In [1]:

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

In [2]:

```
data_train =  np.loadtxt('/home/bhavy/Dropbox/7th-semester/courses/ML/Assignments/A
data_test =  np.loadtxt('/home/bhavy/Dropbox/7th-semester/courses/ML/Assignments/As
```

In [3]:

```
gamma = 0.1
lamb = 0.1
L = 2
rms = np.zeros(5).reshape(5, 1)
Llist = [2, 5, 20, 50, 100]
N =  len(data_train)
N_test = len(data_test)
```

In [4]:

```
#plotting the data
x, y = data_train[:, 0].reshape(N, 1) , data_train[:, 1].reshape(N, 1)
x_test, y_test = data_test[:, 0].reshape(N_test, 1) , data_test[:, 1].reshape(N_tes
```

In [5]:

```
def selecting_L_random_data_points(L):
    idx = np.argsort(np.random.random(N))[:L]
    return data_train[idx]
```

In [6]:

```
def kernel(xn, xm):
    #gamma = 0.1 given
    return np.exp(-gamma*np.dot(xn-xm, xn-xm)) #rbf kernel
```

In [7]:

```
def compute_X(X_psi, landmark):
    for i in range(N):
        for j in range(L):
            X_psi[i, j] = kernel(x[i],landmark[j])
    return X_psi
```

In [8]:

```
def compute_weight(X_psi, y):
    Il = np.identity(L, dtype='float32')
    return np.dot(np.dot( np.linalg.inv( np.dot(np.transpose(X_psi), X_psi) + lamb*
```

In [9]:

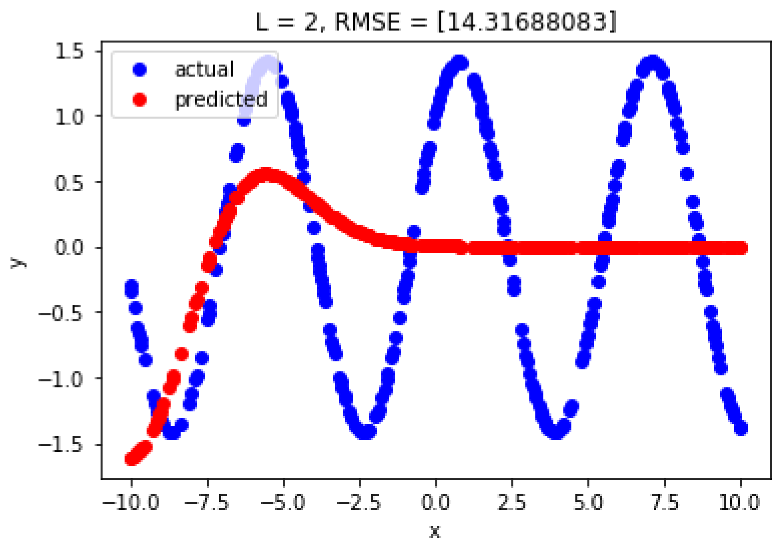
```
def pred(x_test, w):
    psi_test = np.zeros((L, 1),dtype='float32')
    for j in range(L):
        psi_test[j] = kernel(x_test, landmark[j])
    return np.dot(np.transpose(w), psi_test)
```

