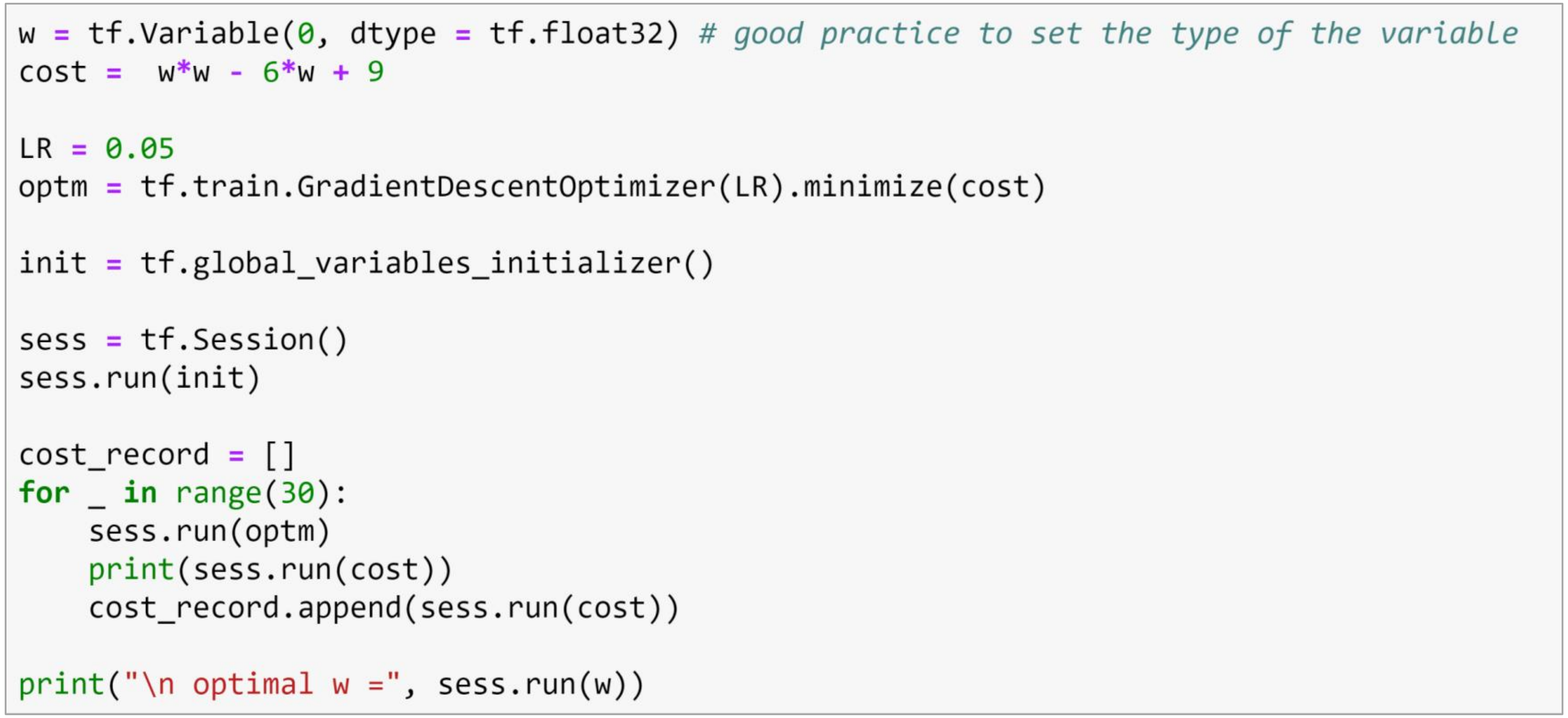
Machine Learning with TensorFlow

Industrial AI Lab.

Prof. Seungchul Lee

TensorFlow as Optimization Solver

$$\min_{\omega} \quad (\omega - 3)^2 = \min_{\omega} \quad \omega^2 - 6\omega + 9$$

