hw1

January 30, 2017

1 Program for Gaussian Radial Basis Function Regression

1.1 Krishna Thiyagarajan

1.2 ECE - 411 - Computational Graphs for Machine Learning

1.3 Professor Chris Curro

1.4 Homework Assignment #1

1.5 January 29, 2017

```
In [1]: import tensorflow as tf
        import numpy as np
        import matplotlib
        import matplotlib.pyplot as plt
        import warnings
        from IPython.display import Image
        from IPython.core.display import HTML
        warnings.filterwarnings('ignore')

        N = 50; # Number of samples
        #Hyper parameters
        M = 6; # Six gaussian curves
        runs = 100; # Number of iterations
        rateLearn = 1e-2; # Learn rate for training
        regConst = 0;  # Ignore regualarization for now
        sigmaNoise = 0.1 # Noise on data
        muNoise = 0 # Noise is centered around original data
```

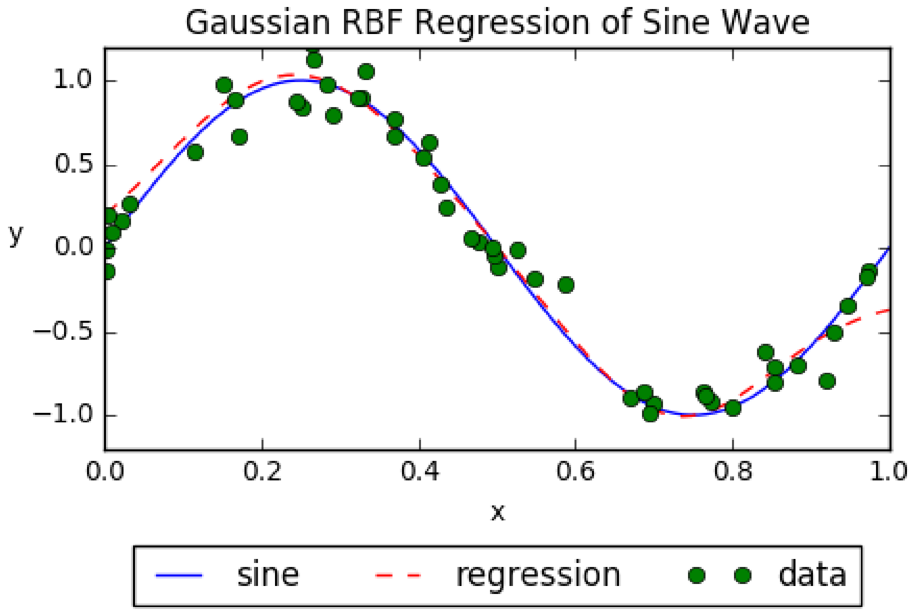
```
In [2]: def f(x):
                return np.sin(2*np.pi*x);

        def gaussian(x, mu, sigma):
                return tf.exp(-0.5*(x-mu)**2/sigma**2);

        def data():
            for _ in range(N):
                x = np.random.uniform()
                y  = f(x) + np.random.normal(loc = muNoise, scale = sigmaNoise)
                yield x, y

        def defVariable(shape, name):
                var = tf.get_variable(name=name,
                                           dtype=tf.float32,
                                           shape=shape,
                                           initializer=tf.random_uniform_initialize
                                      #Works better as U(-1,1) as oppoed to N(0, 0.
                )
                tf.add_to_collection('modelVars', var)
                tf.add_to_collection('l2', tf.reduce_sum(tf.square(var)))
                return var

In [3]: class GaussianRBFModel():
            def __init__(self, sess, data, iterations, learnRate, gamma):
                self.sess = sess
                self.data = data
                self.iterations = iterations
                self.learnRate = learnRate
                self.gamma = gamma
                self.buildModel()

            def buildModel(self):
                self.x = tf.placeholder(tf.float32, shape=[])
                self.y = tf.placeholder(tf.float32, shape=[])

                w = defVariable([1, M], 'w')
                mu = defVariable([M,1], 'mu')
                sigma = defVariable([M,1], 'sigma')
                b = defVariable([], 'b')
                phi = gaussian(self.x, mu, sigma)

                self.yhat = b + tf.matmul(w, phi);
                self.mse = tf.reduce_mean(0.5*tf.square(self.yhat - self.y))
                self.l2_penalty = tf.reduce_sum(tf.get_collection('l2'))
                self.loss = self.mse + self.gamma * self.l2_penalty;

            def initTrainer(self):
                modelVars = tf.get_collection('modelVars')
                self.optim = (tf.train.GradientDescentOptimizer(learning_rate=self.
                self.sess.run(tf.global_variables_initializer())

            def iterateTrainer(self, step, x, y):
                loss, _ = self.sess.run([self.loss,self.optim],
                                                  feed_dict={self.x : x, self.y : y
                #if step % 20 == 0:
                        #print('Step: {} \t Loss: {}'.format(step, loss))
            def train(self):
                for step in range(self.iterations+1):
                    for x, y in self.data():
                        self.iterateTrainer(step, x, y)

            def infer(self, x):
                    y = np.asscalar(self.sess.run(self.yhat, feed_dict={self.x : x}
                    #print(x, y);
                    return y;
```

```
In [4]: sess = tf.Session()
        model = GaussianRBFModel(sess, data, iterations=runs, learnRate=rateLearn,
        model.initTrainer()
        model.train()

        with tf.variable_scope("", reuse = True):
                w = sess.run(tf.get_variable("w"))
                mu = sess.run(tf.transpose(tf.get_variable("mu")))
                sigma = sess.run(tf.transpose(tf.get_variable("sigma")))
                b = sess.run(tf.get_variable("b"));

        print("W =", w);
        print("μ =", mu);
        print("σ =", sigma);
        print("b =", b);
```

