Polynomial Curve Fitting

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

1 Objective

The objective is to fit best line for the polynomial curve using regularization.

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 for

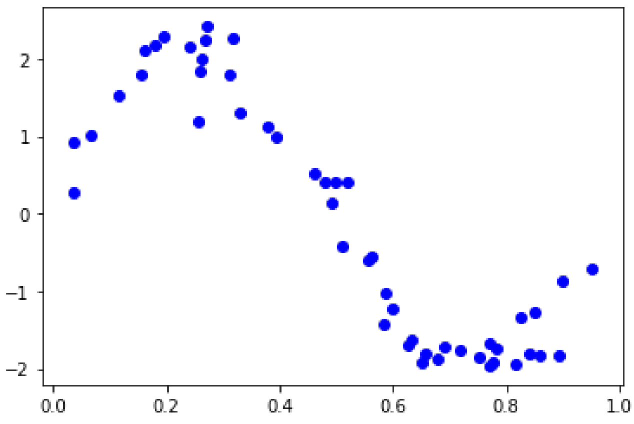

Fig. 0: Sinusoidal Dataset with added noise

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 Polynomial Curve Fitting

The goal is to find the best line that fits into the pattern of the training data shown in the graph. We shall fit the data using a polynomial function of the form,

$$y\left(w, x\right) = \sum_{j=0}^{M} w_j x^j \tag{3.0.1}$$

$$\tag{3.0.2}$$

M is the order of the polynomial The polynomial coefficient are collectively denoted by the vector

$\mathbf{w}$.The proposed vector $\mathbf{w}$ of the model referring to Eq (2.0.4) is given by

$$\hat{\mathbf{w}} = \left(\mathbf{F}^T\mathbf{F}\right)^{-1}\mathbf{F}^T y \tag{3.0.3}$$

4 Bias- Variance Tradeoff

In decision theory, the decision stage consists of choosing a specific estimate $y(\mathbf{x})$ for the target t for each input $\mathbf{x}$

This can be done by using a loss function $L(t, y(\mathbf{x}))$ so the expected loss is

$$E[L] = \int\int L(t, y(\mathbf{x}))p(\mathbf{x}, t)d\mathbf{x}dt \tag{4.0.1}$$

A common loss function is the squared loss function given by

$$L(t, y(\mathbf{x})) = (y(\mathbf{x}) - t)^2 \tag{4.0.2}$$

The expected loss for Eq (4.0.2),

$$E[L] = \int\int (y(\mathbf{x}) - t)^2 p(\mathbf{x}, t)d\mathbf{x}dt \tag{4.0.3}$$

The optimal prediction for the squared loss function is given by the conditional expectation h(x)

$$h(x) = E[t|\mathbf{x}] = \int tp(t|\mathbf{x})dt \tag{4.0.4}$$

where $p(t|\mathbf{x})$ is the conditional distribution

The expected square loss takes the form

$$E[L] = \int (y(\mathbf{x}) - h(\mathbf{x}))^2 p(\mathbf{x})d\mathbf{x} + \int\int (h(\mathbf{x}) - t)^2 p(\mathbf{x}, t)d\mathbf{x}dt \tag{4.0.5}$$

In Eq (4.0.5), Second term is due to the noise in the data.

Consider the integrand of the first term in Eq (4.0.5), which becomes

$$(y(\mathbf{x}; D) - h(\mathbf{x}))^2 \tag{4.0.6}$$

for D data sets.