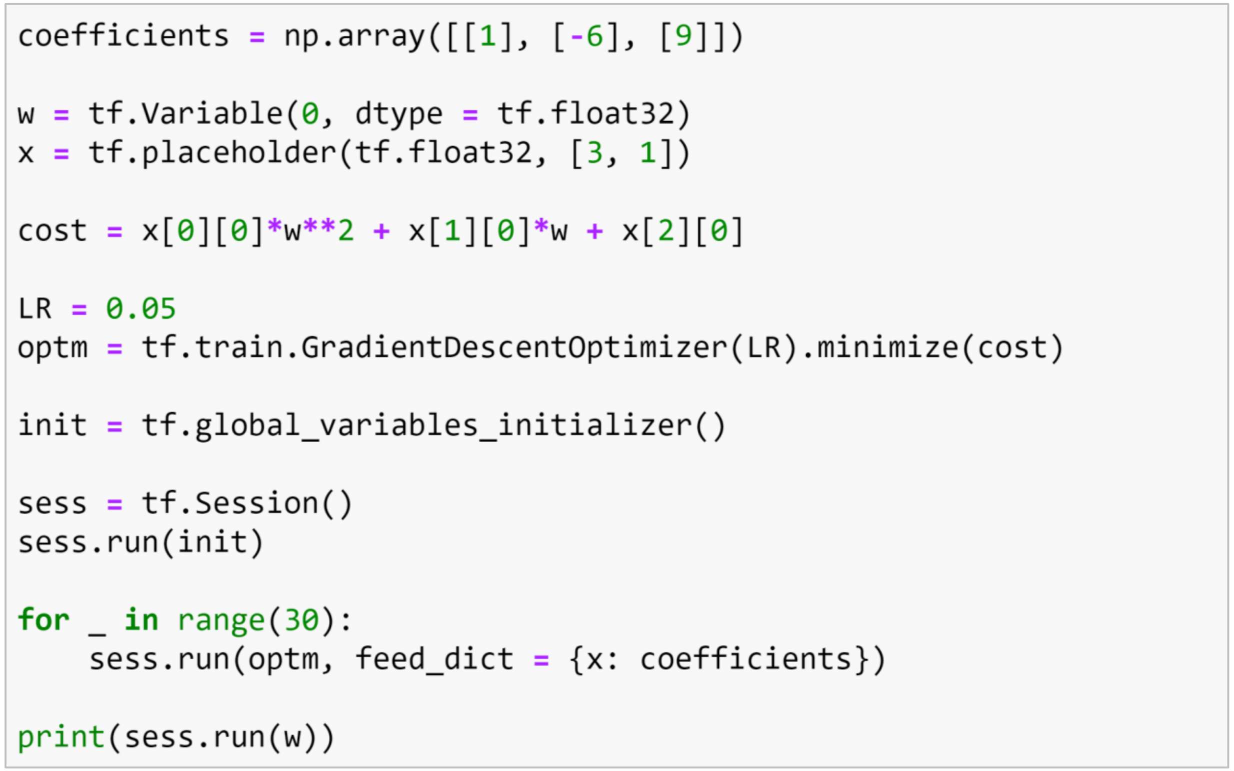


```
w = tf.Variable(0, dtype = tf.float32) # good practice to set the type of the variable
cost =  w*w - 6*w + 9

LR = 0.05
optm = tf.train.GradientDescentOptimizer(LR).minimize(cost)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

cost_record = []
for _ in range(30):
    sess.run(optm)
    print(sess.run(cost))
    cost_record.append(sess.run(cost))

print("\n optimal w =", sess.run(w))
```

TensorFlow as Optimization Solver

- Placeholder

$$\min_{\omega} \ (\omega - 3)^2 = \min_{\omega} \ \omega^2 - 6\omega + 9$$

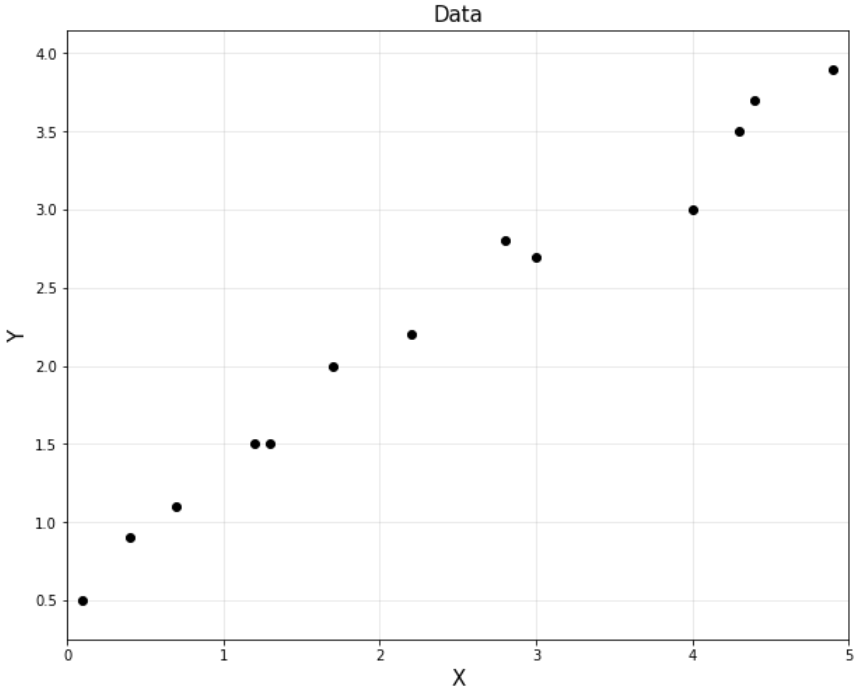


```python
coefficients = np.array([[1], [-6], [9]])

w = tf.Variable(0, dtype = tf.float32)
x = tf.placeholder(tf.float32, [3, 1])

cost = x[0][0]*w**2 + x[1][0]*w + x[2][0]

LR = 0.05
optm = tf.train.GradientDescentOptimizer(LR).minimize(cost)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

for _ in range(30):
    sess.run(optm, feed_dict = {x: coefficients})

print(sess.run(w))
```

