Exercises1

February 2, 2021

1 DATA.STAT.770 - Exercise set 1

1.1 Problem 1

1.1.1 a)

[9]:
```
import numpy as np
from numpy.linalg import norm

# a)
dims = [2,3,5,7,10,13,17]
N = 10000000
print('Proportion of points inside hypersphere:')
for dim in dims:
    points = np.random.uniform(low=-1, high=1, size=(N,dim))
    distances = norm(points, axis=1)
    n_inside = np.sum(distances <= 1)
    print('At dimension {} : {}/{} = {}'.format(dim, n_inside, N, n_inside/N))
```

```
Proportion of points inside hypersphere:
At dimension 2 : 7853032/10000000 = 0.7853032
At dimension 3 : 5235998/10000000 = 0.5235998
At dimension 5 : 1645457/10000000 = 0.1645457
At dimension 7 : 368728/10000000 = 0.0368728
At dimension 10 : 25125/10000000 = 0.0025125
At dimension 13 : 1078/10000000 = 0.0001078
At dimension 17 : 13/10000000 = 1.3e-06
```

1.1.2 b)

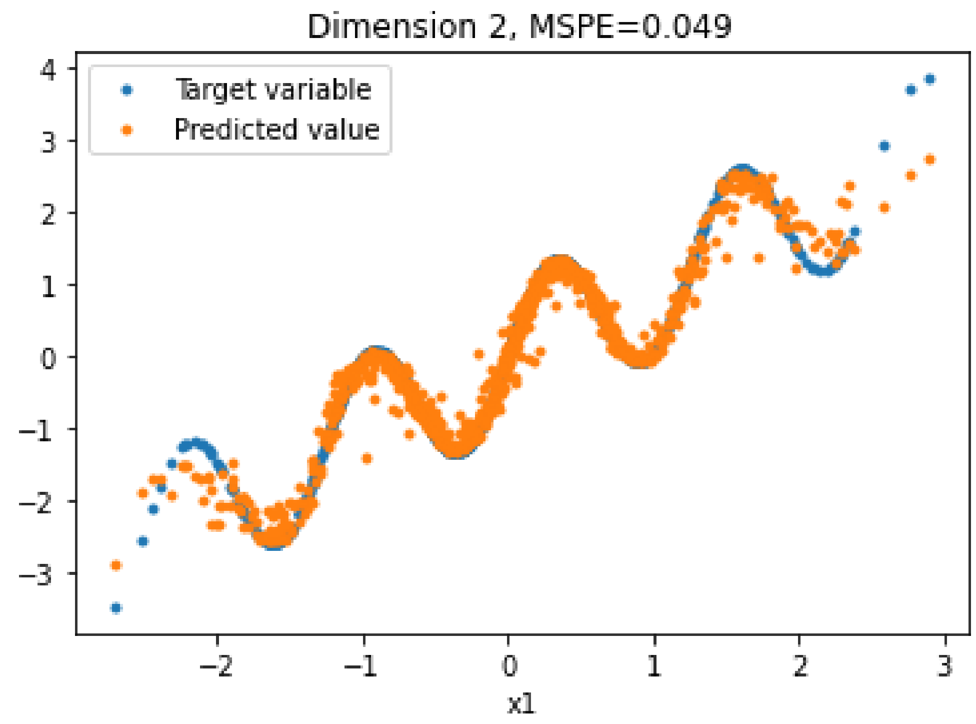
[10]:
```
# b)
print('Proportion of points inside the shell of hypersphere:')
for dim in dims:
    points = np.random.uniform(low=-1, high=1, size=(N,dim))
    distances = norm(points, axis=1)

    inside_hs_mask = distances <= 1
    n_inside_hs = np.sum(inside_hs_mask)
    distances = distances[inside_hs_mask]
    inside_shell_mask = distances >= 0.95
    n_inside_shell = np.sum(inside_shell_mask)

    print('At dimension {} : {}/{} = {}'.format(dim, n_inside_shell,␣
 ↪n_inside_hs, n_inside_shell/n_inside_hs))
```