Adding and subtracting $E[y(\mathbf{x}; D)]$ to Eq (4.0.6)

$$(y(\mathbf{x}; D) - E[y(\mathbf{x}; D)] + E[y(\mathbf{x}; D)] - h(\mathbf{x}))^2 \tag{4.0.7}$$

Expanding,

$$\begin{aligned}(y(\mathbf{x}; D) - E[y(\mathbf{x}; D)] + E[y(\mathbf{x}; D)] - h(\mathbf{x}))^2 = \\ (y(\mathbf{x}; D) - E[y(\mathbf{x}; D)])^2 + (E[y(\mathbf{x}; D)] - h(\mathbf{x}))^2 + \\ 2\left(y(\mathbf{x}; D) - E[y(\mathbf{x}; D)]\right)\left(E[y(\mathbf{x}; D)] - h(\mathbf{x}))^2\right)\end{aligned} \tag{4.0.8}$$

Take the expectation w.r.t D,

$$\begin{aligned}E[(y(\mathbf{x}; D) - h(\mathbf{x}))^2] = (E[y(\mathbf{x}; D)] - h(\mathbf{x}))^2 \\ + E_D[(y(\mathbf{x}; D) - E_D[y(\mathbf{x}; D)])^2]\end{aligned} \tag{4.0.9}$$

In Eq (4.0.9),

First term - $(bias)^2$

Second term - *variance*

Now substituting this expanded eq in Eq (4.0.5), $expectedloss = (bias)^2 + variance + noise$

$$(bias)^2 = \int (E[y(\mathbf{x}; D)] - h(\mathbf{x}))^2 p(\mathbf{x})d\mathbf{x} \tag{4.0.10}$$

$$variance = \int E_D[(y(\mathbf{x}; D) - E_D[y(\mathbf{x}; D)])^2]p(\mathbf{x})d\mathbf{x} \tag{4.0.11}$$

$$noise = \int (h(\mathbf{x}) - t)^2 p(\mathbf{x}, t)d\mathbf{x}dt \tag{4.0.12}$$

The ultimate goal is to minimize the expected loss which we have decomposed into $(bias)^2$, *variance* and constant noise term.

5 Example

We generate 100 datasets, each containing 50 data points independently from the sinusoidal curve $h(x) = sin(2\pi x)$.

For each dataset, we fit the model by using regularization.

large λ, low variance but high bias and

small λ, low bias but high variance.

We have to choose to λ value such that the value of $(bias)^2 + variance$ is minimum.

Plots for different values of λ.

The average prediction is estimated from

$$\bar{y}(\mathbf{x}) = \frac{1}{L}\sum_{l=1}^{L} y^{(l)}(\mathbf{x}) \tag{5.0.1}$$

