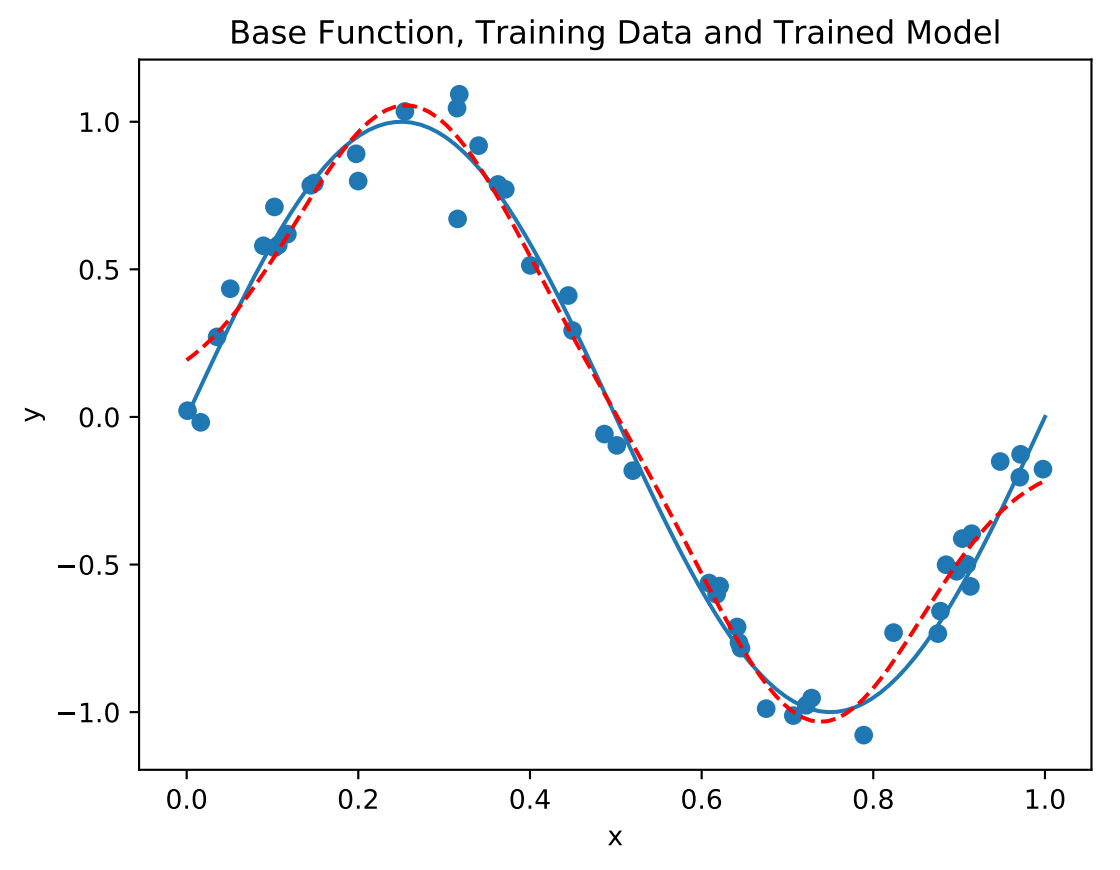
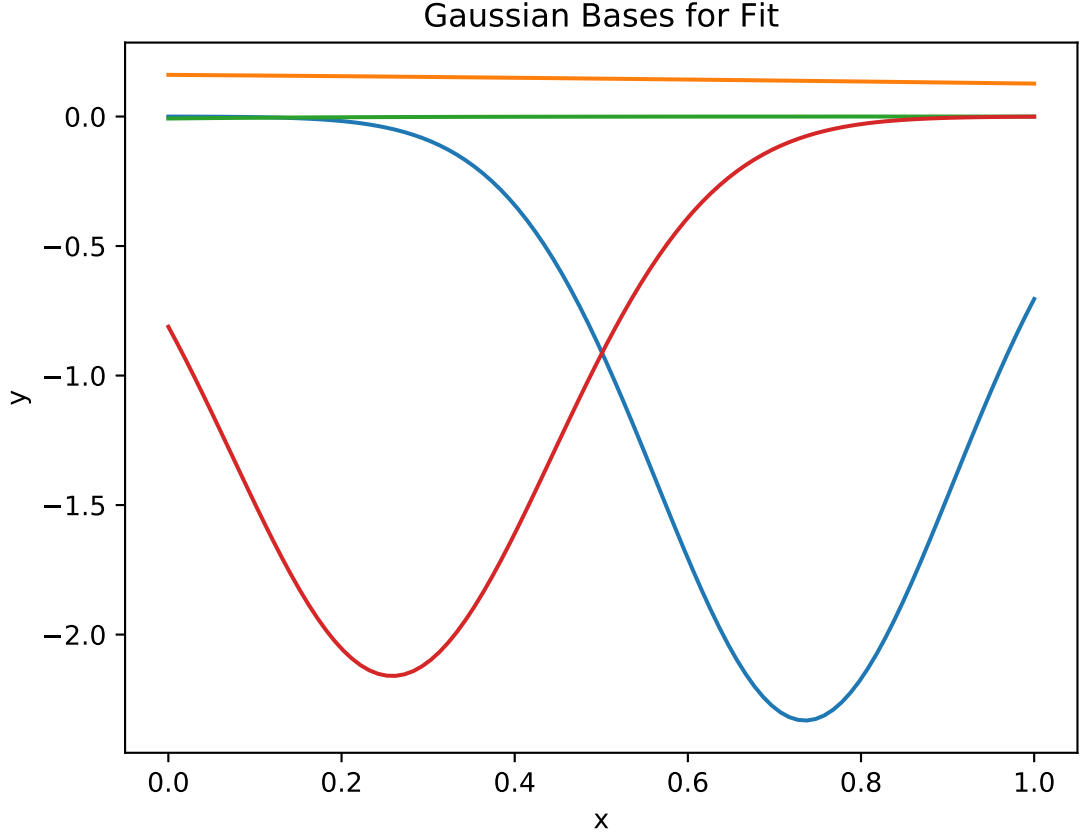
```
 1 #!/bin/python3.6
 2 # Samuel Cheng
 3 # CGML Assignment 1
 4
 5 import numpy as np
 6 import tensorflow as tf
 7 import math
 8 import matplotlib.pyplot as plt
 9 import matplotlib.mlab as mlab
10
11
12 from tqdm import tqdm
13
14 NUM_FEATURES = 4
15 BATCH_SIZE = 32
16 NUM_BATCHES = 300
17
18
19 class Data(object):
20     def __init__(self):
21         num_samp = 50
22         sigma = 0.1
23         np.random.seed(31415)
24
25         self.index = np.arange(num_samp)
26         self.x = np.random.uniform(size=(num_samp, 1))
27         self.y = np.sin(2*np.pi*self.x) + sigma * np.random.normal(
   size=(num_samp, 1))
28
29
30 """
31     def get_batch(self):
32         choices = np.random.choice(self.index, size=BATCH_SIZE)
33
34         return self.x[choices], self.y[choices].flatten()
35 """
36
37 def f(x):
38     w = tf.get_variable('w', [NUM_FEATURES, 1], tf.float32,
39                         tf.random_normal_initializer())
40     b = tf.get_variable('b', [], tf.float32, tf.zeros_initializer())
41     mu = tf.get_variable('mu', [NUM_FEATURES, 1], tf.float32, tf.
   random_normal_initializer())
42     sig = tf.get_variable('sig', [NUM_FEATURES, 1], tf.float32, tf.
   random_normal_initializer())
43     return tf.transpose(tf.matmul(tf.transpose(w), tf.exp((-1)*tf.pow(
   (tf.transpose(x)-mu), 2)/tf.pow(sig, 2))) + b)
44
45
46 x = tf.placeholder(tf.float32,[50,1])
47 y = tf.placeholder(tf.float32,[50,1])
48 y_hat = f(x)
49
50 loss = tf.reduce_mean(tf.pow(y_hat - y, 2))
```

```
optim = tf.train.GradientDescentOptimizer(learning_rate=0.1).minimize
(loss)
init = tf.global_variables_initializer()

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

