02. ML with tensorflow ver1 #src: 9_ML_DL/05_tensorflow_ver1_ML 참고

1) tensorflow ver1을 이용한 linear regression을 구현

(1) 독립변수가 1개; predict를 하기 위한 placeholder 노드

(2) scale이 다른 데이터들의 linear regression; scale이 다른 x,y값

(3) 독립변수 x가 여러 개인 linear regression; Multi-variable Linear Regression

(4) **scale 맞추는 방법**; normalization, standardization

```
#training data set 생성 (data load -> 결측치 처리 -> 독립변수, 종속변수 분리), scale조정
#scale 조정 ① sklearn.preprocessing.MinMaxScaler ② sklearn.preprocessing.StandardScaler
#1. placeholder
X = tf.placeholder(shape=[None, 3], dtype=tf.float32)
Y = tf.placeholder(shape=[None, 1], dtype=tf.float32)
#2. Weight & bias 설정(처음에는 랜덤값 셋팅 후 학습과정에서 변경)
W = tf.Variable(tf.random_normal([3,1]), name='weight')
b = tf.Variable(tf.random_normal([1]), name='bias')
#3. Hypothesis (H = X @ W + b)
H = tf.matmul(X, W) + b
#4. cost function(최소 제곱법); 곡선 위의 미분값이 줄어드는 방향으로 학습
cost = tf.reduce_mean(tf.square(H-Y))
#5. train
train = tf.train.GradientDescentOptimizer(learning_rate=1e-5).minimize(cost)
#6. Session; runner 생성 & Variable 초기화
sess = tf.Session()
sess.run(tf.global_variables_initializer())
#7. 학습
for step in range(1, 6001):
    _, cost_val, W_val, b_val = sess.run([train, cost, W, b], feed_dict={X:x_data, Y:y_data})
    if step%200==0:
        print("{}번째 cost:{}, W:{}, b:{}".format(step, cost_val, W_val, b_val))
#8. prediction
input_data = np.array([[118.,  8.,  72.]])
scaled_input_data = scaler_x.transform(input_data)
scaler_y.inverse_transform(sess.run(H, feed_dict={X:scaled_input_data}))
```

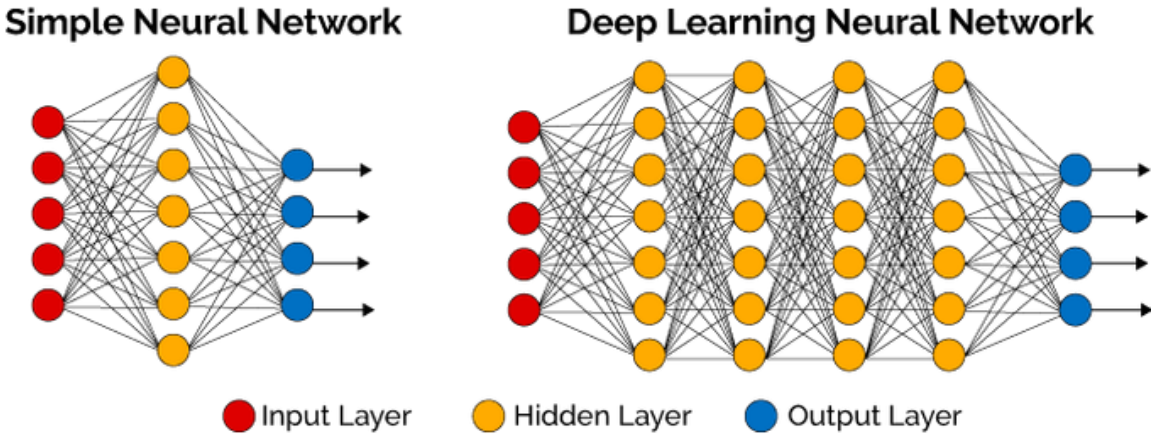
2) Logistic Regression

(1) Binary Classification(2개 그룹) #ex) pass/fail, 스팸메일 예측 등

(2) Multinomial Classfication(3개 이상 그룹) #ex) 등급 분류

3) 머신러닝으로는 예측이 힘든 경우 Neural Network 학습 필요

(1) 은닉층(hidden layer) 이용해서 딥러닝 Neural Network 가능




```
#2. Weight & bias 설정 부분에 Layer 추가
W1 = tf.Variable(tf.random_normal([2,10]), name='weight1')
b1 = tf.Variable(tf.random_normal([10]), name='bias1')
layer1 = tf.nn.relu(tf.matmul(X,W1)+b1) #Neural Network에서 ReLU(Rectified Linear Unit 사용

W2 = tf.Variable(tf.random_normal([10,10]), name='weight2')
B2 = tf.Variable(tf.random_normal([10]), name='bias2')
layer2 = tf.nn.relu(tf.matmul(X, W2)+b2)

W3 = tf.Variable(tf.random_normal([10,1]), name='weight3')
B3 = tf.Variable(tf.random_normal([1]), name='bias3')

#3. Hypothesis (H = X @ W + b)
logits = tf.matmul(layer2, W3) + b3
H = tf.sigmoid(logits)
```

※ 과적합(Overfitting); Training data set(훈련 데이터 셋)에는 적합한 모델이지만 실제 데이터에는 적용이 잘 안되는 경우 ⇒ 많은 training data 이용, feature 개수 줄임, Regularization 등으로 해결