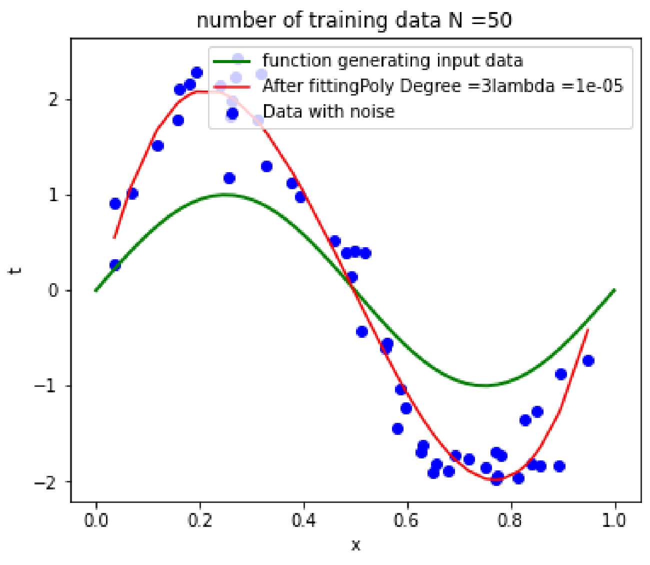

Fig. 0

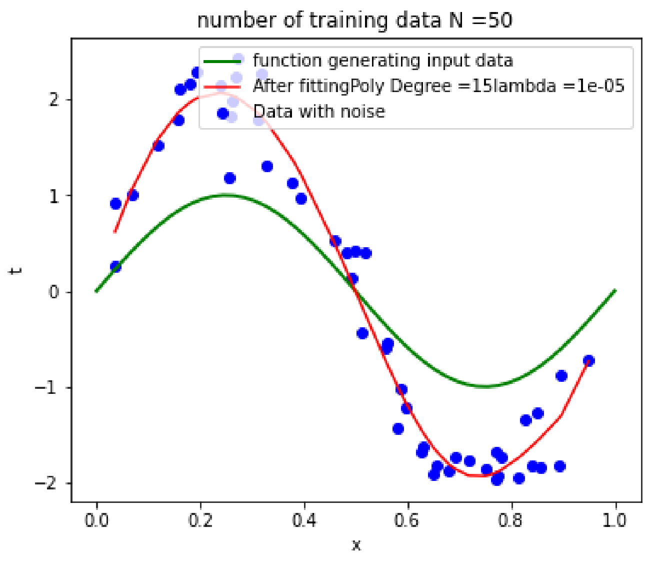

Fig. 0

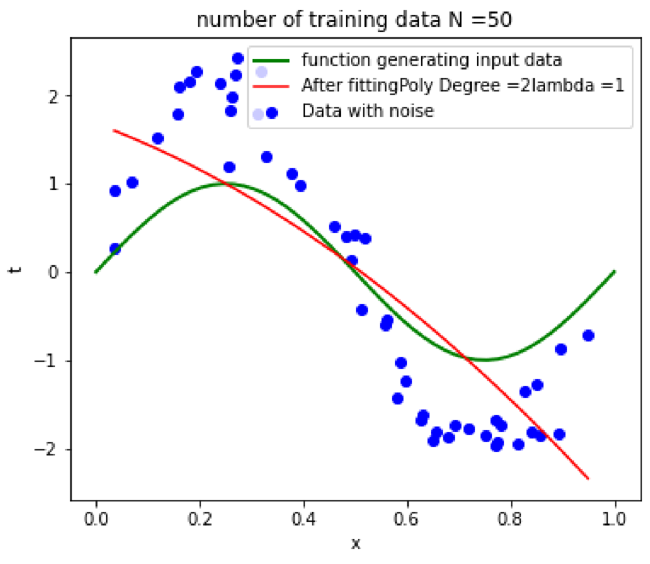

Fig. 0

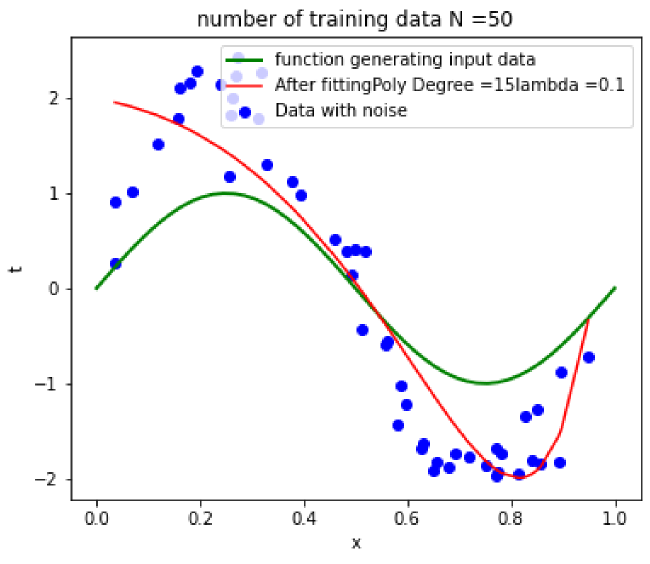

Fig. 0

and the integrated $(bias)^2$ and variance is given by

$$(bias)^2 = \frac{1}{N}\sum_{n=1}^{N}(\bar{y}(x_n) - h(x_n))^2 \quad (5.0.2)$$

$$variance = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{L}\sum_{l=1}^{L}\left(y^{(l)}(x_n) - \bar{y}(x_n)\right) \quad (5.0.3)$$

The model with optimal predictive capability is the one with best balance.

Python code:

```
https://github.com/Hrithikraj2/EE4015_IDP/blob/
main/Assignment_5/Assignment_5.ipynb
```