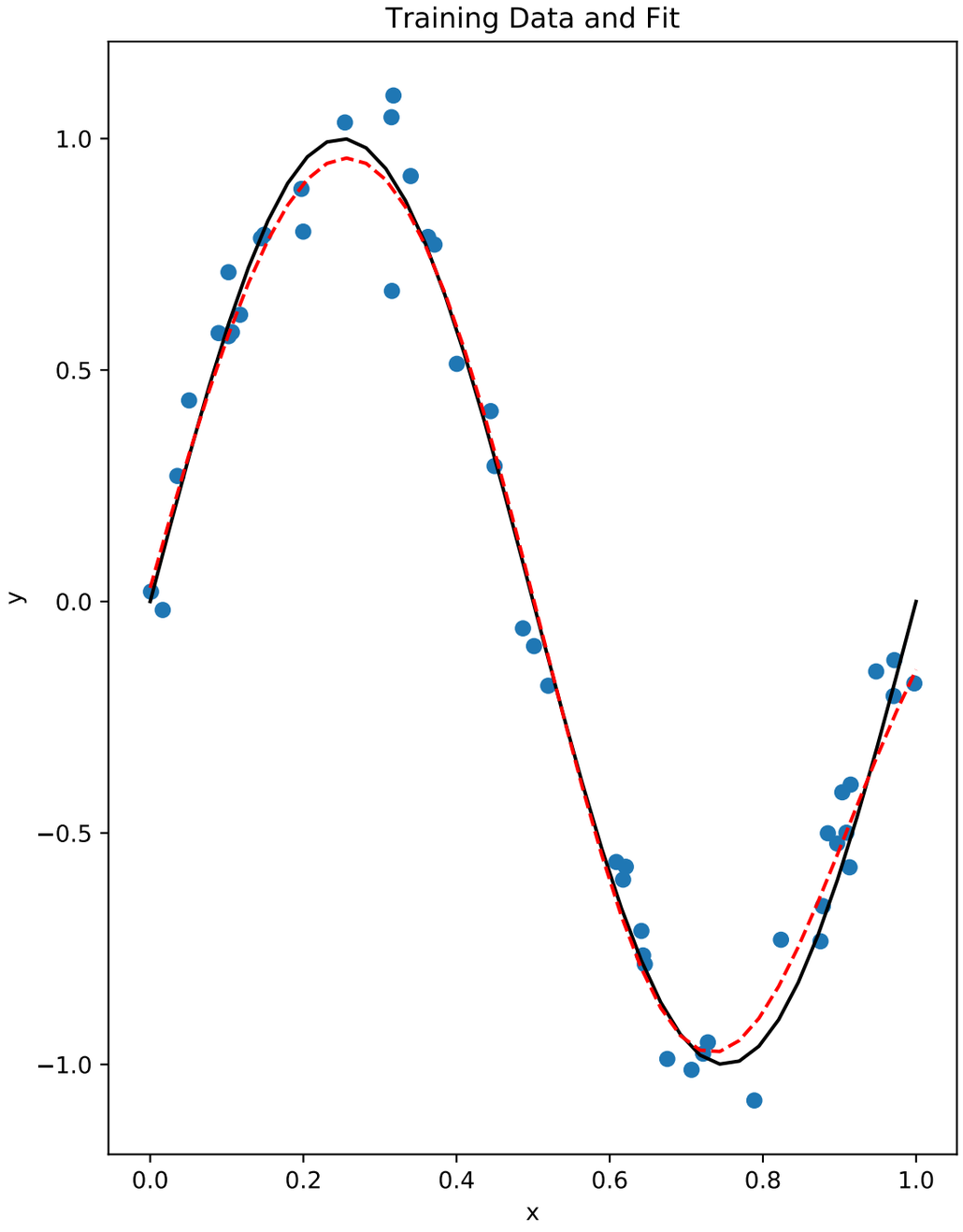
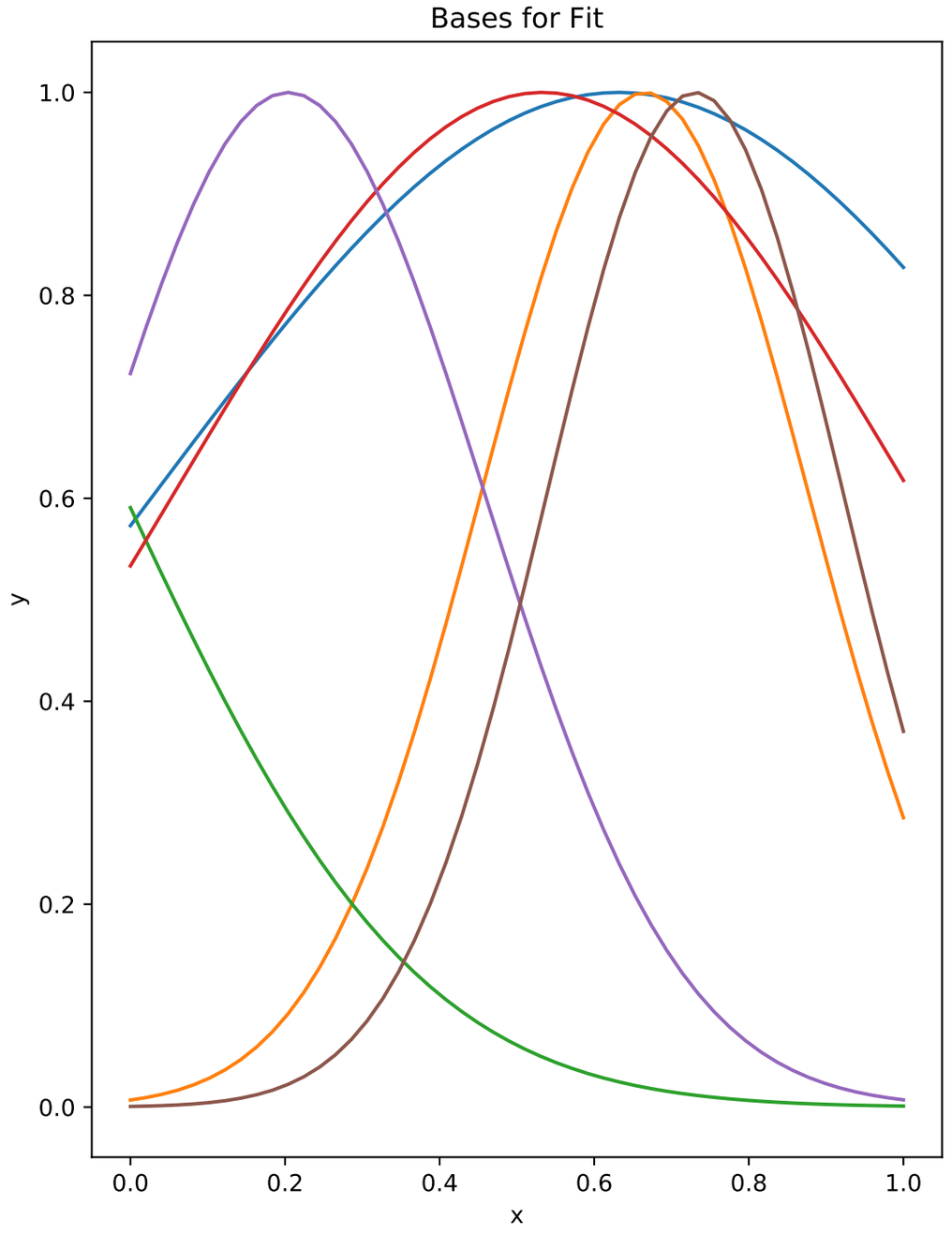
```
#Camille Chow
#ECE 471 Assignment 1

import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
from tqdm import tqdm
from datetime import datetime

M = 6
BATCH_SIZE = 40
NUM_BATCHES = 300

class Data(object):
    def __init__(self):
        num_samp = 50
        sigma = 0.1
        np.random.seed()

        self.index = np.arange(num_samp)
        self.x = np.random.uniform(size=(num_samp))
        self.y = np.sin(2*np.pi*self.x) + np.random.normal(0, sigma, num_samp)

    def get_batch(self):
        choices = np.random.choice(self.index, size=BATCH_SIZE)

        return self.x[choices], self.y[choices].flatten()

def f(x):
        #to be optimized:
    w = tf.get_variable('w', [M, 1], tf.float32, tf.random_normal_initializer())
    b = tf.get_variable('b', [], tf.float32, tf.zeros_initializer())
    mu = tf.get_variable('mu', [M, 1], tf.float32, tf.random_uniform_initializer())
    sig = tf.get_variable('sig', [M, 1], tf.float32, tf.random_uniform_initializer())
    #calculate phi
    phi = tf.exp(- tf.pow((x - mu)/sig,2))
    #calculate yhat
    return tf.squeeze(tf.matmul(tf.transpose(w),phi) + b)

x = tf.placeholder(tf.float32, [BATCH_SIZE])
