ML lec 01

✓ 용어 정리

- Supervised Learning
  - label들이 정해져 있는 data를 가지고 학습을 하는 것
  - 예를 들어, 고양이인지 개인지 구분시키거나 Spam 분류
- Unsupervised Learning
  - Data를 보고 스스로 학습
- Regression
  - 시간에 따른 시험 점수 예측
- Binary Classification
  - 시간에 따른 Pass/Fail 예측
- Multi-Label Classifiation
  - 시간에 따른 성적(A, B, C, E, F) 예측

ML lab 01

✓ TensorFlow란?

- Data Flow Graphs를 이용하여 수적 계산을 하는 라이브러리
- Python을 사용

✓ TensorFlow를 통해 문자 출력

```
1  hello = tf.constant("Hello, TensorFlow!")
2  sess = tf.Session()
3  print(sess.run(hello))
>>> b'Hello, TensorFlow!'
```

- session없이 바로 print를 사용하면 노드값이 나옴

✓ 원하는 값을 전달하여 출력

```
1  a = tf.placeholder(tf.float32)
2  b = tf.placeholder(tf.float32)
3  add = a + b
4
5  print(sess.run(add, feed_dict={a: 3, b: 4.5}))
6  print(sess.run(add, feed_dict={a: [1, 3], b: [2, 4]}))
>>> 7.5
    [ 3.  7.]
```

- Placeholder로 값을 넘겨줘서 대입할 수 있음

✓ TensorFlow는 그래프를 먼저 설계하고 값을 대입해야 한다

ML lec 02

✓ Learning을 하려면 **Hypothesis(가설)**를 설정해 줘야 함

- **Linear Regression**(Hypothesis) : $H(x) = Wx + b$
- **Cost**(loss) **Function** : $\{H(x) - y\}^2$, 정확히는 아래와 같음

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

✓ 제곱을 하는 이유는 차이가 작을수록 제곱한 값이 작기 때문

예측값과 실제값의 차이를 제곱하는 것

ML lab 02

✓ $\boldsymbol{H(x) = Wx + b}$

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

hypothesis = x_train * W + b
```

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

```
cost = tf.reduce_mean(tf.square(hypothesis - y_train))
```

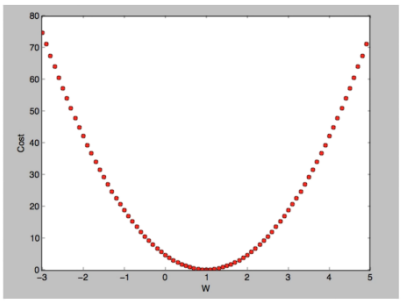
- `tf.reduce_mean(a)`는 평균을 계산하는 함수
- 가설이 맞으면 학습이 진행될수록 cost값은 0에 수렴하게 됨
- Placeholder를 이용하여 X, Y data를 던져줄 수 있음

ML lec 03

✓ Minimizing Cost

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

- **Gradient Descent** 알고리즘을 이용
  - Cost(W, b)를 조금씩 줄이며 지점을 찾음
  - $W := W - \alpha\frac{1}{m}\sum_{i=1}^{m}(Wx^{(i)} - y^{(i)})x^{(i)}$
  - ↑Cost Function의 미분
  - 어디서 시작하든 최저점을 찾을 수 있다는 장점



↑Cost Function의 그래프 모양

ML lab 03

✓ matplotlib.pyplot을 이용하여 그래프를 그려볼 수 있음

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

