Foundations of Machine Learning

Assignment 4

Question 1: Generate 1000 real number for the variable X from the uniform distribution U [0,1]. Construct the training set T = { (x1,y1),(x2,y2),.....,(x100,y100)} using the relation

Yi = sin(2 π xi) + εi where εi ~ N(0,0.25).

In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import math
import pandas as pd
from sklearn import metrics
from sklearn.model_selection import train_test_split
import time
import random
```

In [2]:
```
train_x = np.random.uniform(0, 1, 1000)
train_e = np.random.normal(0, 0.25, 1000)
train_y = np.sin(2 * math.pi * train_x) + train_e

test_x = np.random.uniform(0, 1, 50)
test_e = np.random.normal(0, 0.25, 50)
test_y = np.sin(2 * math.pi * test_x) + test_e
```

In [3]:
```
data_m9 = pd.DataFrame([train_x**9, train_x**8, train_x**7, train_x**6, train_x**5, train_x**4, train_x**3, train_x**2,
data_m9 = data_m9.T
data_m9.columns = ['x^9', 'x^8', 'x^7', 'x^6', 'x^5' , 'x^4', 'x^3', 'x^2', 'x', 'ones']
```

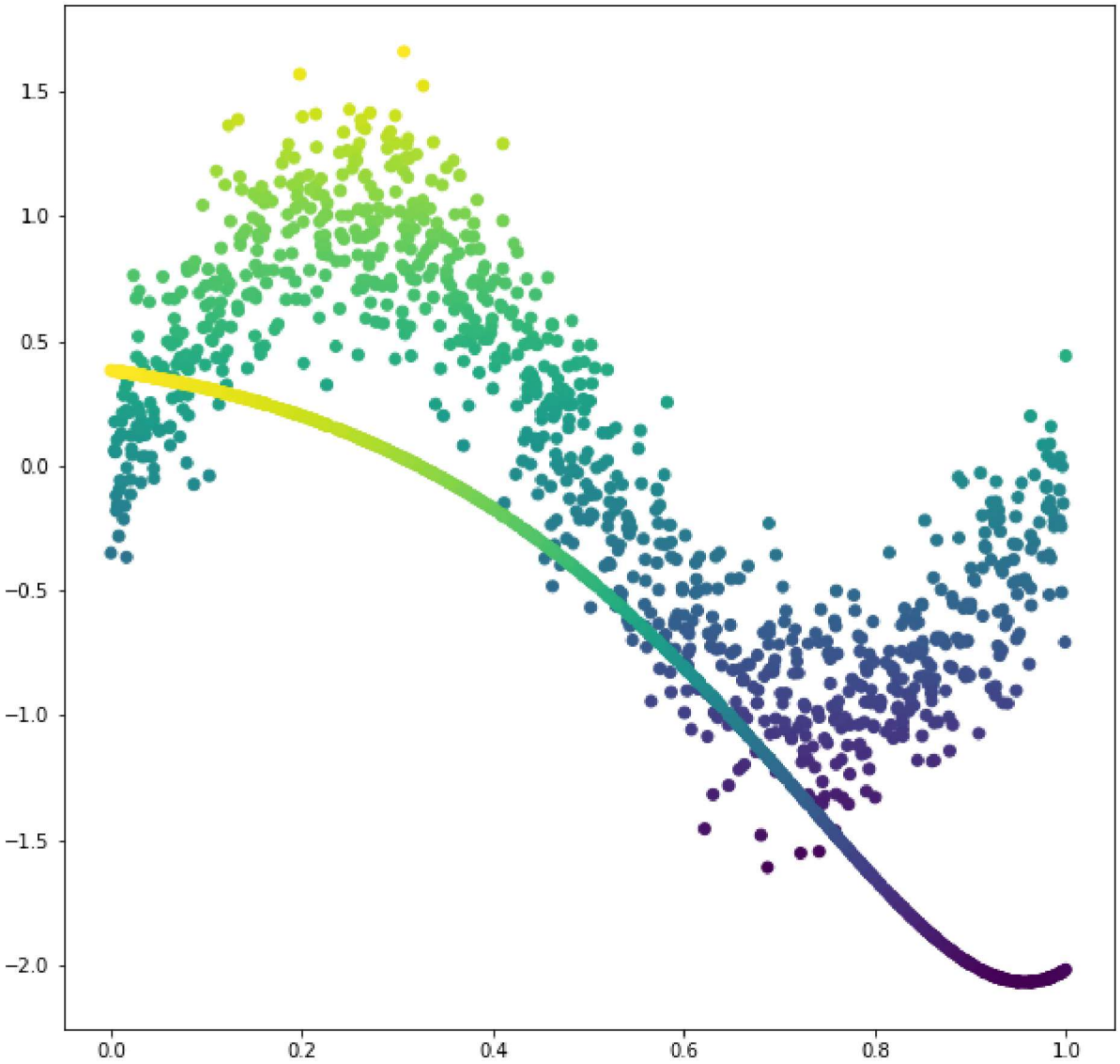
Question 1 -(a) Estimate the L1- norm kernel regression model using the sub gradient descent method and RBF (Gaussian Kernel). Find the best RMSE and MAE by tuning the value of RBF kernel parameter σ and regularization parameter λ. Also obtain the corresponding plot of best estimate.

In [4]:
```
alpha = 0.000982345
lmda = 0.0002
theta = np.array([0 for i in range(10)])