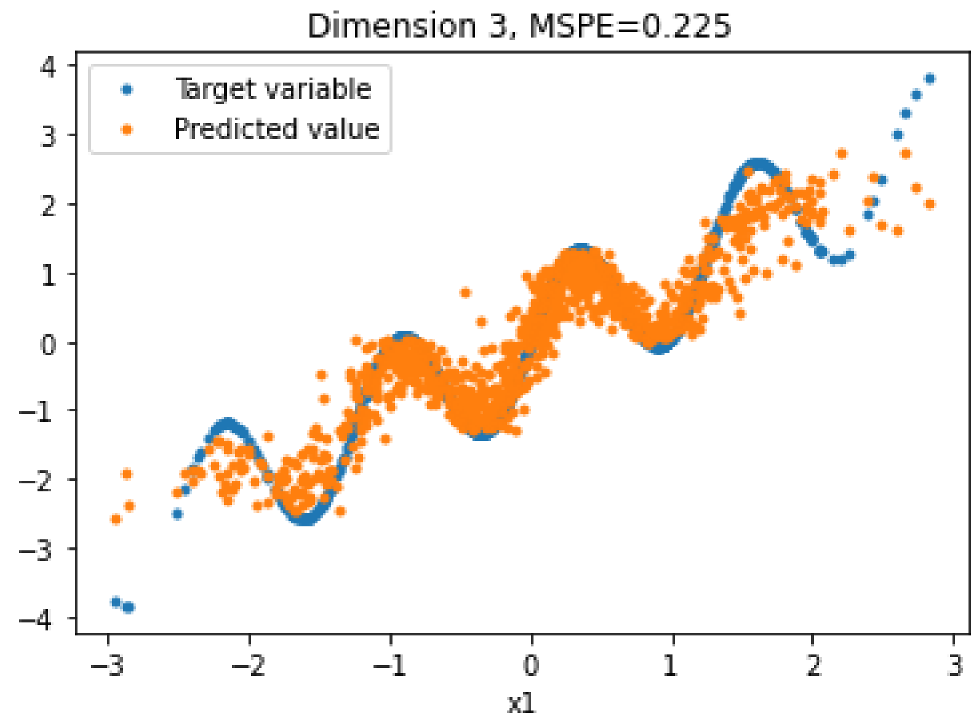
```
Proportion of points inside the shell of hypersphere:
At dimension 2 : 766958/7854144 = 0.09765010674619666
At dimension 3 : 746520/5235165 = 0.14259722472930653
At dimension 5 : 372387/1645135 = 0.22635649961857235
At dimension 7 : 111490/369421 = 0.30179659521250823
At dimension 10 : 10088/25057 = 0.4026020672865866
At dimension 13 : 561/1102 = 0.5090744101633394
At dimension 17 : 7/9 = 0.7777777777777778
```

