Multiple Outputs

***Abstract*—This document contains theory involved in curve fitting.**

1 Objective

The objective is first to generate a sinosuidal dataset and then to implement linear regression for multiple outputs.

2 Generate Dataset

Create a sinusoidal function of the form

$$y = A \sin 2\pi x + n(t) \tag{2.0.1}$$

n(t) is the random noise that is included in the training set. This set consists of N samples of input data i.e. x expressed as shown below

$$x = \left(x_1, x_2, .., x_N\right)^T \tag{2.0.2}$$

which give the corresponding values of y denoted as

$$y = \left(y_1, y_2, .., y_N\right)^T \tag{2.0.3}$$

The Fig 0 is generated by random values of x_n

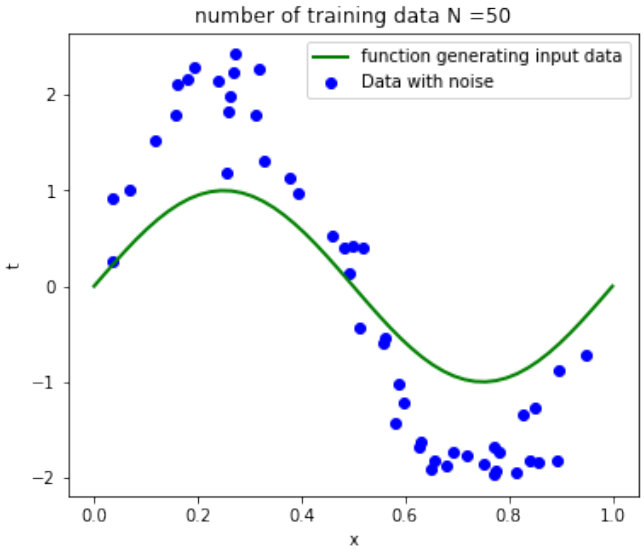

Fig. 0: Sinusoidal Dataset with added noise

for n =1,2,..,N. where N=50 in the range [0,1]. The corresponding values of y were generated from the Eq (2.0.1).The first term $A \sin 2\pi x$ is computed directly and then random noise samples having a normal(Gaussian) distribution are added inorder to get the corresponding values of y.

```
#Generate the sine curve
import numpy as np
import matplotlib.pyplot as plt
N = 50
np.random.seed(20)
x = np.sort(np.random.rand(N,1),axis=0)
noise = np.random.normal(0,0.3,size=(N,1))
A = 2.5
y = A*np.sin(2*np.pi*x) + noise
plt.scatter(x,y,c='b',marker='o',label='Data with
    noise')
plt.xlabel('x');plt.ylabel('y')
```

The following code generates the input matrix F

```
sk_poly_deg=3
poly_feature = PolynomialFeatures(degree=
    sk_poly_deg,include_bias=False)
F = poly_feature.fit_transform(x)
```

The generated matrix would look like

$$\mathbf{F} = \begin{pmatrix} 1 & x_0 & x_0^2 & \dots & x_0^{N-1} \\ 1 & x_1 & x_1^2 & \dots & x_1^{N-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{N-1} \\ \vdots & & \vdots & & \vdots \\ 1 & \dots & \dots & \dots & x_N^{N-1} \end{pmatrix} \tag{2.0.4}$$

3 Multiple Outputs

In single output Linear regression we only considered target variable t,But in some cases for multiple outputs we can consider more number of target variables.This can be done by using same basis function to model all components of the target vector.

$$y(w, x) = \mathbf{W}^T \phi(x) \tag{3.0.1}$$

$$\tag{3.0.2}$$

Where y is a column vector of dimension K,$\mathbf{W}$ is a M × K matrix of parameters and $\phi(x)$ is a M

dimensional column vector . We can maximize the gaussion function by maximum likelihood function and we can do this by considering :

$$\hat{\mathbf{W}} = \left(\phi^T \phi\right)^{-1} \phi^T \mathbf{T} \tag{3.0.3}$$

if we examine the result for each target variable we have :

$$\mathbf{w_k} = \left(\phi^T \phi\right)^{-1} \phi^T \mathbf{t_k} = \phi^{-1} \mathbf{t_k} \tag{3.0.4}$$

we only need to compute a pseudo inverse matrix ϕ^* which is shared by all the vectors w_k.This was the theory behind multiple outputs.
Now for the implementation purpose the sinusoidal dataset we generated earlier can be easily generated through a direct function of a library provided in python. By using the direct function we are now creating that sinusoidal dataset also considering the target variables t, here we are taking n_target=2.Finally after performing the linear regression operation on the data set it will show us two outputs as we considered target variables to be 2.

```
plt.plot(np.linspace(0,1,50),np.sin(2*np.linspace
    (0,1,50)*np.pi),c='g',linewidth=2,label='
    function generating input data')
```

Now, we import make_regression from sklearn

```
from sklearn.datasets import make_regression
from sklearn.linear_model import
    LinearRegression
```

Creating datasets of sinosuidal form considering target variables=2

```
X, y = make_regression(n_samples=50,
    n_features=10, n_informative=5, n_targets
    =2, random_state=1, noise=0.3)
model = LinearRegression()
model.fit(X, y)
```

Plot the predicted output

```
plt.plot(x,model.predict(X),'r',label='Plot after
    LinearRegression for two targets that is
    Multioutput ')
plt.legend()
plt.show()
```

The plot would look like Fig 0 Python code:

```
https://github.com/sahilsin/EE_IDP/blob/main/
    Assignment_3/mp.ipynb
```

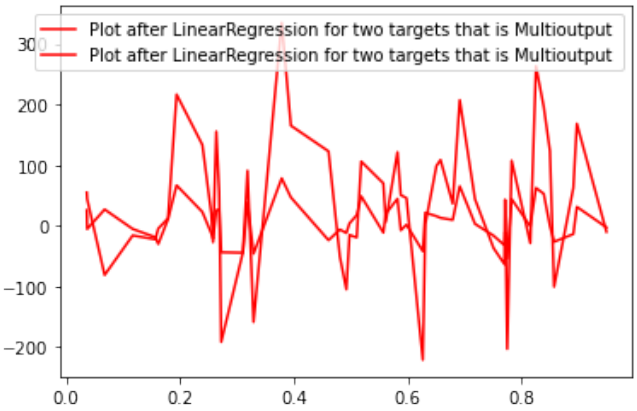

Fig. 0: Linear regression for multiple outputs