for k in range(100):
  temp1 = np.dot(data_m9, theta)
  temp2 = train_y - temp1
  temp3 = []
  for j, i in enumerate(temp2):
    if i > 0:
      temp3.append(data_m9.iloc[j] * (-1))
    elif i < 0:
      temp3.append(data_m9.iloc[j])
    else:
      temp3.append(random.uniform(-1,1) * data_m9.iloc[j])
  temp3 = pd.DataFrame(temp3)
  temp4 = lmda * theta + temp3.sum()
  theta = theta - alpha * temp4
```

In [5]:
```
plt.figure(figsize = (10,10))
y_hat = np.dot(data_m9, theta)
plt.scatter(train_x, train_y,c=train_y)
plt.scatter(train_x, y_hat,c=y_hat)
```

Out[5]:
```
<matplotlib.collections.PathCollection at 0x7fcf0da37610>
```



In [6]:
```
rmse = math.sqrt(((train_y - y_hat)**2).mean())
rmse
```

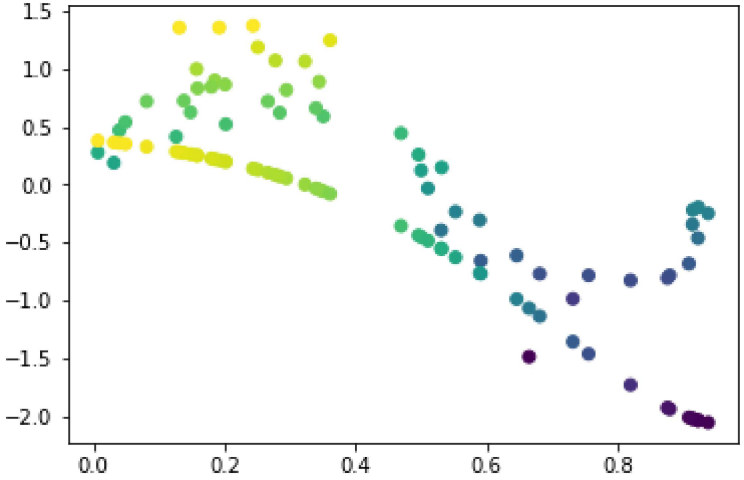
Out[6]:
```
0.8405452791266287
```

In [7]:
```
data_9_test = pd.DataFrame([test_x**9, test_x**8, test_x**7, test_x**6, test_x**5, test_x**4, test_x**3, test_x**2, tes
data_9_test = data_9_test.T
data_9_test.columns = ['x^9', 'x^8', 'x^7', 'x^6', 'x^5' , 'x^4', 'x^3', 'x^2', 'x', 'ones']
```

```
y_hat_test = np.dot(data_9_test, theta)
plt.scatter(test_x, test_y,c=test_y)
plt.scatter(test_x, y_hat_test,c=y_hat_test)

rmse = math.sqrt(((test_y - np.dot(data_9_test, theta)) ** 2).mean())
print(f'RMSE is -> {rmse}')
```

```
RMSE is -> 0.8831437526713702
```



In [8]:
```
print(f'RMSE is -> {rmse}')
```