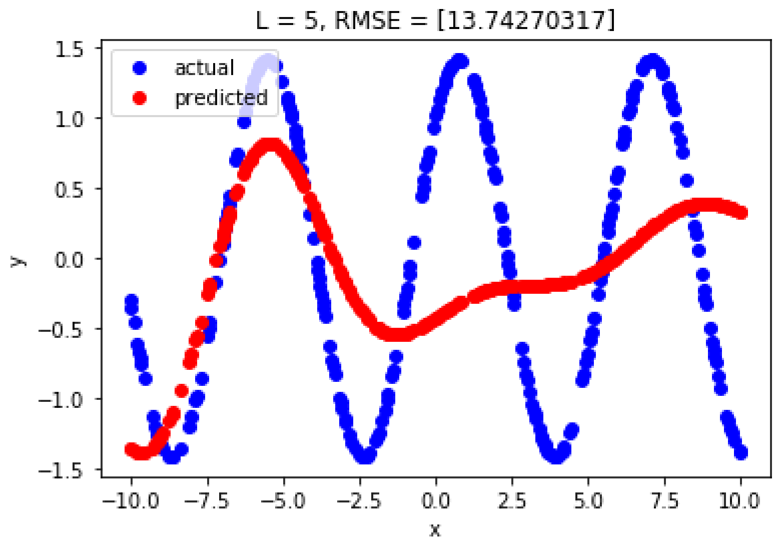
In [10]:

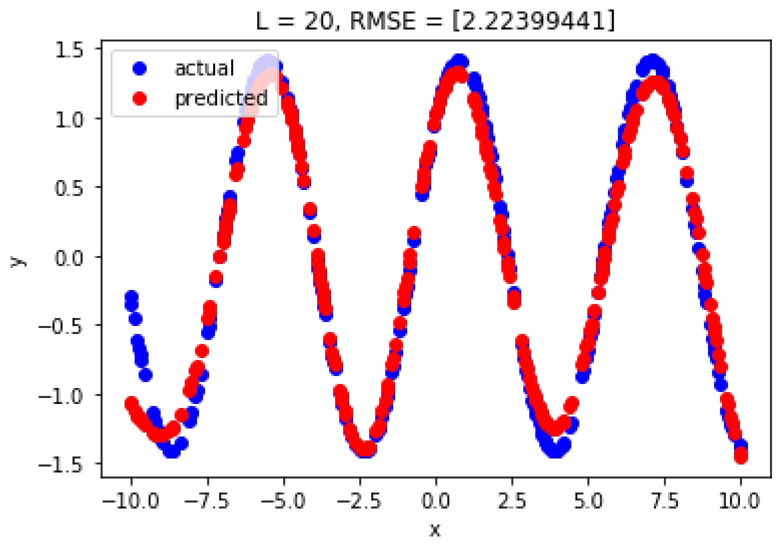
```
def rmse(y1, y2):
    return np.sqrt(np.mean(np.dot(np.transpose(y1 - y2), y1-y2)))
```

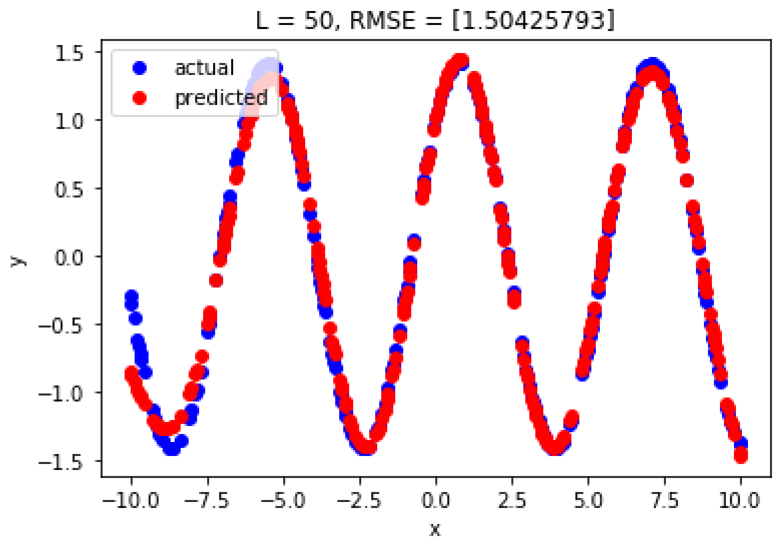
In [11]:

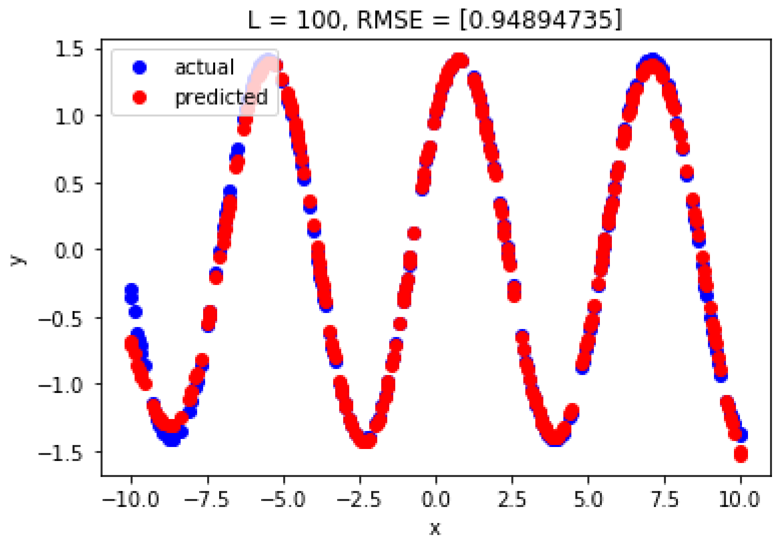
```
idx = 0
for L in Llist:
    landmark = selecting_L_random_data_points(L)[:, 0]
    X_psi = np.zeros((N, L), dtype='float32')
    X_psi = compute_X(X_psi, landmark)
    w = np.zeros((L,1), dtype='float32')
    w = compute_weight(X_psi, y)
    y_pred = np.zeros(N_test).reshape(N_test, 1)
    for i in range(N_test):
        y_pred[i] = pred(x_test[i], w)
    rms[idx] = rmse(y_pred, y_test)
    plt.figure(idx + 1)
    plt.plot(x_test, y_test, 'bo', label = 'actual')
    plt.plot(x_test, y_pred, 'ro', label = 'predicted')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend(loc = 'upper left')
    plt.title('L = '+ str(L) + ', RMSE = ' + str(rms[idx]))
    plt.savefig(str(idx)+'-landmark-ridge-regression.png')
    idx = idx+1
```





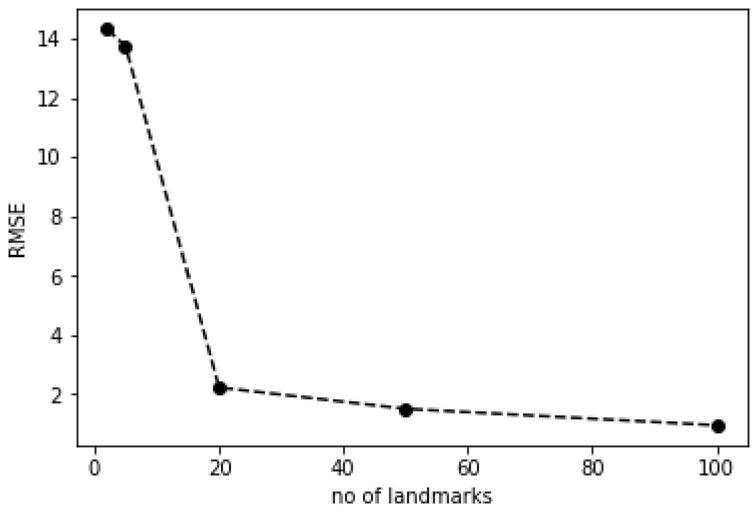

L = 20, RMSE = [2.22399441]
actual
predicted
y
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


L = 50, RMSE = [1.50425793]
actual
predicted
y
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0


L = 100, RMSE = [0.94894735]
actual
predicted
y
x
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

In [12]:

```
plt.plot(Llist, rms, 'ko--')
plt.xlabel('no of landmarks')
plt.ylabel('RMSE')
plt.savefig('landmark-rmse.png')
```


RMSE
no of landmarks