```
learning_rate = 0.1 # alpha
gradient = tf.reduc_mean((W * X - Y) * X)
descent = W - learning_rate * gradient
update = W.assign(descent) # update를 sess로 실행시키면 알고리즘이 작동됨
```

- `tf.train.GradientDescentOptimizer(alpha)`로 optimize 가능

ML lec 04

✓ input이 많아져도 $H(x_1, x_2, x_3) = w_1x_1 + w_2x_2 + w_3x_3 + b$로 구현가능

- **Matrix**로 더 간단히 구현 가능

$$\rightarrow (x_1 \quad x_2 \quad x_3) \cdot \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix} = (x_1w_1 + x_2w_2 + x_3w_3) \rightarrow H(x) = \underline{\boldsymbol{XW}}$$

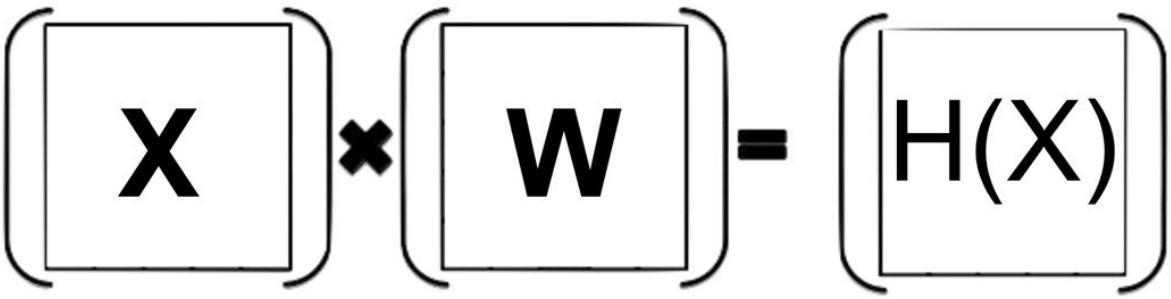
- 더 복잡한 것도 구현 가능

$$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$$

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

[5, 3] **[3, 1]** **[5, 1]**

- Matrix를 이용하여 W도 쉽게 계산 가능



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

ML lab 04-1

✓ **Matrix**로 input 입력하는 법

```
x_data = [[73., 80., 75.], [93., 88., 93.],
          [89., 91., 90.], [96., 98., 100.], [73., 66., 70.]]
y_data = [[152.], [185.], [180.], [196.], [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
hypothesis = tf.matmul(X, W) + b
```

ML lab 04-2

✓ **data 파일 불러오는 법**

```
1  xy = np.loadtxt('file_name.csv', delimiter='data구분기호', dtype=np.float32
2  x_data = xy[:, 0:-1]
3  y_data = xy[:, [-1]]
```

✓ **Queue Runners로 광범위한 데이터를 불러올 수 있음**

- tf.train.batch()로 불러온 data를 가져올 수 있음
  - tf.train.batch([xy[0:-1], xy[-1]], batch_size=10)
- shuffle_batch로 순서를 섞을 수 있음

ML lec 5-1

✓ **Logistic**(regression) **Classification**

- Binary Classification → Spam Detection, Facebook feed
  - Linear Regression으로 구현하기 힘듦
- 주어진 값을 Binary로 바꿔주는 식이 없을까?

✓ **Sigmoid Function**(hypothesis)

- $0 < x < 1$을 만족하기 위해 만들어졌다
- $z = WX, \quad H(x) = g(z)$로 설정
- $H(X) = \frac{1}{1 + e^{-W^T X}}$
  - Linear Regression($WX+b$)을 Sigmoid 함수의 z에 대입한 가설

ML lec 5-2

✓ **Sigmoid Function으로 그래프를 그리면 울퉁불퉁해짐**

- 그렇기에 Local Minimum이 생김

✓ **log함수로 잡아줄 수 있다**

- 가설과 Cost 함수가 일치하면 Cost값은 0이 되고,

  틀리게 되면 Cost값은 ∞으로 발산

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

ML lab 05

✓ Sigmoid Function을 구현하는 법

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

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

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

- 이것 또한 마찬가지로 xy = np.loadtxt("file_name.csv", …)을 통해 data를 불러올 수 있다.
- 구현할 때 data에 따라 변하는 shape에 주의할 것

```
X = tf.placeholder(tf.float32, shape=[None, 8])
…
W = tf.Variable(tf.random_normal([8, 1]), name='weight')
```