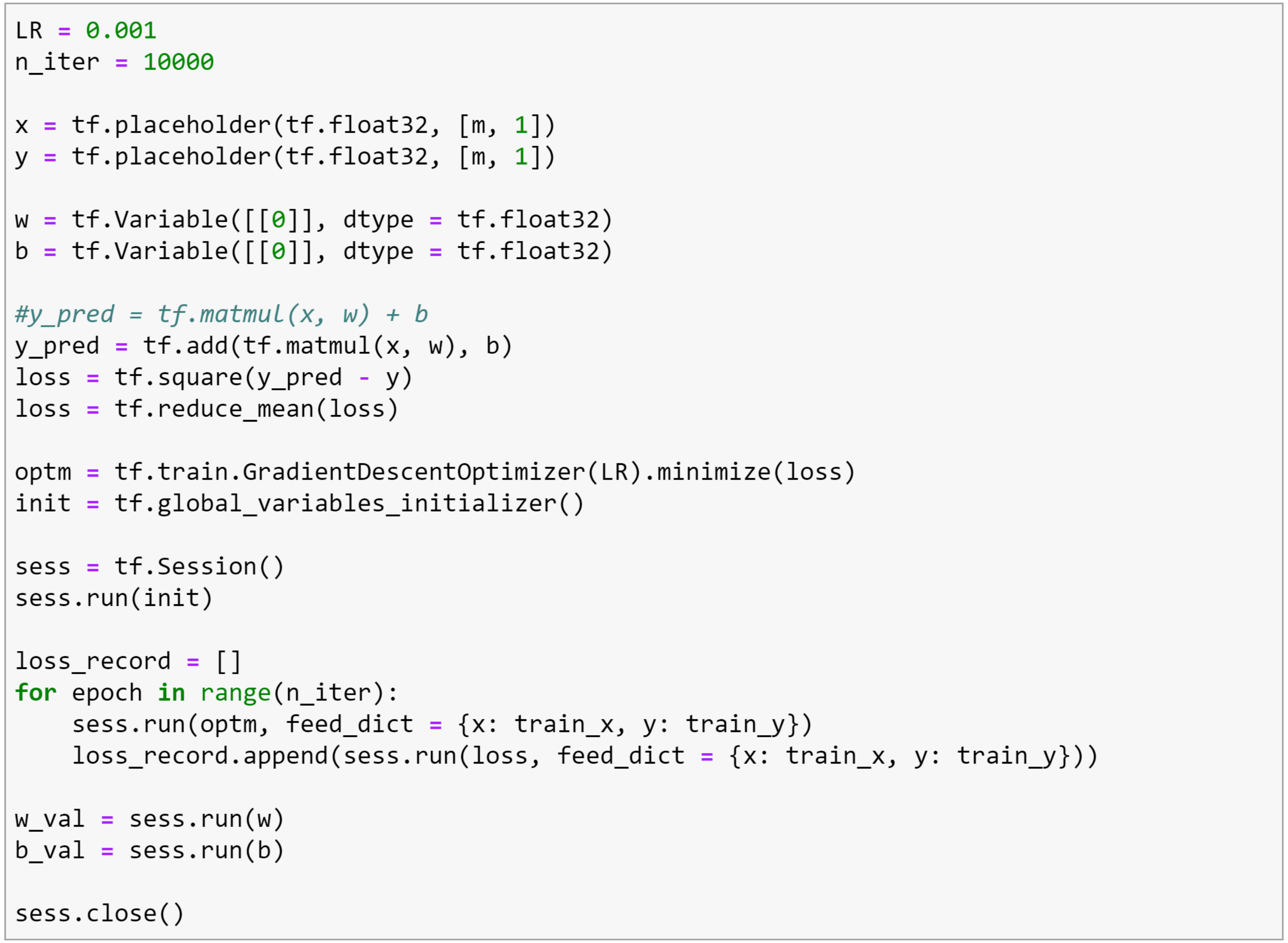
Regression

- Given (x_i, y_i) for $i = 1, \cdots, m$,
- Want to estimate

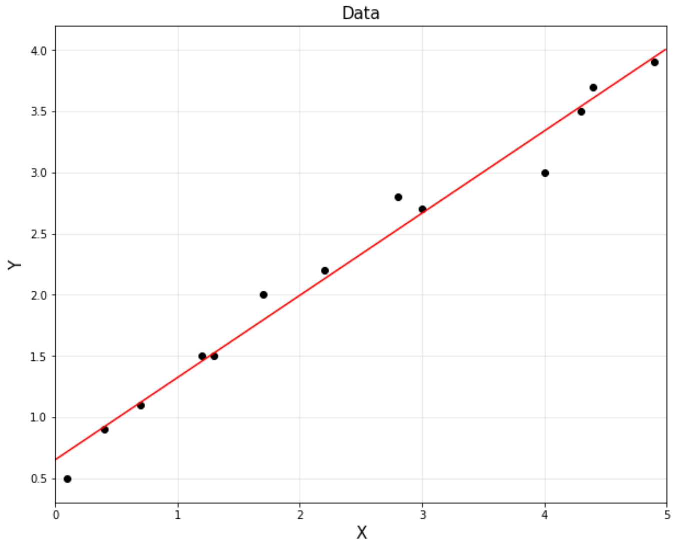
$$\hat{y}_i = \omega x_i + b \qquad \text{such that} \qquad \min_{\omega, b} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2$$