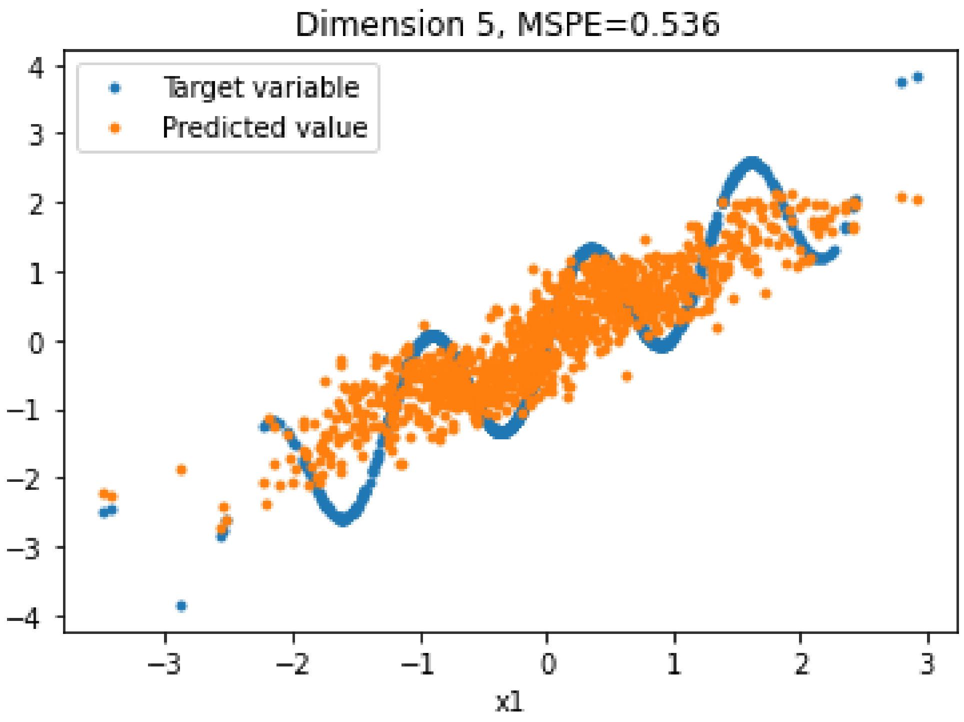
1.2 Problem 2

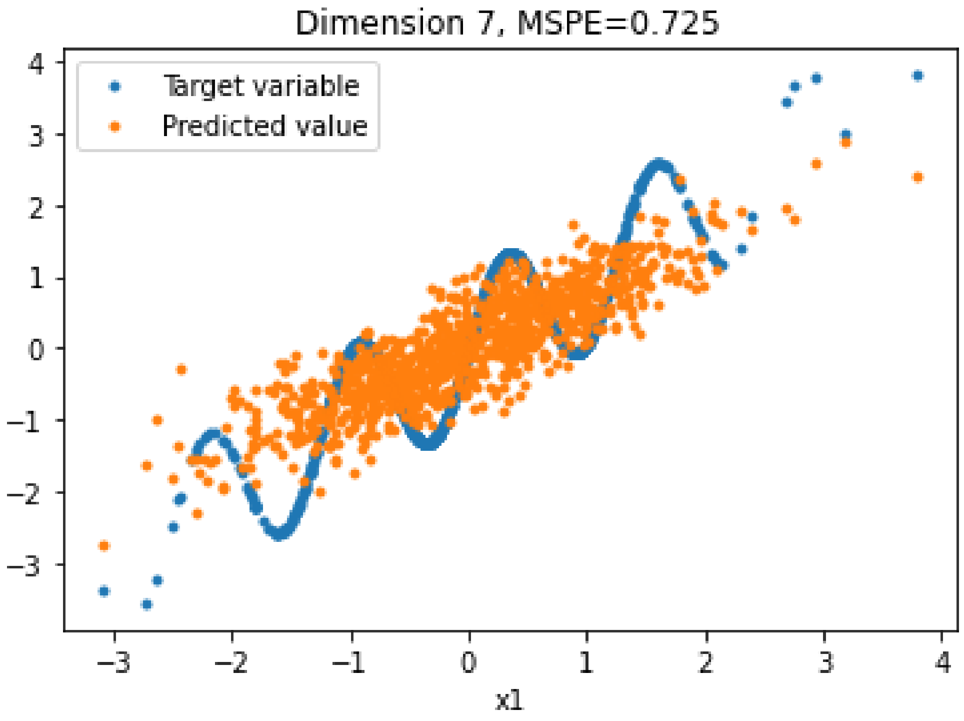
[8]:
```
import numpy as np
from numpy.linalg import norm
import matplotlib.pyplot as plt

def generate_data(N, dim):
    xs = np.random.multivariate_normal(np.zeros(dim), np.diag(np.ones(dim)),␣
 ↪size=N)
    # Target
    ys = xs[:,0] + np.sin(5*xs[:,0])

    trainX = xs[:1000]
    trainY = ys[:1000]
    testX = xs[1000:]
    testY = ys[1000:]

    return trainX, trainY, testX, testY

def fiveNN(trainX, trainY, testX):
    predY = []
    for tstx in testX:
        distances = norm(tstx-trainX, axis=1)
        idxs = np.argpartition(distances, 5)[:5]
        py = np.mean(trainY[idxs])
        predY.append(py)

    return predY

def pred_error_mse(testY, predY):
    sqdiff = (testY-predY)**2

    return np.mean(sqdiff)

def plot(x1, testY, predY, n, dim, mspe):
    plt.figure(n)
    plt.title('Dimension {}, MSPE={}'.format(dim, np.round(mspe,3)))
    tv, = plt.plot(x1, testY, '.', label='Target variable')
    pv, = plt.plot(x1, predY, '.', label='Predicted value')
    plt.xlabel('x1')
    plt.legend(handles=[tv,pv])

    return

dims = [2,3,5,7,10,13,17]
N = 2000
for n in range(len(dims)):
    d = dims[n]
    print('Dimension {}'.format(d))
    trainX, trainY, testX, testY = generate_data(N, d)
    predY = fiveNN(trainX, trainY, testX)

    # Mean-squared prediction error
    mspe = pred_error_mse(testY, predY)
    # Plotting
    plot(testX[:,0], testY, predY, n, d, mspe)
```