```
RMSE is -> 0.8831437526713702
```

Kernelised

In [9]:
```
X = np.random.uniform(0.0,1.0,1000)
def my_kernel(X,Y,sigma):
    K = np.zeros((X.shape[0],Y.shape[0]))
    for i,x in enumerate(X):
        for j,y in enumerate(Y):
            K[i,j] = np.exp((-(np.linalg.norm(x-y)**2))/(sigma))
    return K

x = my_kernel(X,X, 2**-8)
```

In [10]:
```
x.shape
```

Out[10]: (1000, 1000)

Question 1-(b) Also, estimate the ε- Support Vector Regression model using the sub gradient descent method and RBF (Gaussian Kernel). Find the best RMSE and MAE by tuning the value of RBF kernel parameter σ, regularization parameter λ and ε . . Find the sparsity of the obtained solution vector α.

In [11]:
```
h_mat = np.zeros([1000,1000])
sigma = 2**-7

def kernel(x1, x2, sigma):
  return np.exp(((-1) / (2 * sigma)) * np.linalg.norm(x1-x2)**2)

for i in range(len(data_m9)):
  if i % 100 == 0:
    print(i)
  for j in range(len(data_m9)):
    h_mat[i,j] = kernel(data_m9.iloc[i], data_m9.iloc[j], sigma)
```

```
0
100
200
300
400
500
600
700
800
900
```

In [12]:
```
lmda = 2**(-1)
alpha = 0.00001
theta = np.array([0 for _ in range(len(data_m9) + 1)])
b = 0
sigma = 2**-8


h_mat = np.column_stack((h_mat, np.ones(1000)))
```

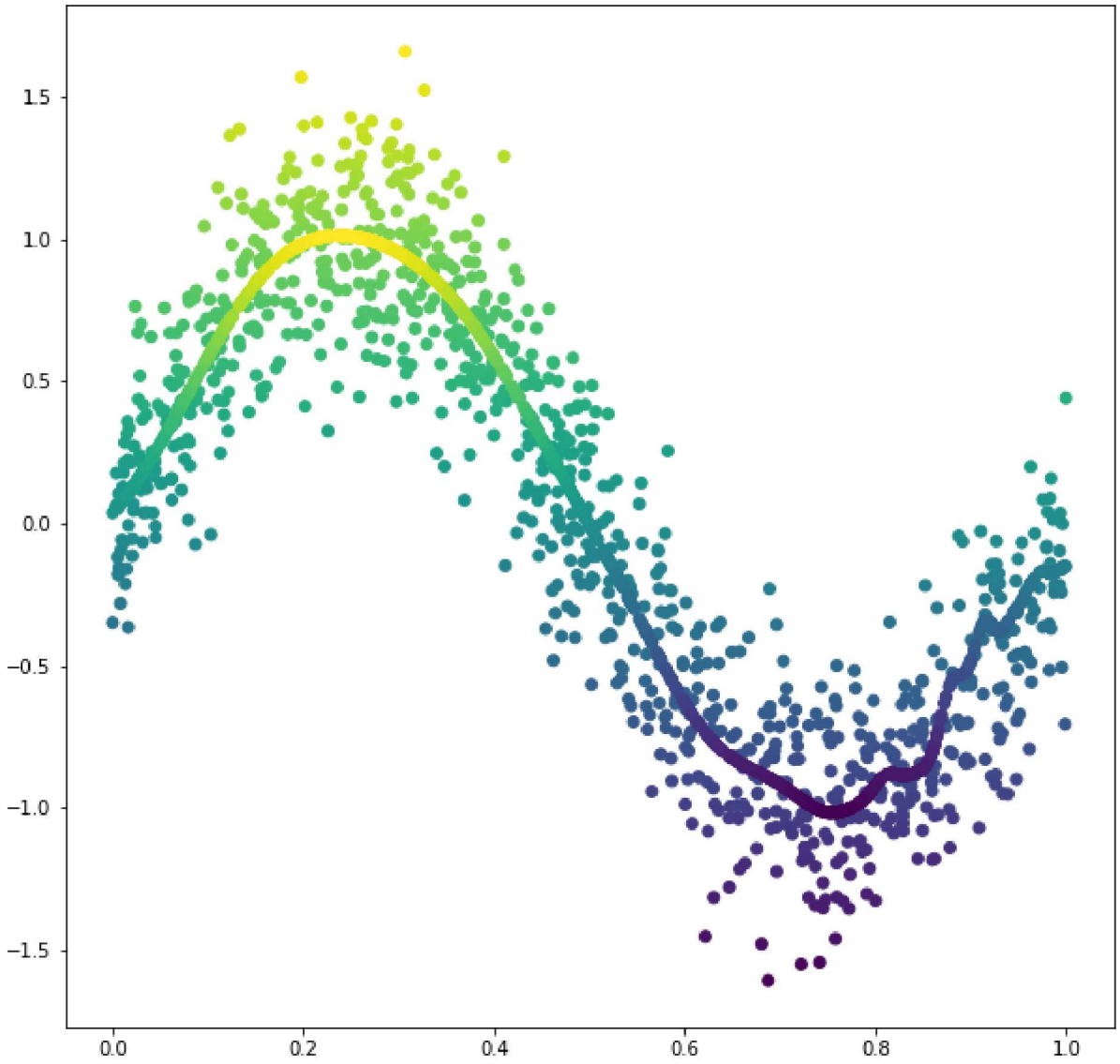
```
for k in range(100):
  sub_grad = (2**(-1)*theta - np.dot(h_mat.T,np.sign(train_y - np.dot(h_mat,theta))))
  theta = theta - sub_grad*alpha
```

In [13]:
```
y_hat = []

