ML/DL for Everyone Season2

with TensorFlow

Lab06-1
Softmax Classifier

Code: https://github.com/deeplearningzerotoall/TensorFlow
Slides: http://bit.ly/2LQMKvk
Lecturer: sungjin7127@gmail.com

Lab6-1: Softmax Classifier

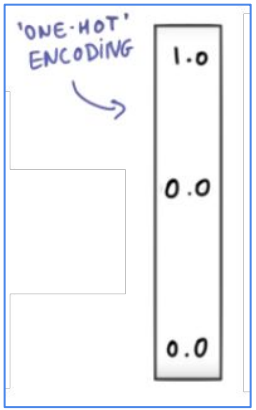
- Sample Dataset
- Softmax function
- Cost function
- Gradient function
- Train & Result
- What’s Next

Sample Dataset

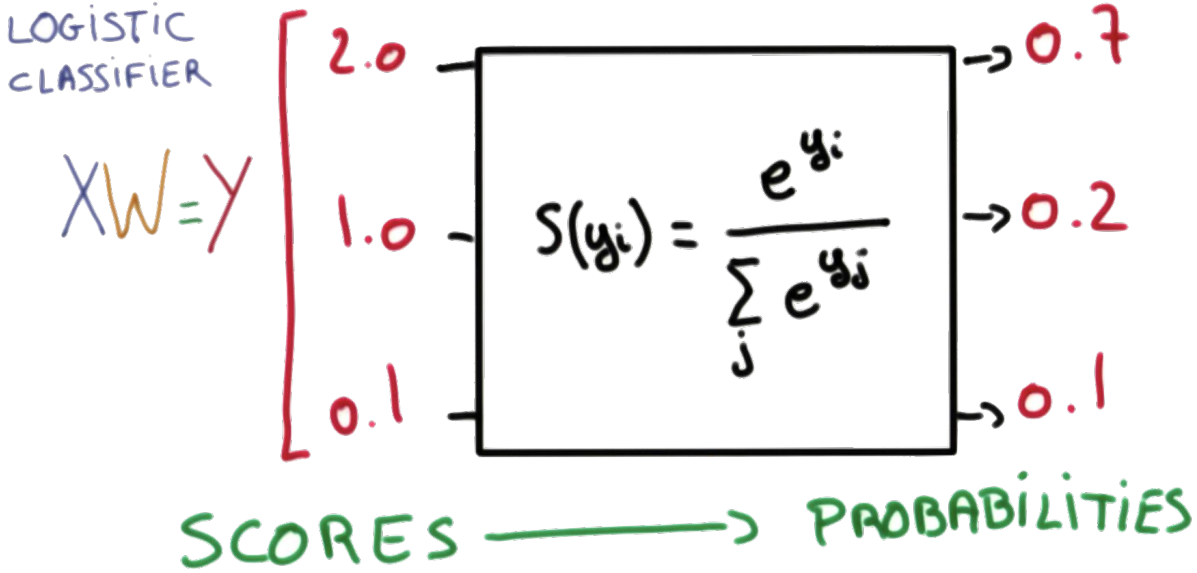
```
x_data = [[1, 2, 1, 1],
          [2, 1, 3, 2],
          [3, 1, 3, 4],
          [4, 1, 5, 5],
          [1, 7, 5, 5],
          [1, 2, 5, 6],
          [1, 6, 6, 6],
          [1, 7, 7, 7]]
y_data = [[0, 0, 1],
          [0, 0, 1],
          [0, 0, 1],
          [0, 1, 0],
          [0, 1, 0],
          [0, 1, 0],
          [1, 0, 0],
          [1, 0, 0]]
```

```
#convert into numpy and float format
x_data = np.asarray(x_data, dtype=np.float32)
y_data = np.asarray(y_data, dtype=np.float32)

nb_classes = 3 #num classes
```