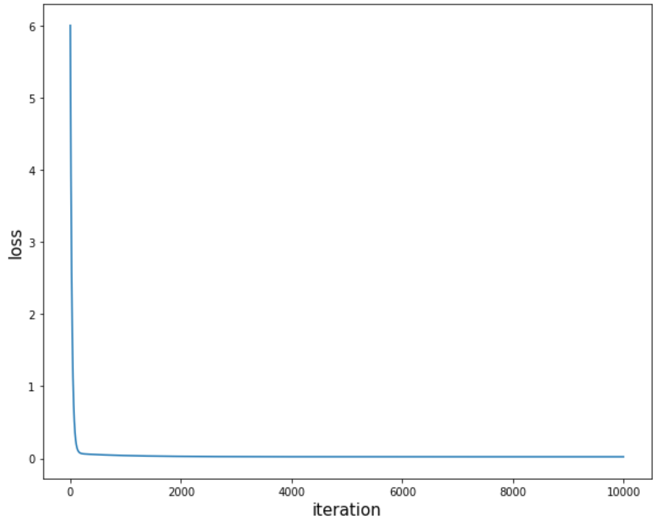


Regression with TensorFlow



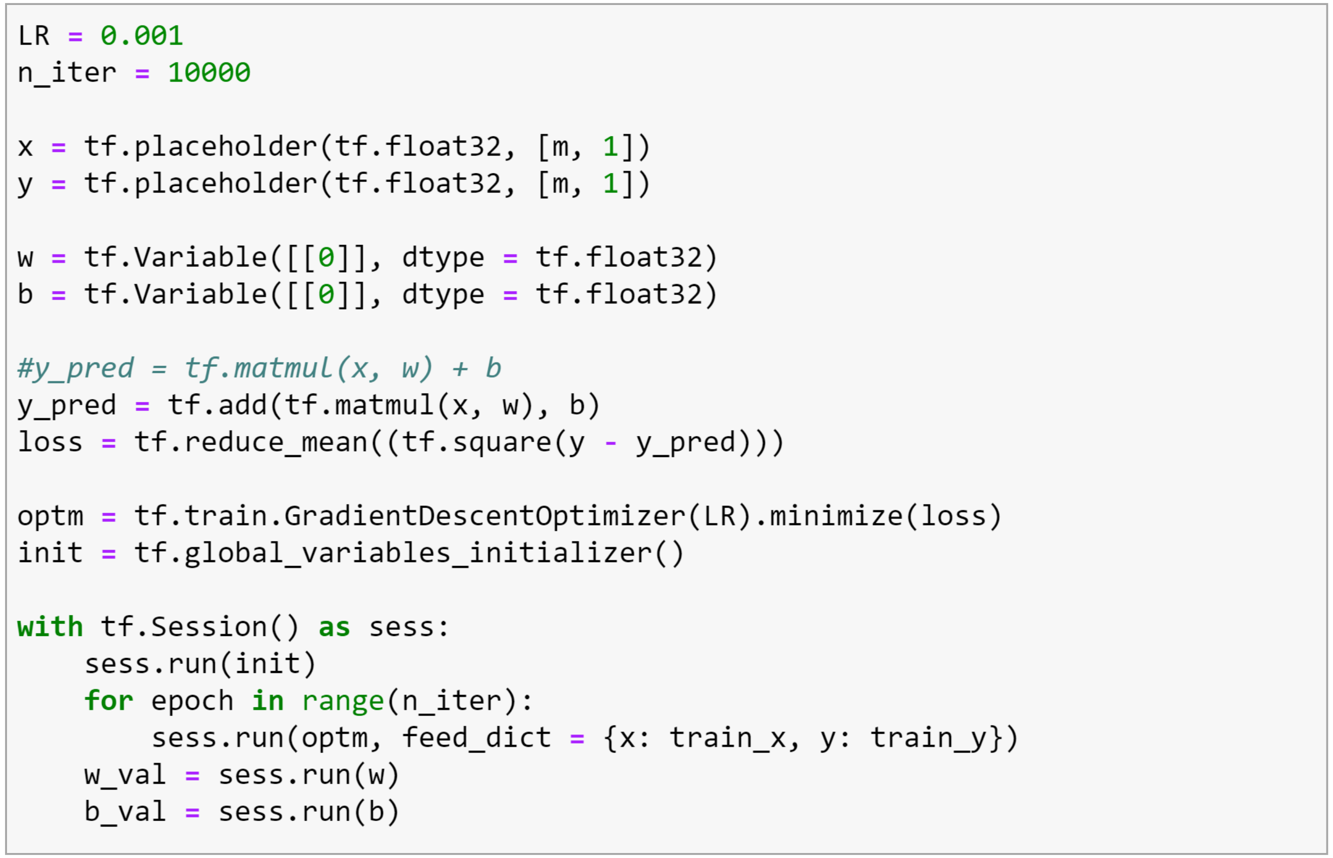
```
LR = 0.001
n_iter = 10000

x = tf.placeholder(tf.float32, [m, 1])
y = tf.placeholder(tf.float32, [m, 1])

w = tf.Variable([[0]], dtype = tf.float32)
b = tf.Variable([[0]], dtype = tf.float32)

#y_pred = tf.matmul(x, w) + b
y_pred = tf.add(tf.matmul(x, w), b)
loss = tf.square(y_pred - y)
loss = tf.reduce_mean(loss)

optm = tf.train.GradientDescentOptimizer(LR).minimize(loss)
init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)

loss_record = []
for epoch in range(n_iter):
    sess.run(optm, feed_dict = {x: train_x, y: train_y})
    loss_record.append(sess.run(loss, feed_dict = {x: train_x, y: train_y}))

w_val = sess.run(w)
b_val = sess.run(b)

sess.close()
```



