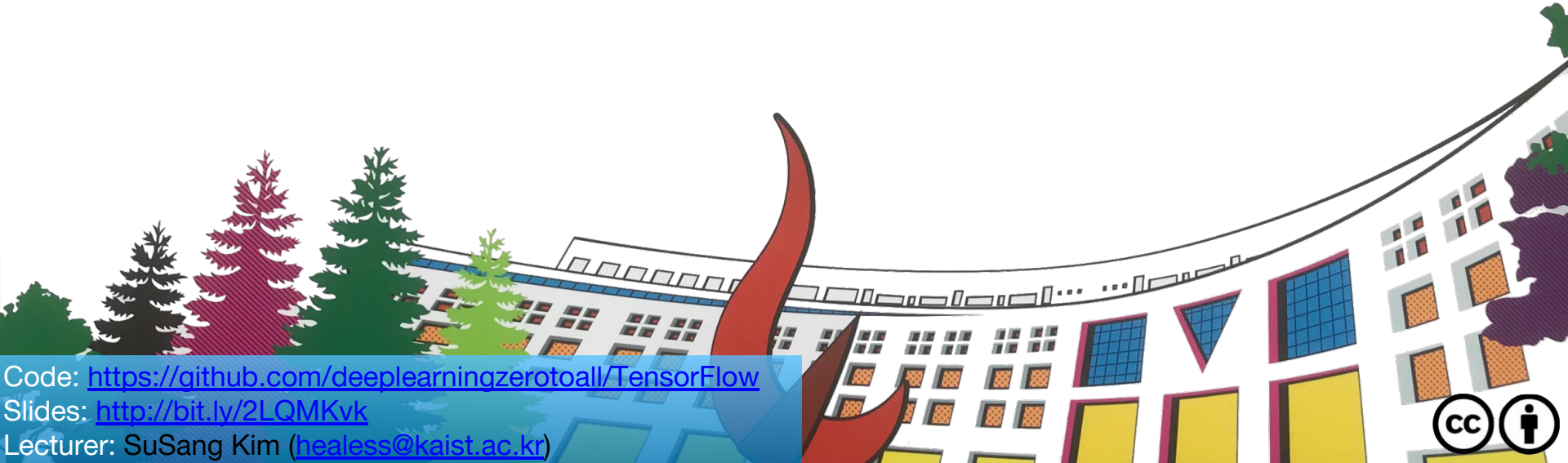
ML/DL for Everyone Season2

with TensorFlow

Lab 09-1 Neural Nets for XOR



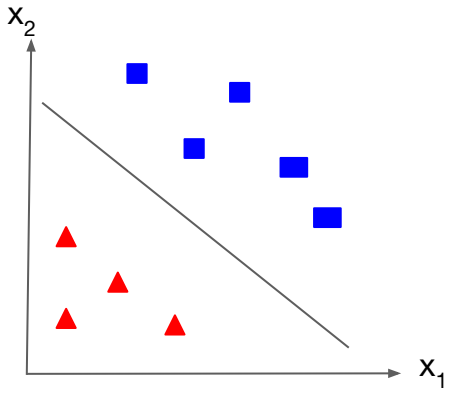
Code: https://github.com/deeplearningzerotoall/TensorFlow
Slides: http://bit.ly/2LQMKvk
Lecturer: SuSang Kim (healess@kaist.ac.kr)

Neural Nets for XOR

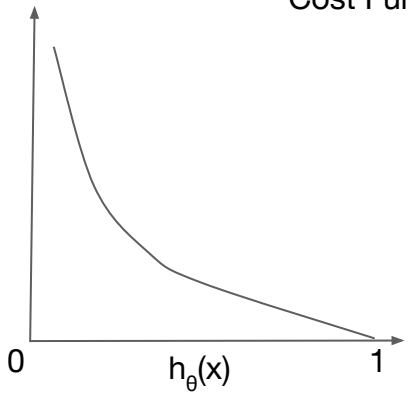
- XOR Problem
  - Logistic Regression (Recap)
  - Neural Network
  - Chart
  - Codes (Eager Execution)
  - Summary

