```
# Lab 9 XOR
import tensorflow as tf
import numpy as np

x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)
```

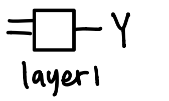
0과 1만 이용하기 때문에 logistic regression을 사용함

```
tf.model = tf.keras.Sequential()
tf.model.add(tf.keras.layers.Dense(units=1, input_dim=2, activation='sigmoid'))
tf.model.compile(loss='binary_crossentropy', optimizer=tf.optimizers.SGD(lr=0.01),
metrics=['accuracy'])
tf.model.summary()

history = tf.model.fit(x_data, y_data, epochs=1000)

predictions = tf.model.predict(x_data)
print('Prediction: \n', predictions)

score = tf.model.evaluate(x_data, y_data)
print('Accuracy: ', score[1])
```

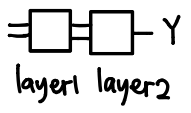
→ 하나의 네트워크를 이용함

↳ Accuracy : 0.5
정확도가 비교적 낮게 측정됨
(∵ 하나의 네트워크만 이용)

1개의 layer만 이용



2개의 layer 이용

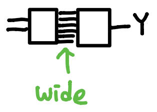


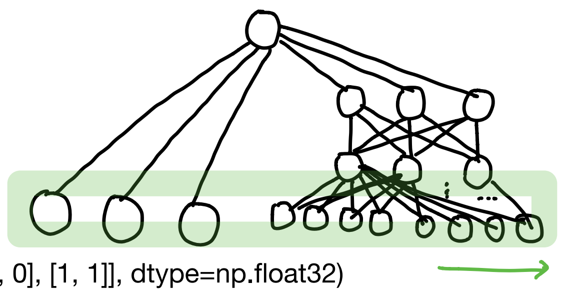
```
# Lab 9 XOR
import tensorflow as tf
import numpy as np

x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

tf.model = tf.keras.Sequential()
tf.model.add(tf.keras.layers.Dense(units=2, input_dim=2))
tf.model.add(tf.keras.layers.Activation('sigmoid'))
tf.model.add(tf.keras.layers.Dense(units=1, input_dim=2))
tf.model.add(tf.keras.layers.Activation('sigmoid'))
tf.model.compile(loss='binary_crossentropy', optimizer=tf.optimizers.SGD(lr=0.1),
metrics=['accuracy'])
tf.model.summary()

history = tf.model.fit(x_data, y_data, epochs=10000)

predictions = tf.model.predict(x_data)
print('Prediction: \n', predictions)

score = tf.model.evaluate(x_data, y_data)
print('Accuracy: ', score[1])
```

→ 두 개의 네트워크를 이용하여 Neural Net을 구성함

→ 정확도가 1.0으로 출력됨

Wide NN for XOR





```
# Lab 9 XOR
# 9-3 deep and wide
import tensorflow as tf
import numpy as np

x_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)
y_data = np.array([[0], [1], [1], [0]], dtype=np.float32)
```

Out 값의 개수 : 10개　　In 값의 개수 : 2개

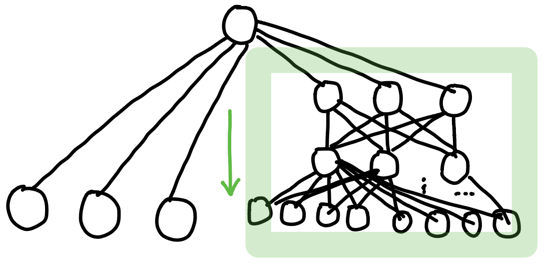
```
tf.model = tf.keras.Sequential()
tf.model.add(tf.keras.layers.Dense(units=10, input_dim=2, activation='sigmoid'))
tf.model.add(tf.keras.layers.Dense(units=10, activation='sigmoid'))
tf.model.add(tf.keras.layers.Dense(units=10, activation='sigmoid'))
tf.model.add(tf.keras.layers.Dense(units=10, activation='sigmoid'))
tf.model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
```

가로로 5개의 단계를 거치게 됩니다.

최종 out값 y

```
# SGD not working very well due to vanishing gradient problem, switched to Adam for now
# or you may use activation='relu', study chapter 10 to know more on vanishing gradient problem.
tf.model.compile(loss='binary_crossentropy', optimizer=tf.optimizers.Adam(lr=0.1),
metrics=['accuracy'])
tf.model.summary()

history = tf.model.fit(x_data, y_data, epochs=5000)

predictions = tf.model.predict(x_data)
print('Prediction: \n', predictions)

score = tf.model.evaluate(x_data, y_data)
print('Accuracy: ', score[1])
```

Deep NN for XOR



layer 0
layer 1
layer 2
⋮

layer를 여러 개 만들어줍니다.