```
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt

def add_variable(shape, name):
    variable = tf.get_variable(name=name,
                               dtype=tf.float32,
                               shape=shape,
                               initializer=tf.random_uniform_initializer(minval=-0.75, maxval=0.75)
                               )
    return variable

#Set some constants
N = 50
Deviation = 0.1
M = 6

#Generate data
x_axis = np.sort(np.random.uniform(0.0, 1.0, N)) #the sort makes the graphing easier later
clean_data = np.sin(np.pi*2*x_axis)
noisy_data = clean_data + np.random.normal(0, Deviation, N)

#Create Graph Variables
x = tf.placeholder("float")
y = tf.placeholder("float")
w = add_variable([1,M], "w")
u = add_variable([M,1], "u")
sigma = add_variable([M,1], "sigma")
b = add_variable([], "b")
#Create Graph Ops
basis = tf.exp(-1.0*(x-u)**2/sigma**2)
yhat = tf.matmul(w, basis) + b
error = tf.reduce_mean(0.5*(y-yhat)**2)
train = tf.train.GradientDescentOptimizer(learning_rate=0.01).minimize(error)

#Train and iterate
session = tf.Session()
session.run(tf.global_variables_initializer())
for _ in range(100):
    for data_x, data_y in np.c_[x_axis, noisy_data]:
        session.run([error, train], feed_dict={x:data_x, y:data_y})


w_value = session.run(w)[0]
u_value = session.run(u)
sigma_value = session.run(sigma)
b_value = session.run(b)

print("W: ", w_value, "\nU: ", u_value, "\nSigma: ", sigma_value, "\nB: ", b_value, "\n")

#Generate function from the results...
#A little bit inelegant, but I want to convince myself that
#The result is indeed a superposition of the basis

x_axis_model = np.linspace(0.0,1.0,N*100)

basis_curves = []
for weight, u_, s_ in zip(w_value, u_value, sigma_value):
    y_curve = np.power((x_axis_model-u_[0]),2)
    y_curve = -1*y_curve/s_[0]**2
    y_curve = weight*np.exp(y_curve)
    basis_curves.append(y_curve)

y_axis_model = np.zeros(N*100)
for curve in basis_curves:
    y_axis_model = np.add(y_axis_model, curve)

y_axis_model = y_axis_model + b_value
#Plot noise, sine wave, and manifold
plt.plot(x_axis_model, y_axis_model, ’r−−’, label=’Model Result’)
plt.plot(x_axis, clean_data, ’b’, label=’Sine’)
plt.plot(x_axis, noisy_data, ’g^’, label=’Noisy Data’)
plt.title("Gaussian Regressing of Sine Wave with Noise")
plt.xlabel("x")
plt.ylabel("y")
plt.legend()
plt.show()

for curve in basis_curves:
    plt.plot(x_axis_model, curve)

plt.title("Gaussian Basis Curves")
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```