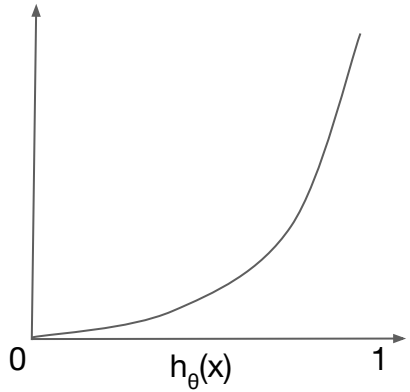
Logistic Regression

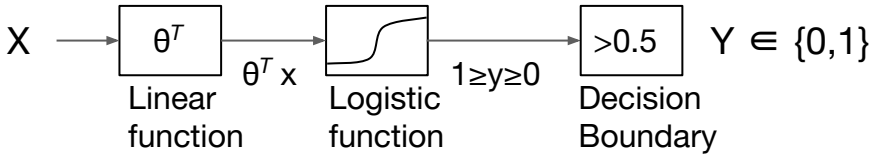
Recap



Cost Function





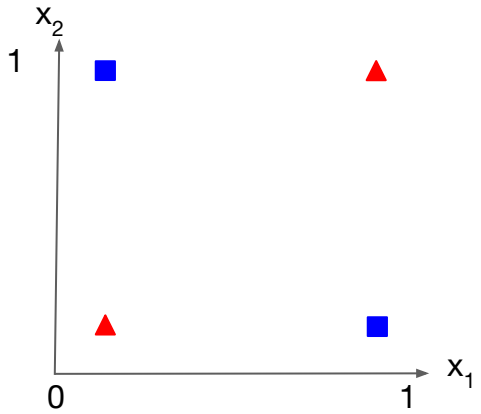


$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

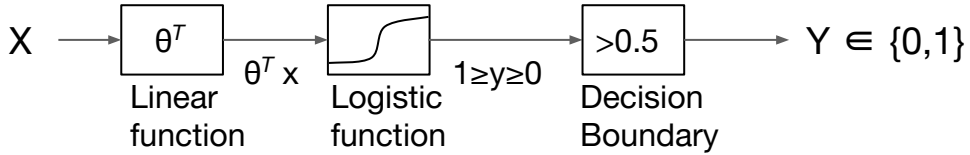
$\text{cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1-y) \log(1 - h_\theta(x))$

Logistic Regression

XOR

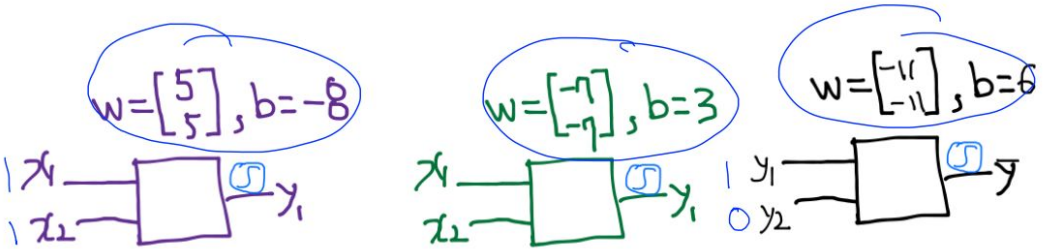


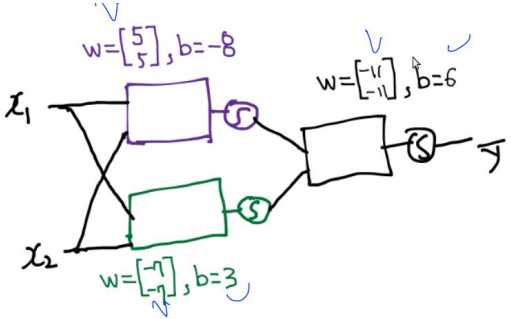
x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0