y = tf.placeholder(tf.float32, [BATCH_SIZE])
y_hat = f(x)

loss = .5 * 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()
#perform gradient descent
for _ in tqdm(range(0, NUM_BATCHES)):
    x_np, y_np = data.get_batch()
    loss_np, _ = sess.run([loss, optim], feed_dict={x: x_np, y: y_np})
#plots
plt.figure(1, figsize=[18,12])
t1 = np.linspace(0,1,BATCH_SIZE)

plt.subplot(121)

plt.plot(t1, np.sin(2 * np.pi * t1), 'k') #perfect sine
plt.scatter(data.x, data.y) #training data
yhats = sess.run(y_hat, feed_dict={x:t1})
plt.plot(t1, yhats, 'r--') #fit

plt.xlabel('x')
plt.ylabel('y')
plt.title('Training Data and Fit')
```

```
plt.subplot(122)
#get mu's and sigma's
for var in tf.get_collection(tf.GraphKeys.TRAINABLE_VARIABLES):
    if var.name.rstrip(":0") == "mu":
        mus = np.array(sess.run(var)).flatten()
    if var.name.rstrip(":0") == "sig":
        sigs = np.array(sess.run(var)).flatten()

t2 = np.linspace(0,1)
#plot gaussians
for i in range(M):
        y = np.exp(-((t2 - mus[i])/sigs[i])**2)
        plt.plot(t2,y)

plt.xlabel('x')
plt.ylabel('y')
plt.title('Bases for Fit')

plt.show()
```


Training Data and Fit
1.0
0.5
0.0
−0.5
−1.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x


Bases for Fit
1.0
0.8
0.6
0.4
0.2
0.0
y
0.0
0.2
0.4
0.6
0.8
1.0
x