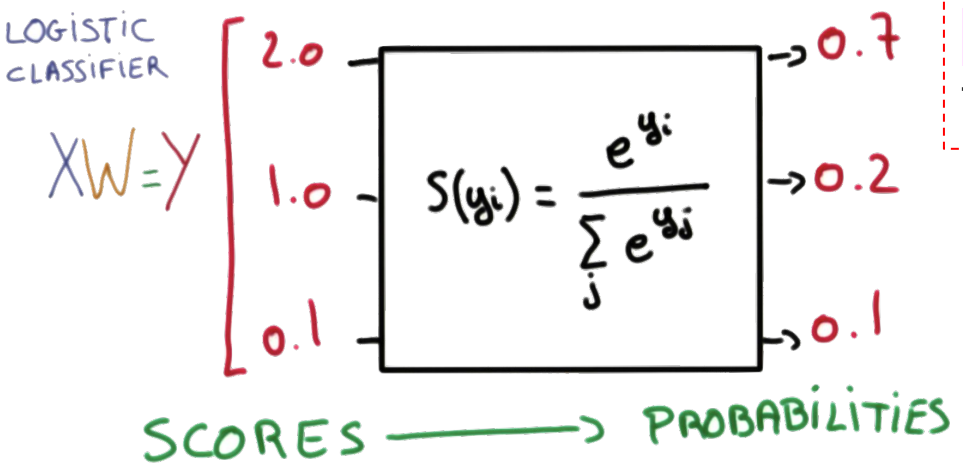


Softmax function



https://www.udacity.com/course/viewer#!/c-ud730/l-6370362152/m-6379811817

Softmax function



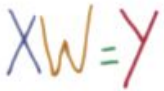
hypothesis =
tf.nn.**softmax**(tf.matmul(X,W)+b)

tf.matmul(X,W)+b

https://www.udacity.com/course/viewer#!/c-ud730/l-6370362152/m-6379811817

Softmax function

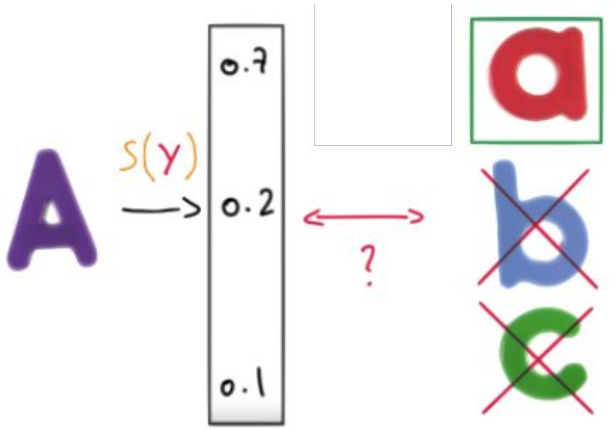
```
#Weight and bias setting
W = tfe.Variable(tf.random_normal([4, nb_classes]), name='weight')
b = tfe.Variable(tf.random_normal([nb_classes]), name='bias')
variables = [W, b]
```




```
<tf.Variable 'weight:0' shape=(4, 3) dtype=float32, numpy=
array([[ 2.0265348 , -0.19990598,  0.187595  ],
       [-1.8624718 ,  1.1830902 , -0.75108314],
       [ 0.7819291 ,  0.19707595,  0.6640797 ],
       [ 1.5643852 , -0.04990807, -0.38255563]], dtype=float32)>
<tf.Variable 'bias:0' shape=(3,) dtype=float32, numpy=array([-1.4564867 ,  0.53983474,
-1.1366715 ], dtype=float32)>
```

Softmax function

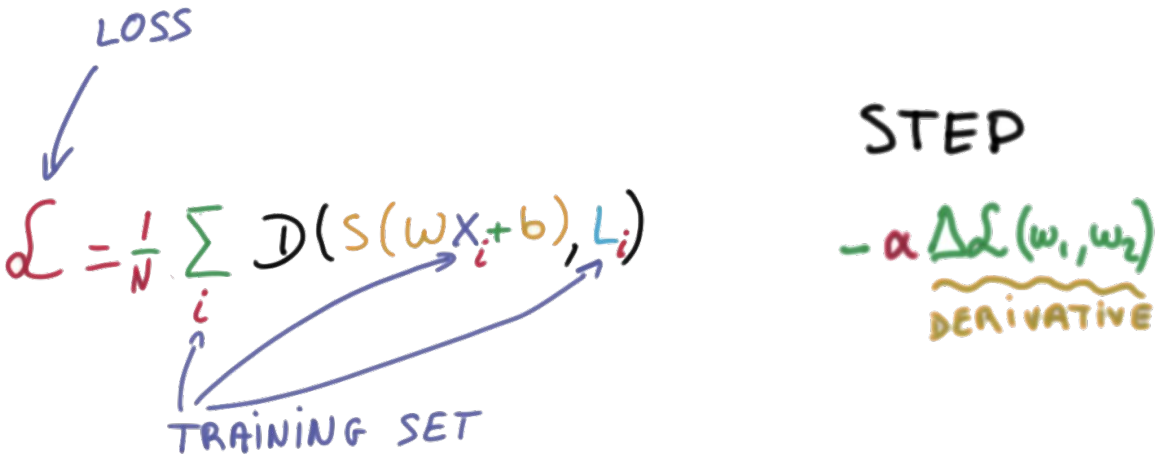
```
hypothesis = tf.nn.softmax(tf.matmul(x_data, W) + b)
```




```
# Softmax onehot test
sample_db = [[8,2,1,4]]
sample_db = np.asarray(sample_db, dtype=np.float32)

# Output
tf.Tensor([[0.9302204  0.06200533 0.00777428]], shape=(1, 3),
dtype=float32)
```