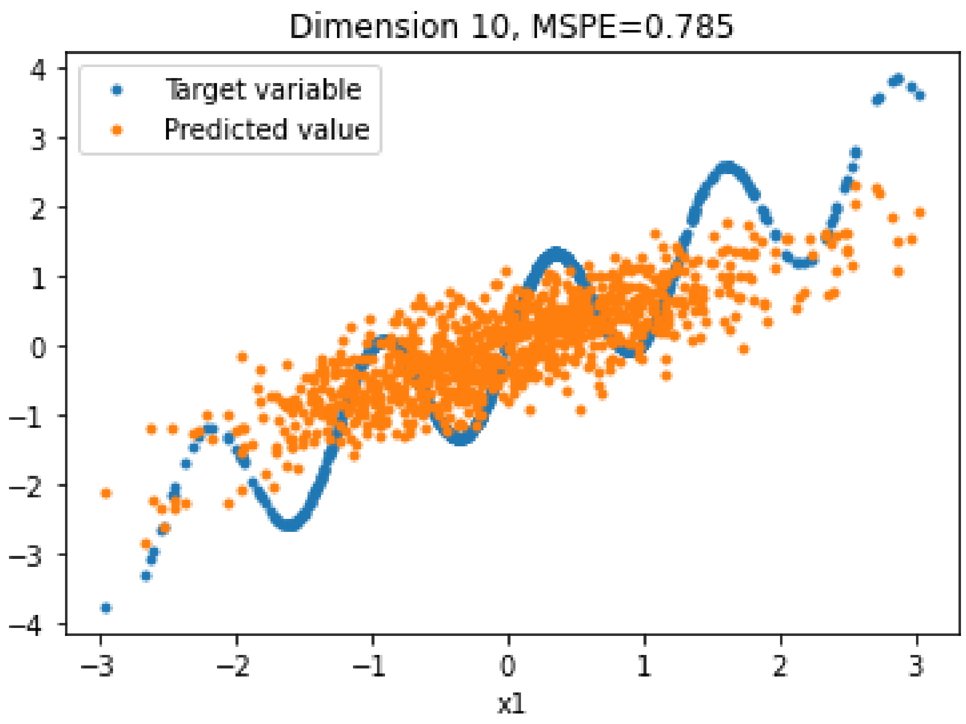
```
Dimension 2
Dimension 3
Dimension 5
Dimension 7
Dimension 10
Dimension 13
Dimension 17
```