for i in range(len(data_m9)):
  y_hat.append(np.dot(h_mat[i].T, theta) + b)

plt.figure(figsize=(10,10))
plt.scatter(train_x, train_y,c=train_y)
plt.scatter(train_x, y_hat,c=y_hat)
```

Out[13]:
```
<matplotlib.collections.PathCollection at 0x7fcf0d9dee90>
```



Question- 2 Consider the motorcycle dataset. Estimate the L1- norm kernel regression model

using RBF kernel (Gaussian Kernel) . Find the best RMSE and MAE using leave- one out by tuning the value of kernel parameter σ and regularization

parameter

λ. Also obtain the corresponding plot of best estimate.

In [14]:
```
motorcycle_data = pd.read_excel('/content/motorcycle.xlsx')
```

In [15]:
```
scaled_x = (motorcycle_data['x'] - motorcycle_data['x'].mean()) / (motorcycle_data['x'].max() - motorcycle_data['x'].mi
# scaled_x = motorcycle_data['x']

data_m9 = pd.DataFrame([scaled_x**9 ,scaled_x**8, scaled_x**7 ,scaled_x**6, scaled_x**5, scaled_x**4, scaled_x**3, scal
data_m9 = data_m9.T
data_m9.columns = ['x^9', 'x^8', 'x^7', 'x^6', 'x^5' , 'x^4', 'x^3', 'x^2', 'x', 'ones']
data_m9
```

Out[15]:

	x^9	x^8	x^7	x^6	x^5	x^4	x^3	x^2	x	ones
0	-0.000347	0.000841	-0.002038	0.004938	-0.011967	0.028999	-0.070272	0.170290	-0.412662	1.0
1	-0.000321	0.000784	-0.001916	0.004684	-0.011450	0.027994	-0.068437	0.167313	-0.409039	1.0
2	-0.000252	0.000632	-0.001587	0.003985	-0.010008	0.025135	-0.063125	0.158539	-0.398169	1.0
3	-0.000213	0.000545	-0.001395	0.003569	-0.009130	0.023354	-0.059741	0.152821	-0.390923	1.0
4	-0.000180	0.000470	-0.001224	0.003190	-0.008314	0.021670	-0.056480	0.147208	-0.383677	1.0
...	...	...	...	...	...	...	...	...	...	...
128	0.002238	0.004409	0.008686	0.017111	0.033708	0.066402	0.130809	0.257686	0.507628	1.0
129	0.003920	0.007256	0.013430	0.024860	0.046017	0.085180	0.157671	0.291855	0.540236	1.0
130	0.003920	0.007256	0.013430	0.024860	0.046017	0.085180	0.157671	0.291855	0.540236	1.0
131	0.004419	0.008072	0.014743	0.026929	0.049187	0.089843	0.164101	0.299737	0.547483	1.0
132	0.008317	0.014161	0.024111	0.041051	0.069894	0.119001	0.202612	0.344966	0.587338	1.0

133 rows × 10 columns

In [16]:
```
lmda = 0.00015
alpha = 0.8
theta = np.array([0 for _ in range(len(data_m9))])
```