Data sets

Forward propagation





$[1\ 1]\begin{bmatrix}5\\5\end{bmatrix}-8=5+5-8=2$, Sigmoid(2) = 1

$[1\ 1]\begin{bmatrix}-7\\-7\end{bmatrix}+3=-7-7+3=-11$, Sigmoid(-11) = 0

$[1\ 0]\begin{bmatrix}-11\\-11\end{bmatrix}+6=-11+0+6=-5$

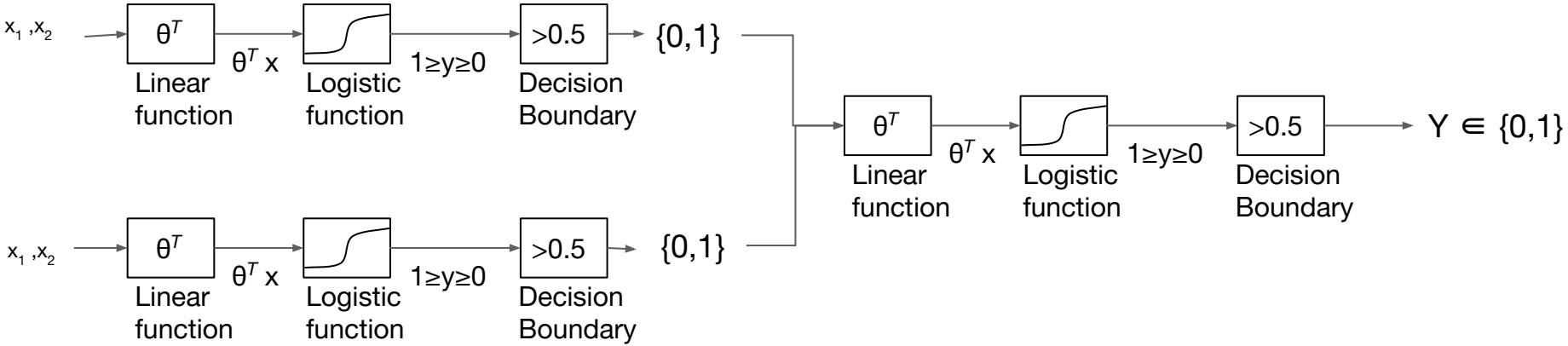
Sigmoid(-5) = 0

x_1	x_2	y_1	y_2	y	XOR
0	0	0	1	0	0
0	1	0	0	1	1
1	0	0	0	1	1
1	1	1	0	0	0

http://hunkim.github.io/ml/

Neural Net

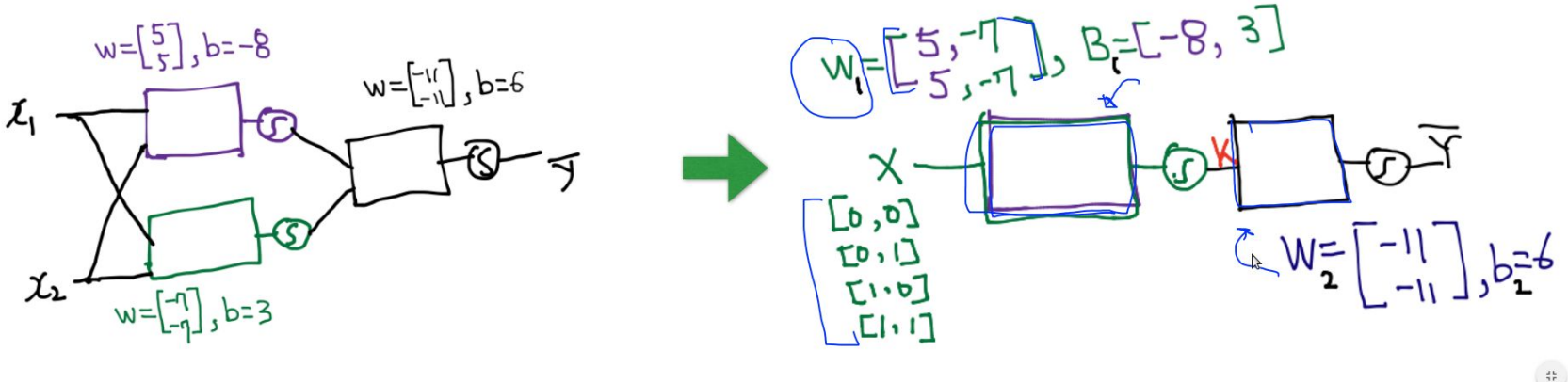
2 layer




```
[Tensorflow Code]
def neural_net(features):
    layer1 = tf.sigmoid(tf.matmul(features, W1) + b1) # W1=[2,1], b1=[1]
    layer2 = tf.sigmoid(tf.matmul(features, W2) + b2) # W2=[2,1], b2=[1]
    hypothesis = tf.sigmoid(tf.matmul(tf.concat([layer1, layer2],-1), W3) + b3) # W3=[2,1], b3=[1]
    return hypothesis
```

