359985, 823, 1188100, 818.469971, 818.97998],
               [819, 823, 1198100, 816, 820.450012],
               [811.700012, 815.25, 1098100, 809.780029, 813.669983],
               [809.51001, 816.659973, 1398100, 804.539978, 809.559998]])
```

```
xy = MinMaxScaler(xy)
print(xy)
```

w_1

Training epoch/batch

In the neural network terminology:

- one **epoch** = one forward pass and one backward pass of *all* the training examples
- **batch size** = the number of training examples in one forward/backward pass. The higher the batch size, the more memory space you'll need.
- number of **iterations** = number of passes, each pass using [batch size] number of examples. To be clear, one pass = one forward pass + one backward pass (we do not count the forward pass and backward pass as two different passes).

Example: if you have *1000 training examples*, and your *batch size is 500*, then it will take 2 iterations to complete 1 epoch.

```python
# parameters
training_epochs = 15
batch_size = 100

with tf.Session() as sess:
    # Initialize TensorFlow variables
    sess.run(tf.global_variables_initializer())
    # Training cycle
    for epoch in range(training_epochs):
        avg_cost = 0
        total_batch = int(mnist.train.num_examples / batch_size)

        for i in range(total_batch):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            c, _ = sess.run([cost, optimizer], feed_dict={X: batch_xs, Y: batch_ys})
            avg_cost += c / total_batch

        print('Epoch:', '%04d' % (epoch + 1), 'cost =', '{:.9f}'.format(avg_cost))
```

유상현 yush0123@snu.ac.kr

END OF PRESENTATION