```
b = 0
sigma = 2**-8

def kernel(x1, x2, sigma):
  d = (x1 - x2)**2
  d = ((-1) / (2 * sigma**2)) * d
  ans = math.e ** (d.sum())
  return ans

def create_kernel_vector(i):
  temp = []
  for j in range(len(data_m9)):
    temp.append(kernel(data_m9.iloc[i], data_m9.iloc[j], sigma))

  return np.array(temp)

for k in range(50):
  main_vector = []
  b_vector = []
  for i in range(len(data_m9)):
    b_temp = 1
    temp1 = create_kernel_vector(i)
    temp2 = np.dot(temp1, theta.T) + b
    if motorcycle_data['y'].iloc[i] - temp2 > 0:
      temp1 = temp1 * (-1)
      b_temp *= (-1)
    elif motorcycle_data['y'].iloc[i] - temp2 == 0:
      temp1 = np.random.uniform(-1,1) * temp1
      b_temp *= np.random.uniform(-1,1)

    main_vector.append(temp1)
    b_vector.append(b_temp)

  main_vector = pd.DataFrame(main_vector)
  temp = lmda * theta + (1/2) * main_vector.sum()

  theta = theta - alpha * temp
  b = b - alpha * (1/2) * sum(b_vector)
```

In [22]:

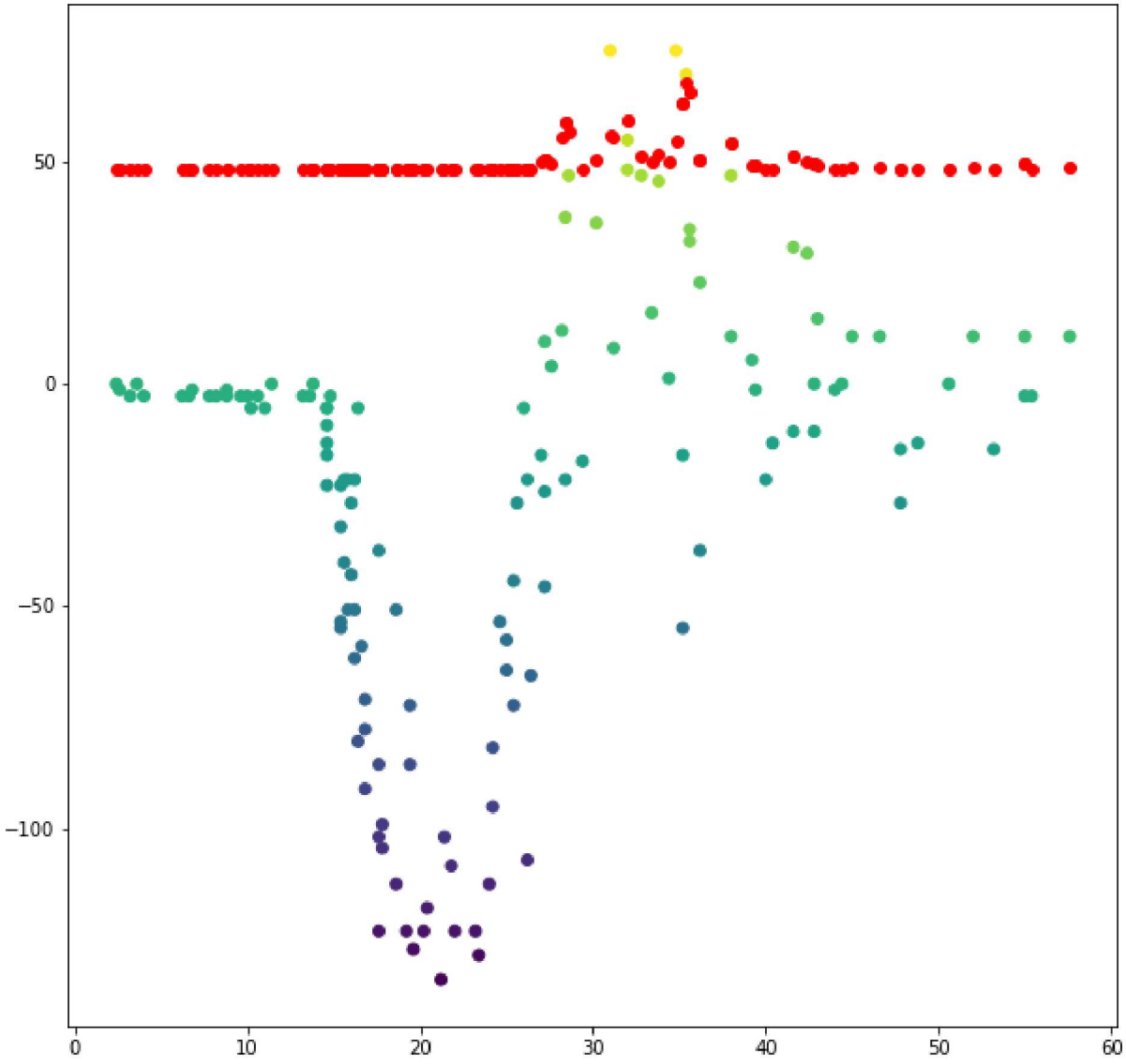
```
y_hat = []