data = Data()
for i in range(0,3000):
    loss_np, _ = sess.run([loss, optim], feed_dict={x: data.x, y:
data.y})

weightholder = []

print("Parameter estimates:")
for var in tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES):
    weightholder.append(np.array(sess.run(var)).flatten())
    print(
        var.name.rstrip(":0"),
        np.array_str(np.array(sess.run(var)).flatten(), precision=3))

w1 = weightholder[0]
b1 = weightholder[1]
mu1 = weightholder[2]
sig1 = weightholder[3]

x1 = np.linspace(0,1,100,dtype=np.float32)
y1 = np.sin(2*np.pi*x1)

y_hat1 = (w1[0] * tf.exp((-1)*tf.pow((x1-mu1[0]), 2)/tf.pow(sig1[0],
2)) + w1[1] * tf.exp((-1)*tf.pow((x1-mu1[1]), 2)/tf.pow(sig1[1], 2))
          + w1[2] * tf.exp((-1)*tf.pow((x1-mu1[2]), 2)/tf.pow(sig1[2]
, 2)) + w1[3] * tf.exp((-1)*tf.pow((x1-mu1[3]), 2)/tf.pow(sig1[3], 2)
)
          + b1
          )


x2 = np.linspace(0, 1, 100)
print(mu1)
plt.figure(1)

plt.scatter(data.x, data.y)
plt.plot(x1, y1)
plt.plot(x1, sess.run(y_hat1), 'r--')
plt.ylabel('y')
plt.xlabel('x')
plt.title('Base Function, Training Data and Trained Model')

plt.figure(2)
plt.plot(x2, mlab.normpdf(x2, mu1[0], sig1[0]), label='gaussian 1')
plt.plot(x2, mlab.normpdf(x2, mu1[1], sig1[1]), label='gaussian 2')
plt.plot(x2, mlab.normpdf(x2, mu1[2], sig1[2]), label='gaussian 3')
plt.plot(x2, mlab.normpdf(x2, mu1[3], sig1[3]), label='gaussian 4')
```

```
100 plt.ylabel('y')
101 plt.xlabel('x')
102 plt.title('Gaussian Bases for Fit')
103 plt.show()
104
```


Base Function, Training Data and Trained Model
1.0
0.5
0.0
−0.5
−1.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x


Gaussian Bases for Fit
0.0
−0.5
−1.0
−1.5
−2.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x