Dimension 2, MSPE=0.049
Target variable
Predicted value
x1

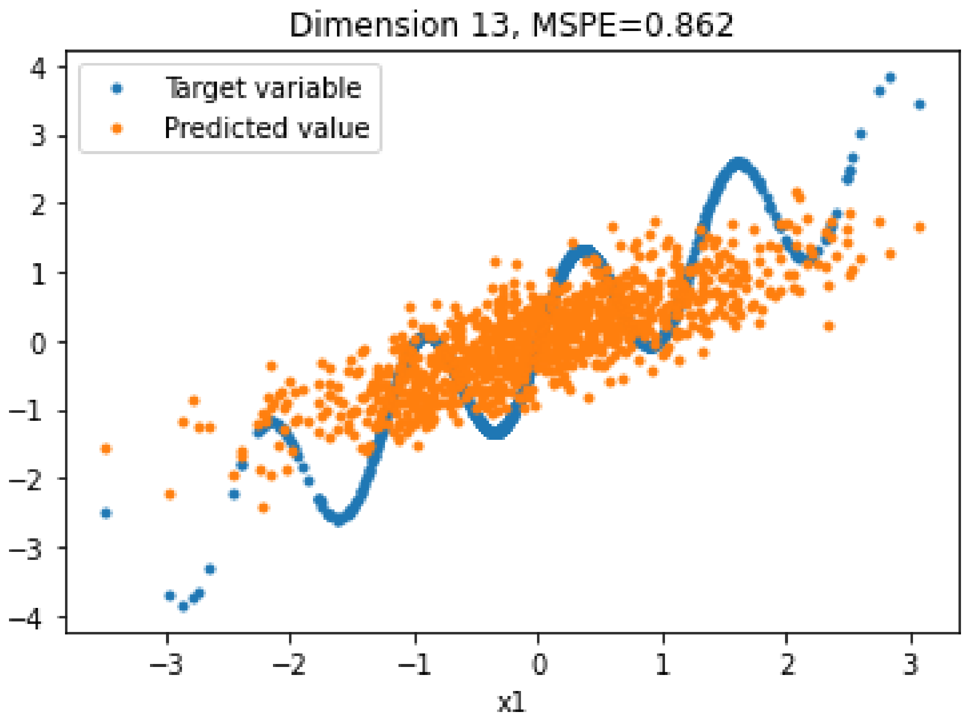

Dimension 3, MSPE=0.225
Target variable
Predicted value
x1


Dimension 5, MSPE=0.536
Target variable
Predicted value
x1
4
3
2
1
0
−1
−2
−3
−4
−3
−2
−1
0
1
2
3

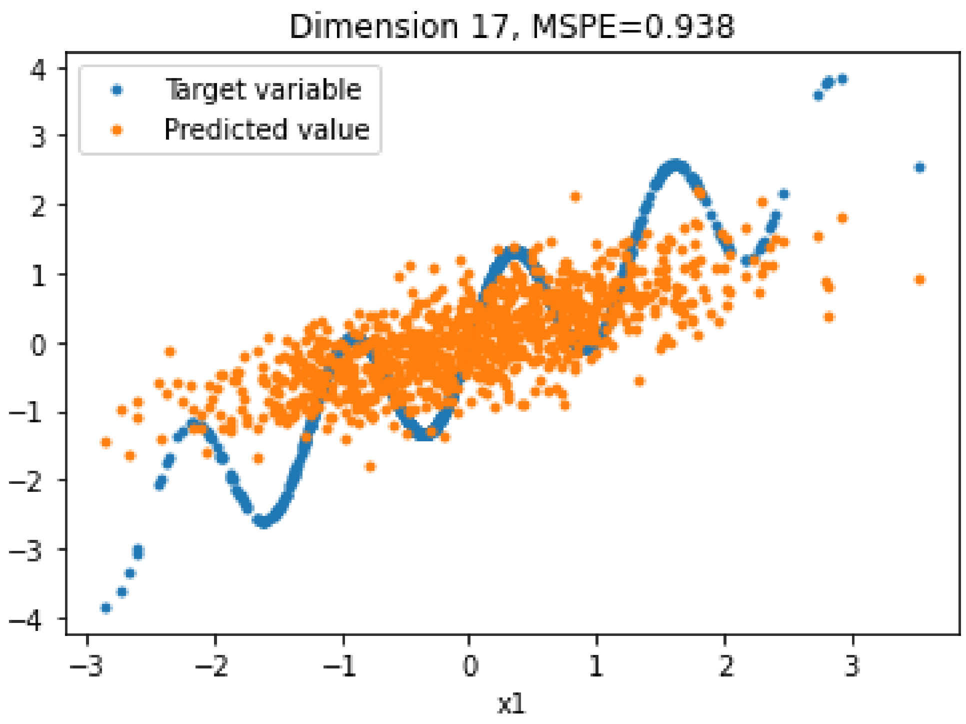

Dimension 7, MSPE=0.725
Target variable
Predicted value
x1
4
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3
4



Dimension 10, MSPE=0.785
Target variable
Predicted value
x1



Dimension 13, MSPE=0.862
Target variable
Predicted value
x1


Dimension 17, MSPE=0.938
Target variable
Predicted value
x1

```
[ ]:
```