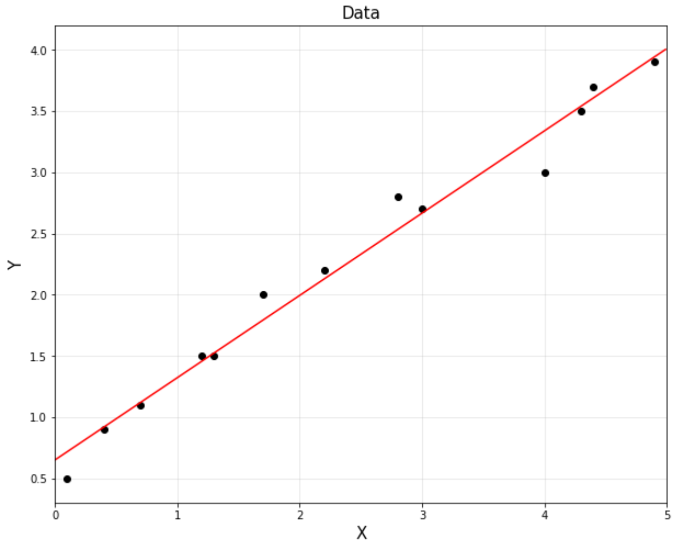


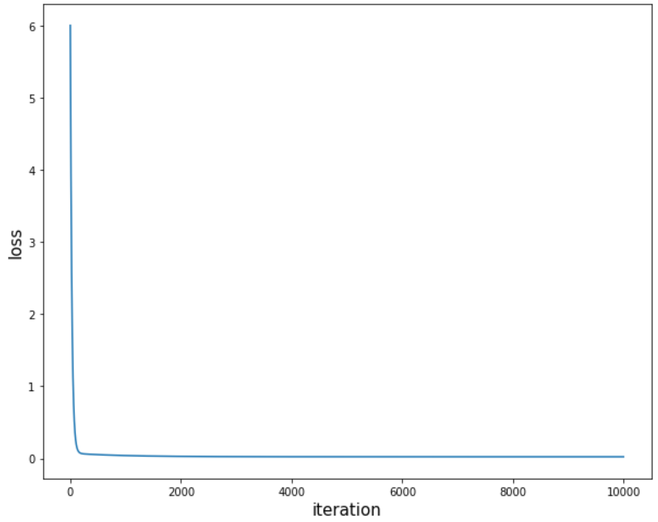
Regression with TensorFlow

- with tf.Session() as sess:



```
LR = 0.001
n_iter = 10000

x = tf.placeholder(tf.float32, [m, 1])
y = tf.placeholder(tf.float32, [m, 1])

w = tf.Variable([[0]], dtype = tf.float32)
b = tf.Variable([[0]], dtype = tf.float32)

#y_pred = tf.matmul(x, w) + b
y_pred = tf.add(tf.matmul(x, w), b)
loss = tf.reduce_mean((tf.square(y - y_pred)))

optm = tf.train.GradientDescentOptimizer(LR).minimize(loss)
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    for epoch in range(n_iter):
        sess.run(optm, feed_dict = {x: train_x, y: train_y})
    w_val = sess.run(w)
    b_val = sess.run(b)
```

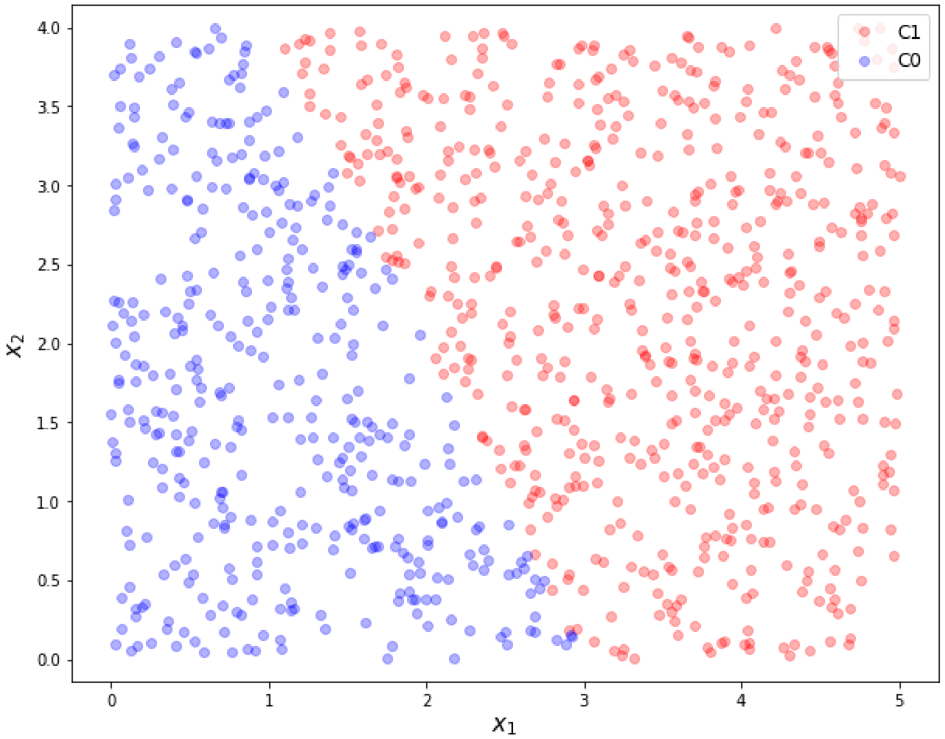




Logistic Regression

$$\omega = \begin{bmatrix} \omega_0 \\ \omega_1 \\ \omega_2 \end{bmatrix}, \qquad x = \begin{bmatrix} 1 \\ x_1 \\ x_2 \end{bmatrix}$$

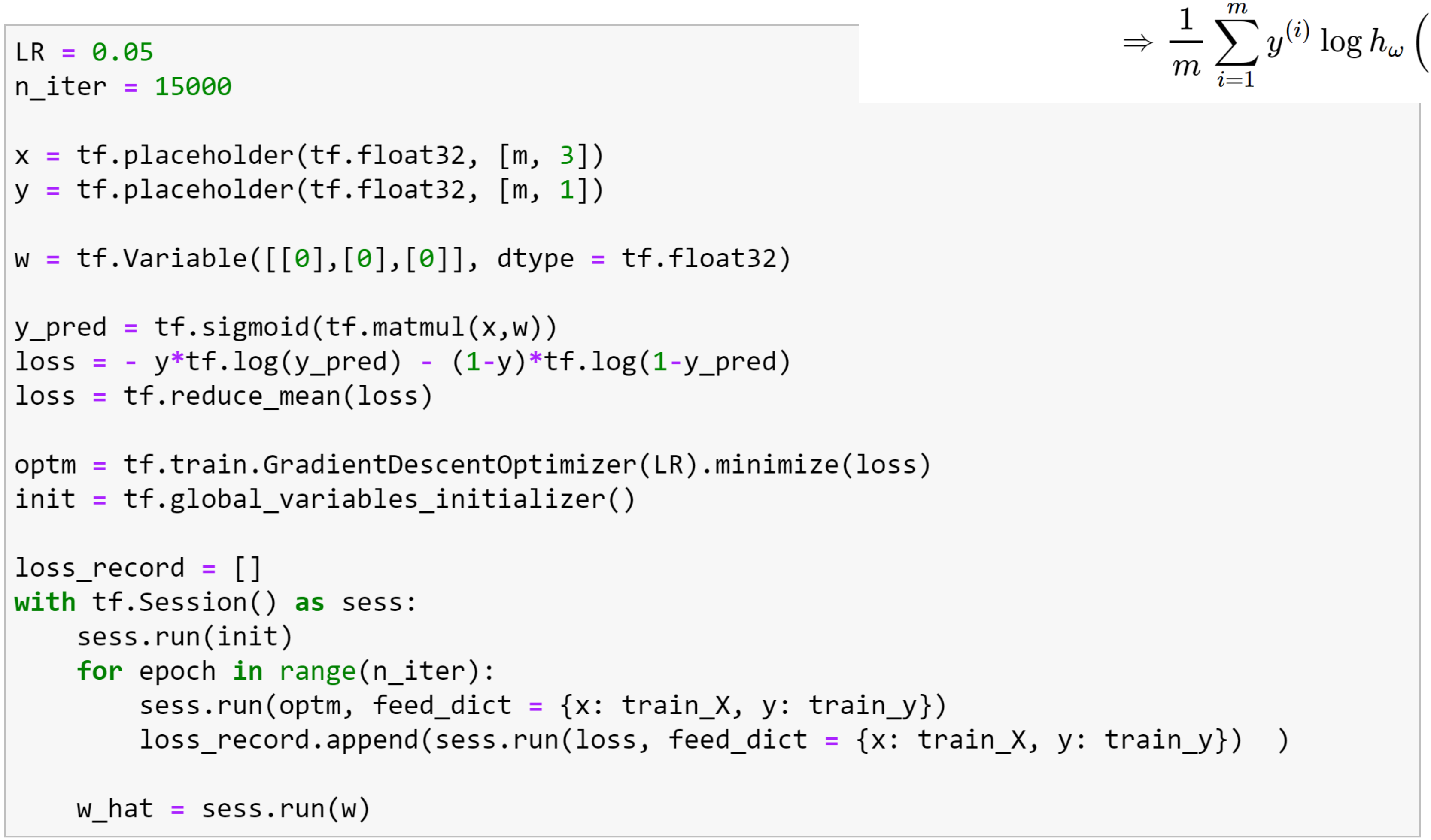
$$X = \begin{bmatrix} \left(x^{(1)}\right)^T \\ \left(x^{(2)}\right)^T \\ \left(x^{(3)}\right)^T \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} \\ 1 & x_1^{(3)} & x_2^{(3)} \\ \vdots & \vdots & \vdots \end{bmatrix}, \quad y = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ y^{(3)} \\ \vdots \end{bmatrix}$$



Logistic Regression with TensorFlow

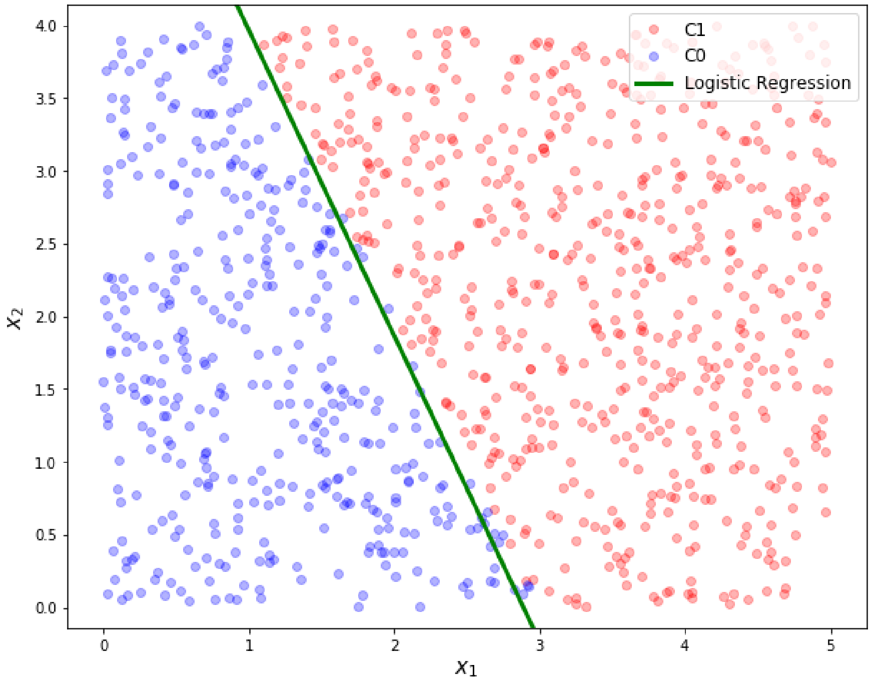
$$\ell(\omega) = \log \mathscr{L}(\omega) = \sum_{i=1}^{m} y^{(i)} \log h_\omega\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\omega\left(x^{(i)}\right)\right)$$

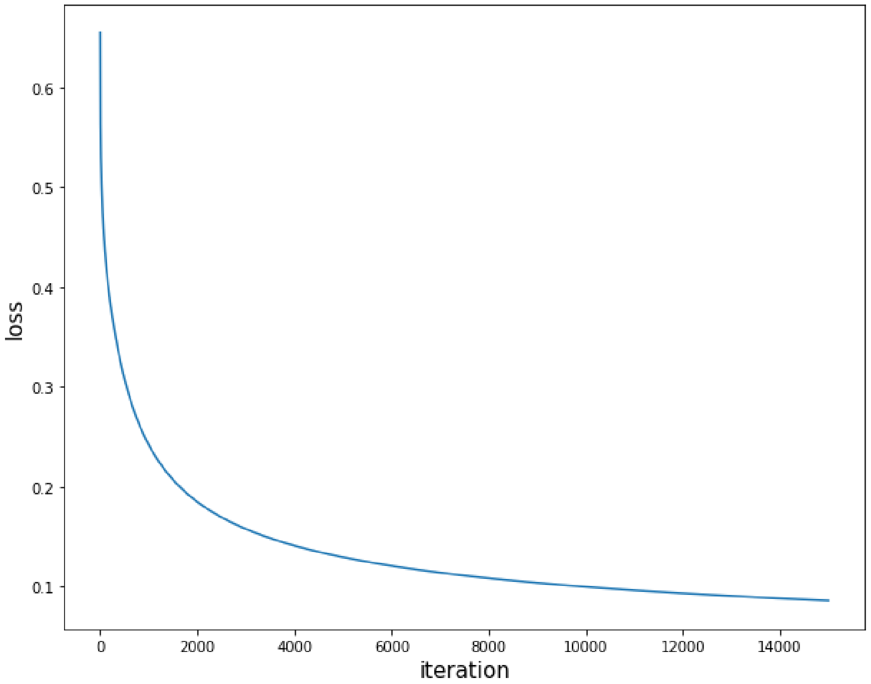
$$\Rightarrow \frac{1}{m} \sum_{i=1}^{m} y^{(i)} \log h_\omega\left(x^{(i)}\right) + \left(1 - y^{(i)}\right) \log\left(1 - h_\omega\left(x^{(i)}\right)\right)$$



```
LR = 0.05
n_iter = 15000

x = tf.placeholder(tf.float32, [m, 3])
y = tf.placeholder(tf.float32, [m, 1])

w = tf.Variable([[0],[0],[0]], dtype = tf.float32)

y_pred = tf.sigmoid(tf.matmul(x,w))
loss = - y*tf.log(y_pred) - (1-y)*tf.log(1-y_pred)
loss = tf.reduce_mean(loss)

optm = tf.train.GradientDescentOptimizer(LR).minimize(loss)
init = tf.global_variables_initializer()

loss_record = []
with tf.Session() as sess:
    sess.run(init)
    for epoch in range(n_iter):
        sess.run(optm, feed_dict = {x: train_X, y: train_y})
        loss_record.append(sess.run(loss, feed_dict = {x: train_X, y: train_y})  )

    w_hat = sess.run(w)
```

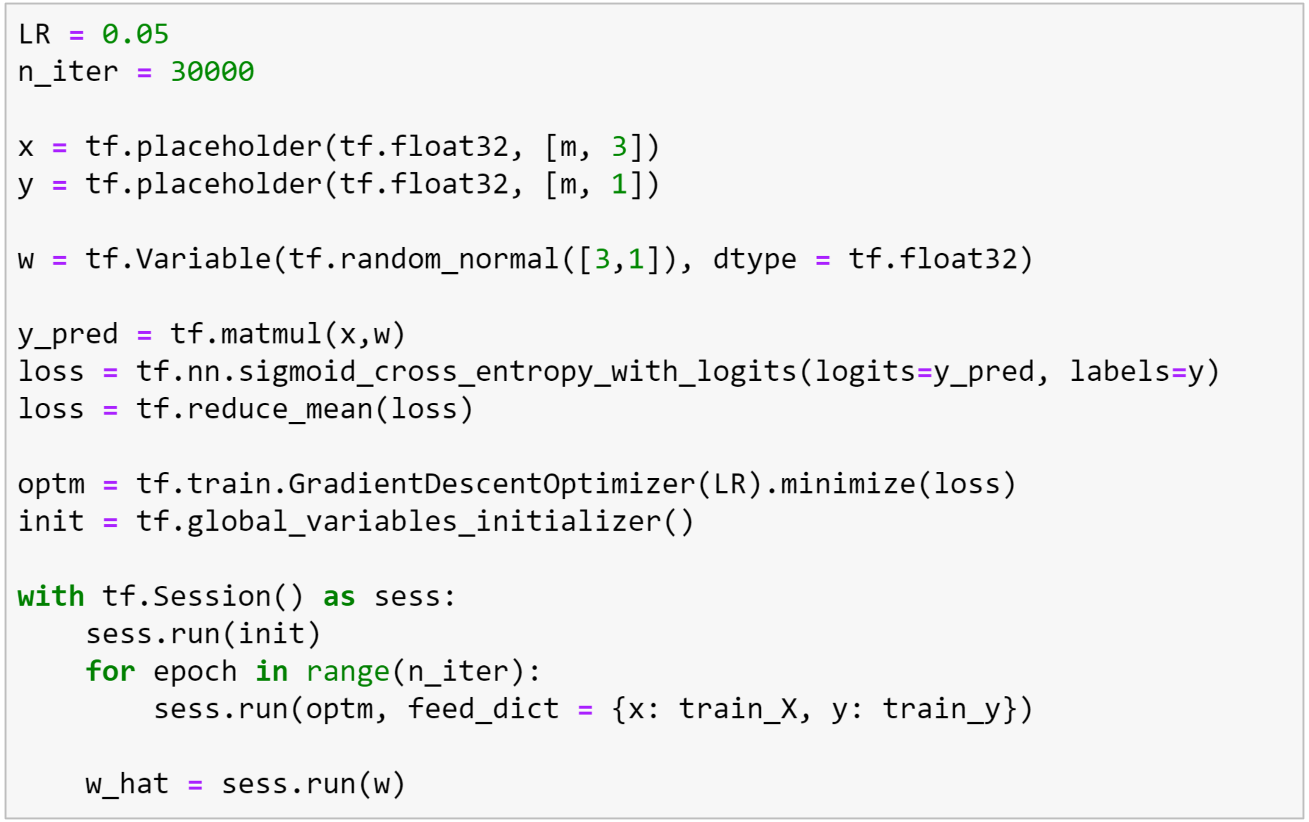
Logistic Regression with TensorFlow





Logistic Regression with TensorFlow

- TensorFlow embedded functions
  - tf.nn.sigmoid_cross_entropy_with_logits for binary classification
  - tf.nn.softmax_cross_entropy_with_logits for multiclass classification



```
LR = 0.05
n_iter = 30000

x = tf.placeholder(tf.float32, [m, 3])
y = tf.placeholder(tf.float32, [m, 1])

w = tf.Variable(tf.random_normal([3,1]), dtype = tf.float32)

y_pred = tf.matmul(x,w)
loss = tf.nn.sigmoid_cross_entropy_with_logits(logits=y_pred, labels=y)
loss = tf.reduce_mean(loss)

optm = tf.train.GradientDescentOptimizer(LR).minimize(loss)
init = tf.global_variables_initializer()

with tf.Session() as sess:
    sess.run(init)
    for epoch in range(n_iter):
        sess.run(optm, feed_dict = {x: train_X, y: train_y})

    w_hat = sess.run(w)
```