for i in range(len(data_m9)):
```

```
    temp1 = create_kernel_vector(i)
    y_hat.append(np.dot(temp1.T, theta) + b)
```

In [23]:
```
plt.figure(figsize=(10,10))
plt.scatter(motorcycle_data['x'], motorcycle_data['y'],c= motorcycle_data['y'])
plt.scatter(motorcycle_data['x'], y_hat,c='red')
```

Out[23]:
```
<matplotlib.collections.PathCollection at 0x7fcf0abff910>
```



Question -3 Consider the motorcycle dataset. Estimate the ε- Support Vector Regression model using RBF kernel (Gaussian Kernel). Find the best RMSE and MAE using leave-one out by tuning the value of kernel parameter σ, regularization parameter λ and user defined parameter ε. Also obtain the corresponding plot of best estimate. Find the sparsity of the obtained solution vector α.

In [24]:
```
scaled_x = (motorcycle_data['x'] - motorcycle_data['x'].mean()) / (motorcycle_data['x'].max() - motorcycle_data['x'].mi

data_m9 = pd.DataFrame([scaled_x**9 ,scaled_x**8, scaled_x**7 ,scaled_x**6, scaled_x**5, scaled_x**4, scaled_x**3, scal
data_m9 = data_m9.T
data_m9.columns = ['x^9', 'x^8', 'x^7', 'x^6', 'x^5' , 'x^4', 'x^3', 'x^2', 'x', 'ones']
```

In [25]:
```
lmda = 0.0015
alpha = 0.999
theta = np.array([0 for _ in range(len(data_m9))])
b = 0
sigma = 2**-8

def kernel(x1, x2, sigma):
  d = (x1 - x2)**2
  d = ((-1) / (2 * sigma**2)) * d
  ans = math.e ** (d.sum())
  return ans

def create_kernel_vector(i):
  temp = []
  for j in range(len(data_m9)):
    temp.append(kernel(data_m9.iloc[i], data_m9.iloc[j], sigma))

  return np.array(temp)

for k in range(10):
  main_vector = []
  b_vector = []
  for i in range(len(data_m9)):
    b_temp = 1
    temp1 = create_kernel_vector(i)
    temp2 = np.dot(temp1, theta.T) + b
    if motorcycle_data['y'].iloc[i] - temp2 < math.e:
```

```
      temp1 = np.zeros(len(temp1))
      b_temp = 0
    elif motorcycle_data['y'].iloc[i] - temp2 > math.e:
      temp1 = temp1 * (-1)
      b_temp *= (-1)
    elif motorcycle_data['y'].iloc[i] - temp2 == 0:
      temp1 = np.random.uniform(-1,1) * temp1
      b_temp *= np.random.uniform(-1,1)

    main_vector.append(temp1)
    b_vector.append(b_temp)

  main_vector = pd.DataFrame(main_vector)
  temp = lmda * theta + (1/2) * main_vector.sum()