Neural Net

Vector

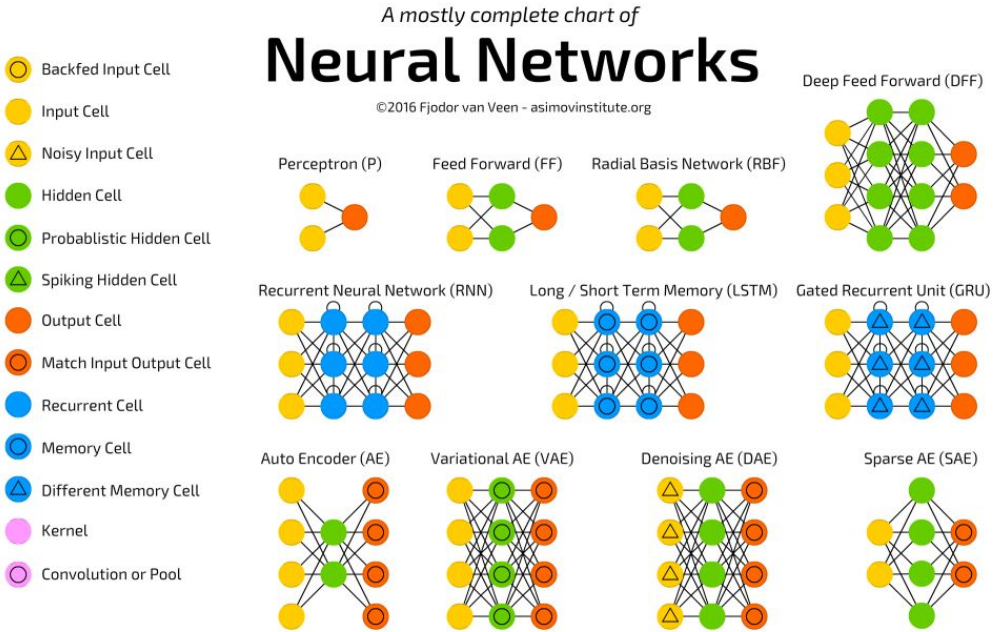


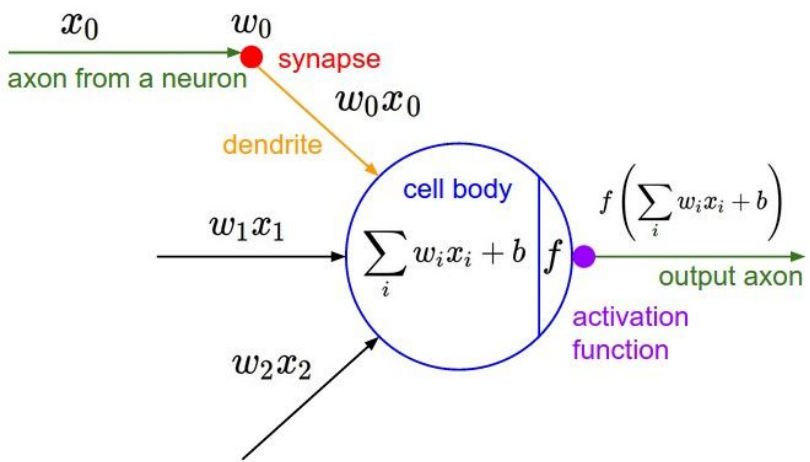

```
[Tensorflow Code]
def neural_net(features):
    layer = tf.sigmoid(tf.matmul(features, W1) + b1) # W1=[2,2], b1=[2]
    hypothesis = tf.sigmoid(tf.matmul(layer, W2) + b2) # W2=[2,1], b2=[1]
    return hypothesis
```