Cost function: cross entropy




```
# 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)
```

https://www.udacity.com/course/viewer#!/c-ud730/l-6370362152/m-6379811817

Cost Function

```
def cost_fn(X, Y):
    logits = hypothesis(X)
    cost = -tf.reduce_sum(Y * tf.log(logits), axis=1)
    cost_mean = tf.reduce_mean(cost)
    return cost_mean

print(cost_fn(x_data, y_data))
```

```
# Cost
tf.Tensor(
[3.4761162e+00 8.2235537e+00 6.6874886e+00 6.9770794e+00 6.4782157e+00
 3.7971997e+00 8.9059100e-03 6.9166054e-03], shape=(8,), dtype=float32)

# Cost mean
tf.Tensor(4.4569345, shape=(), dtype=float32)
```

Gradient Function

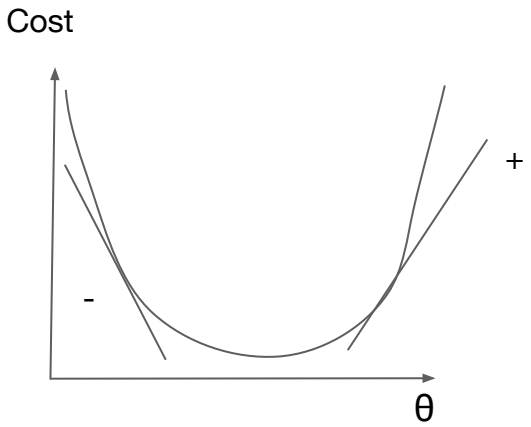
```
def grad_fn(X, Y):
    with tf.GradientTape() as tape:
        cost = cost_fn(X, Y)
        grads = tape.gradient(cost, variables)
        return grads

print(grad_fn(x_data, y_data))
```

```
[<tf.Tensor: id=533, shape=(4, 3), dtype=float32, numpy=
 array([[ 0.06991257, -0.00614021, -0.06377237],
        [ 0.02443989, -0.00085829, -0.02358155],
        [ 0.06863485, -0.01986565, -0.04876918],
        [ 0.08595259, -0.01490254, -0.07105   ]], dtype=float32)>,
 <tf.Tensor: id=531, shape=(3,), dtype=float32, numpy=array([ 0.02441996,  0.00028941,
-0.02470937], dtype=float32)>]
```

Train

```
def fit(X, Y, epochs=2000, verbose=100):
    optimizer =  tf.train.GradientDescentOptimizer(learning_rate=0.1)
    for i in range(epochs):
        grads = grad_fn(X, Y)
        optimizer.apply_gradients(zip(grads, variables))
        if (i==0) | ((i+1)%verbose==0):
        print('Loss at epoch %d: %f' %(i+1, cost_fn(X,Y).numpy()))
```



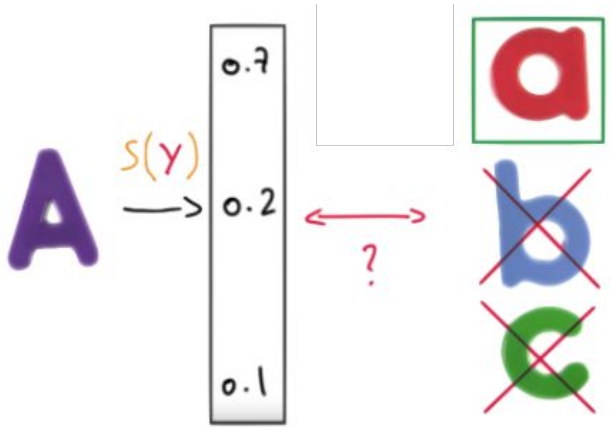

```
Loss at epoch 100: 0.153950
Loss at epoch 200: 0.148822
...
Loss at epoch 1900: 0.094386
Loss at epoch 2000: 0.092371
```

* lab-05-1-logistic_regression

Prediction

```
a = hypothesis(x_data)

print(a)
print(tf.argmax(a, 1))
print(tf.argmax(y_data, 1)) # matches with y_data
```




```
tf.Tensor(
[[1.5791884e-09 1.2361944e-06 9.9999881e-01]
 [3.4672192e-03 2.1560978e-02 9.7497183e-01]
 [2.7782797e-11 2.3855740e-02 9.7614425e-01]
 [2.1569745e-08 8.3984965e-01 1.6015036e-01]
 [5.1191613e-02 9.3995154e-01 8.8568293e-03]
 [2.9994551e-02 9.7000545e-01 6.1867158e-18]
 [9.0973479e-01 9.0265274e-02 8.0962785e-11]
 [9.7926140e-01 2.0738611e-02 7.9181873e-14]], shape=(8, 3), dtype=float32)

tf.Tensor([2 2 2 1 1 1 0 0], shape=(8,), dtype=int64)
tf.Tensor([2 2 2 1 1 1 0 0], shape=(8,), dtype=int64)
```

What’s Next?

- Softmax Classifier: Animal classification