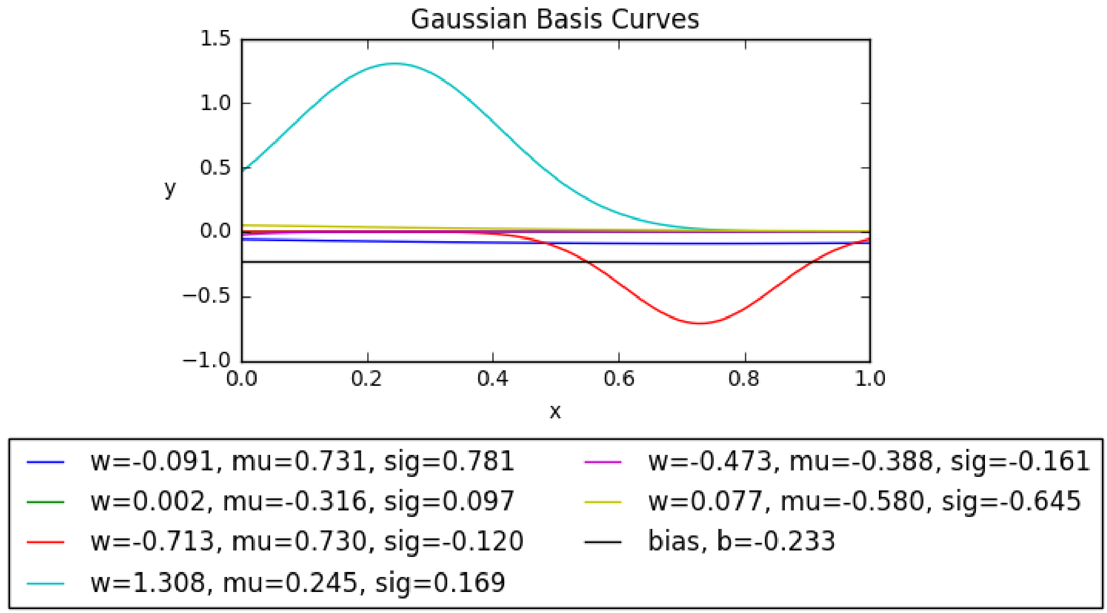
```
W = [[-0.09081559  0.00228237 -0.71259677  1.30823374 -0.47260669  0.07686862]]
μ = [[ 0.73111904 -0.31556752  0.72970486  0.24464355 -0.38821542 -0.5798738 ]]
σ = [[ 0.78119057  0.09740398 -0.11980312  0.1693102  -0.16091737 -0.64478976]]
b = -0.233338
```

```
In [5]: x_model = np.linspace(0.0, 1.0, 100);

        y_model = [];

        for a in x_model:
                y_model.append(model.infer(a));
        y_model = np.array(y_model);

        x_real = np.linspace(0.0, 1.0, 100);
        y_real = f(x_real);

        examples, targets = zip(*list(data()))
```

```
In [6]: fig, ax = plt.subplots(1,1)
        fig.set_size_inches(5, 3)
        plt.plot(x_real, y_real, 'b-',label='sine')
        plt.plot(x_model, y_model, 'r--', label='regression')
        plt.plot(np.array(examples), np.array(targets), 'go', label="data");
        plt.xlim([0.0, 1.0])
        plt.ylim([-1.2, 1.2])
        ax.set_xlabel('x')
        ax.set_ylabel('y').set_rotation(0)
        plt.title('Gaussian RBF Regression of Sine Wave')
        plt.tight_layout()
        plt.legend(loc=9, bbox_to_anchor=(0.5, -0.2), ncol=3)
        plt.show()
```



```
In [7]: fig, ax = plt.subplots(1,1)
        fig.set_size_inches(5, 3)
        ax.set_xlabel('x')
        ax.set_ylabel('y').set_rotation(0)
        plt.xlim([0.0,1.0]);
        plt.ylim([-2,2]);
        plt.title('Gaussian Basis Curves')
        plt.tight_layout()
        ax.autoscale(enable=True, axis='y', tight=False)
        #Auto scaled to visualize all functions between (0,1)

        x_gauss = np.linspace(0.0, 1.0, 100);
        for k in range(M):
                with sess.as_default():
                        y_gauss = np.asscalar(w[0][k]) * gaussian(x_gauss, mu[0][k
                        lab = "w=%0.3f, mu=%0.3f, sig=%0.3f" % (np.asscalar(w[0][k
                plt.plot(x_gauss, y_gauss, label=lab);
        plt.plot(x_gauss, np.full(shape=x_gauss.shape, fill_value=b),label="bias, b
        plt.legend(loc=9, bbox_to_anchor=(0.5,-0.2), ncol=2)
        plt.show();
```



1.6 Sample Previous Runs

```
In [8]: Image(url= "./regression1.png")

Out[8]: <IPython.core.display.Image object>

In [9]: Image(url="./basis1.png")

Out[9]: <IPython.core.display.Image object>

In [10]: Image(url="./regression2.png")

Out[10]: <IPython.core.display.Image object>

In [11]: Image(url= "./basis2.png")

Out[11]: <IPython.core.display.Image object>

In [ ]:
```