http://hunkim.github.io/ml/

Neural Net

Chart

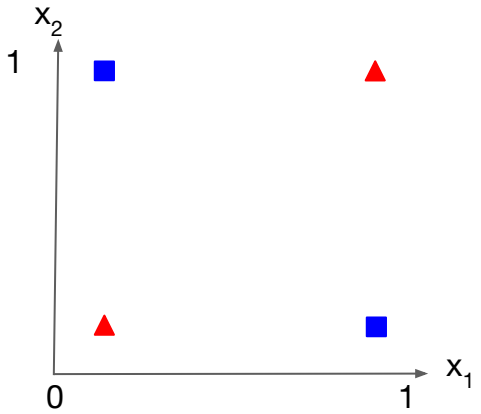


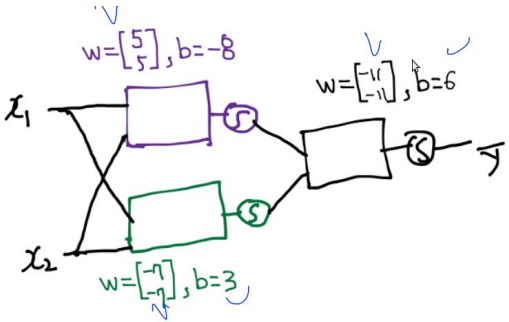


https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/neural_networks.html

Code(Eager)

```
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import tensorflow as tf
import tensorflow.contrib.eager as tfe

tf.enable_eager_execution()
tf.set_random_seed(777)  # for reproducibility

print(tf.__version__)

x_data = [[0, 0],
          [0, 1],
          [1, 0],
          [1, 1]]
y_data = [[0],
          [1],
          [1],
          [0]]

dataset = tf.data.Dataset.from_tensor_slices((x_data, y_data)).batch(len(x_data))

def preprocess_data(features, labels):
   features = tf.cast(features, tf.float32)
   labels = tf.cast(labels, tf.float32)
   return features, labels

W1 = tf.Variable(tf.random_normal([2, 1]), name='weight1')
b1 = tf.Variable(tf.random_normal([1]), name='bias1')

W2 = tf.Variable(tf.random_normal([2, 1]), name='weight2')
b2 = tf.Variable(tf.random_normal([1]), name='bias2')

W3 = tf.Variable(tf.random_normal([2, 1]), name='weight3')
b3 = tf.Variable(tf.random_normal([1]), name='bias3')
```

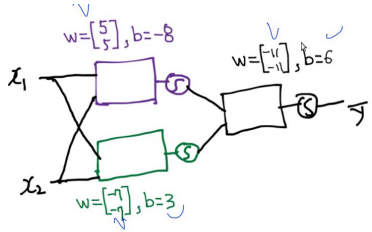




https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-09-2-XOR-NN-eager.ipynb

Code(Eager)

```python
def neural_net(features):
    layer1 = tf.sigmoid(tf.matmul(features, W1) + b1)
    layer2 = tf.sigmoid(tf.matmul(features, W2) + b2)
    layer3 = tf.concat([layer1, layer2],-1)
    layer3 = tf.reshape(layer3, shape = [-1,2])
    hypothesis = tf.sigmoid(tf.matmul(layer3, W3) + b3)
    return hypothesis

def loss_fn(hypothesis, features, labels):
    cost = -tf.reduce_mean(labels * tf.log(hypothesis) + (1 - labels) * tf.log(1 - hypothesis))
    return cost

optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.01)

def accuracy_fn(hypothesis, labels):
    predicted = tf.cast(hypothesis > 0.5, dtype=tf.float32)
    accuracy = tf.reduce_mean(tf.cast(tf.equal(predicted, labels), dtype=tf.float32))
    return accuracy

def grad(features, labels):
    with tf.GradientTape() as tape:
        loss_value = loss_fn(neural_net(features),features,labels)
    return tape.gradient(loss_value, [W1, W2, W3, b1, b2, b3])

EPOCHS = 50000

for step in range(EPOCHS):
    for features, labels  in tfe.Iterator(dataset):
        features, labels = preprocess_data(features, labels)
        grads = grad(neural_net(features), labels)
        optimizer.apply_gradients(grads_and_vars=zip(grads,[W1, W2, W3, b1, b2, b3]))
        if step % 5000 == 0:
            print("Iter: {}, Loss: {:.4f}".format(step, loss_fn(neural_net(features),labels)))
x_data, y_data = preprocess_data(x_data, y_data)
test_acc = accuracy_fn(neural_net(x_data),y_data)
print("Testset Accuracy: {:.4f}".format(test_acc))
```