  theta = theta - alpha * temp
  b = b - alpha * (1/2) * sum(b_vector)
```

In [26]:

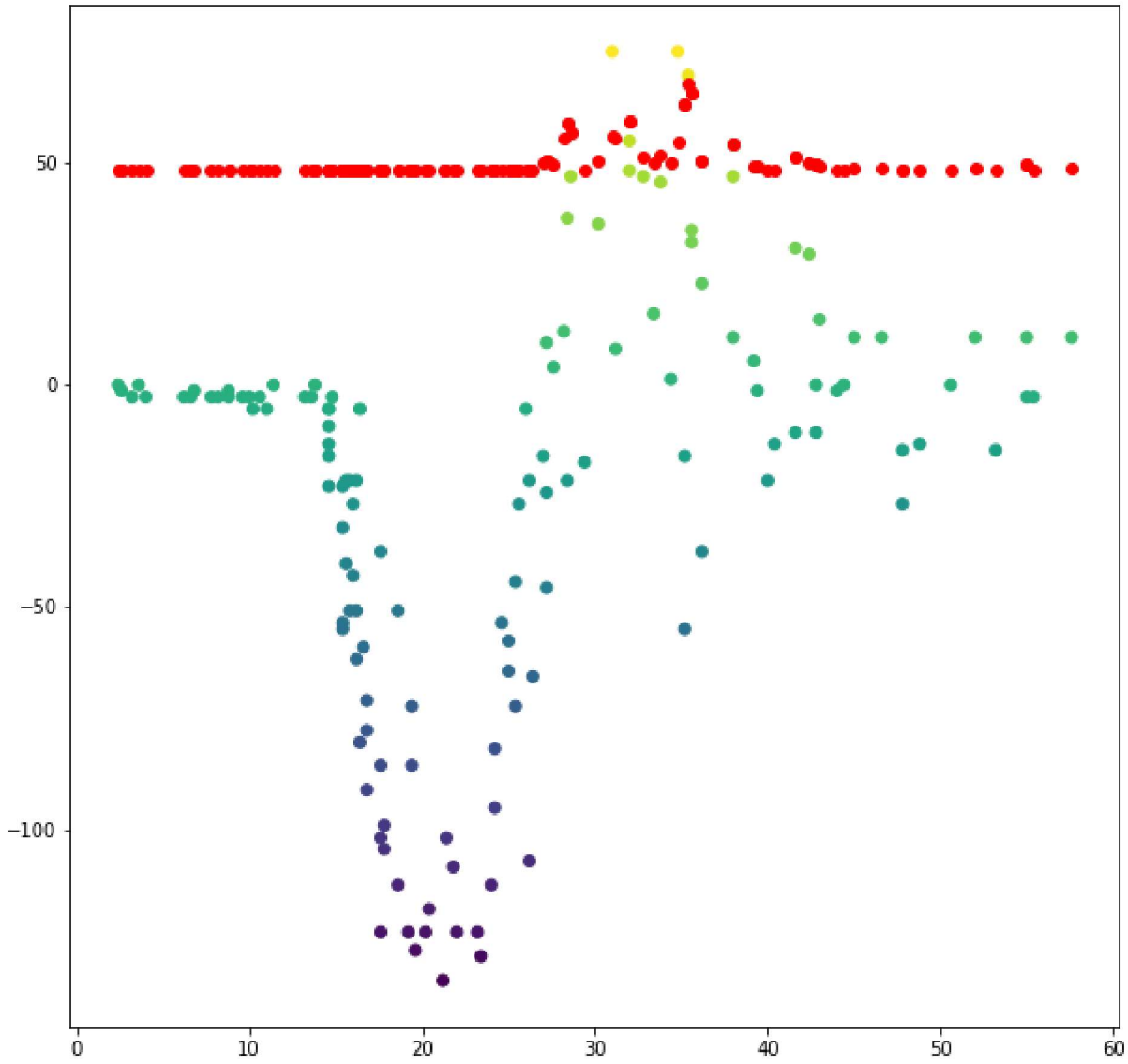
```
y_hat = []

for i in range(len(data_m9)):
  temp1 = create_kernel_vector(i)
  y_hat.append(np.dot(temp1.T, theta) + b)

plt.figure(figsize=(10,10))
plt.scatter(motorcycle_data['x'], motorcycle_data['y'],c= motorcycle_data['y'])
plt.scatter(motorcycle_data['x'], y_hat,c='red')
```

Out[26]:

```
<matplotlib.collections.PathCollection at 0x7fcf0aae8510>
```



In [21]: