

tf1-custom

August 22, 2019

1 Example of TF-1 custom model

```
[1]: import tensorflow as tf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

2 Build training data

```
[2]: n = 100
TRUE_W = 3.0
TRUE_b = 2.0

# random samples from normal distribution
np.random.seed(1)
r = np.random.normal(loc=0, scale=0.5, size=n)

# build data
inputs = np.random.normal(loc=0, scale=0.5, size=n)
outputs = TRUE_W * inputs + TRUE_b + r
```

3 Define model

3.1 Construct model

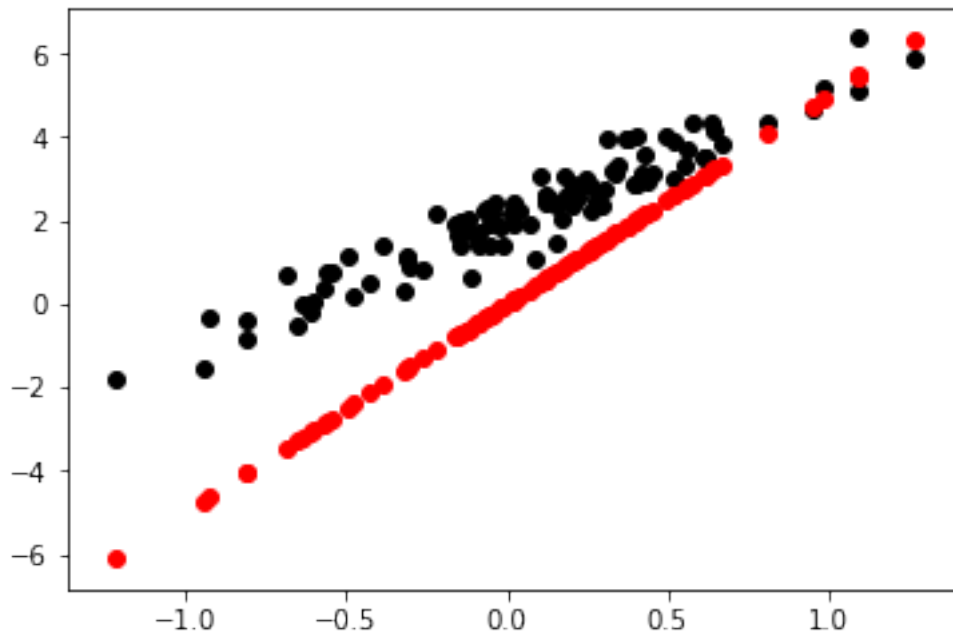
```
[3]: # graph input placeholders
X = tf.placeholder("float")
Y = tf.placeholder("float")

# trainable variables
W = tf.Variable(5.0, name="weight")
b = tf.Variable(0.0, name="bias")

# linear model
pred = tf.add(tf.multiply(X, W), b)
```

3.2 Show model prediction before training

```
[4]: def plot_model(p):  
    plt.scatter(inputs, outputs, c='black')  
    plt.scatter(inputs, p, c='r')  
    plt.show()  
  
[5]: # create a session  
untrained_sess = tf.Session()  
  
# initialize variables  
init = tf.global_variables_initializer()  
untrained_sess.run(init)  
  
untrained_pred = untrained_sess.run(pred, feed_dict={X: inputs, Y: outputs})  
plot_model(untrained_pred)
```



4 Train model

4.1 Setup configurations

```
[6]: N_EPOCHS = 100  
LEARNING_RATE = 0.1  
  
# loss
```

```
loss = tf.reduce_mean(tf.square(pred - Y))

# Gradient descent
optimizer = tf.train.GradientDescentOptimizer(LEARNING_RATE).minimize(loss)
```

4.2 Execute training

```
[7]: with tf.Session() as sess:
      sess.run(init)

      epochs = range(N_EPOCHS)
      for epoch in epochs:
          for (x, y) in zip(inputs, outputs):
              sess.run(optimizer, feed_dict={X: x, Y: y})

      trained_pred = sess.run(pred, feed_dict={X: inputs, Y: outputs})
```

4.3 Show model prediction after training

```
[8]: plot_model(trained_pred)
```