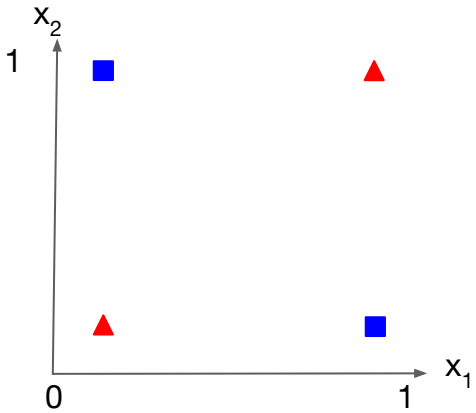


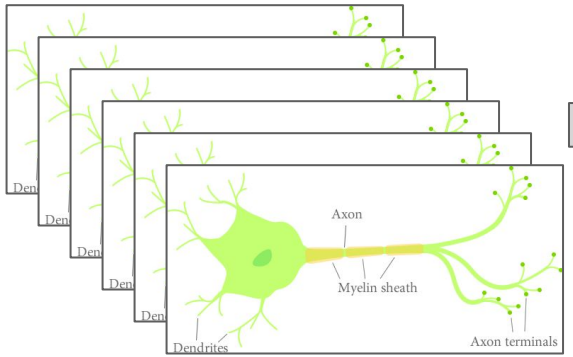

```
Iter: 0, Loss: 0.6931
Iter: 100, Loss: 0.6931
Iter: 200, Loss: 0.6931
Iter: 300, Loss: 0.6931
Iter: 400, Loss: 0.6931
Iter: 500, Loss: 0.6931
Iter: 600, Loss: 0.6931
Iter: 700, Loss: 0.6931
Iter: 800, Loss: 0.6931
Iter: 900, Loss: 0.6931
Iter: 1000, Loss: 0.6931
Testset Accuracy: 0.5000
```

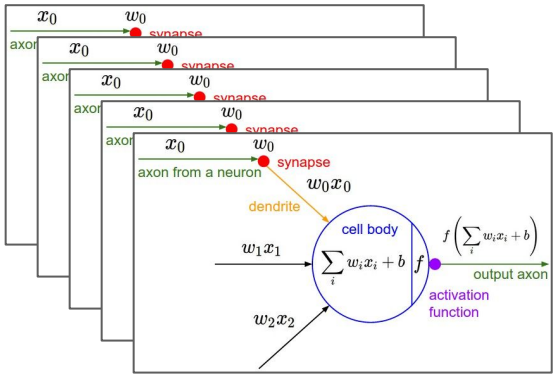
```
Iter: 0, Loss: 0.8487
Iter: 5000, Loss: 0.6847
Iter: 10000, Loss: 0.6610
Iter: 15000, Loss: 0.6154
Iter: 20000, Loss: 0.5722
Iter: 25000, Loss: 0.5433
Iter: 30000, Loss: 0.5211
Testset Accuracy: 1.0000
```

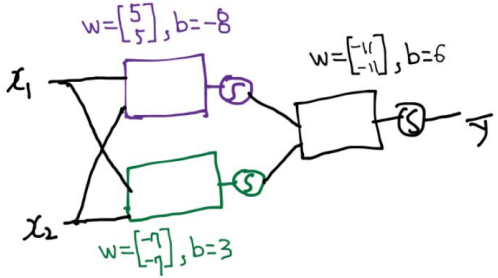
https://github.com/deeplearningzerotoall/TensorFlow/blob/master/lab-09-2-XOR-NN-eager.ipynb

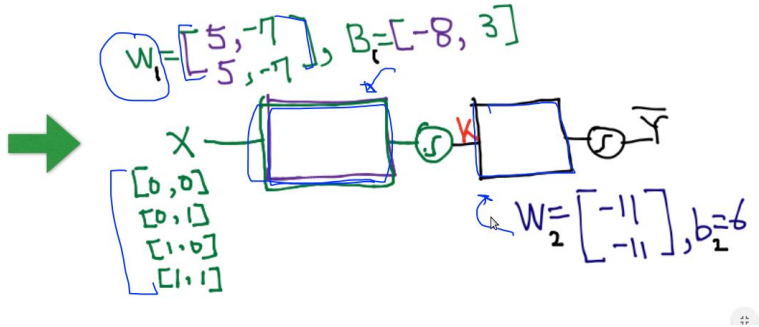
Summary











https://leonardoaraujosantos.gitbooks.io/artificial-inteligence/content/neural_networks.html

What's Next?

Sigmoid -> Relu