CS554 Introduction to Machine Learning Polynomial Regression Report

Mert Erkol

March 15, 2023

1 Introduction

In this study multi degree polynomial regression models fitted from degree 0 to 6 by given data and calculated errors. According to error plots, the best performing model is decided and explained why.

2 Background

The goal is to fit a model to given data that generalizes the problem and predicts next outcomes when new input is given. To be able to fit the model correctly polynomial equations have been used. Degree of 2 polynomial equation can seen below in 1. NumPy library in Python has a function called polyfit to fit the given data (X,R) with specific degree of polynomial with minimum sum of squared error can seen in equation 2.

$$y = w_0 x^2 + w_1 x + b, \tag{1}$$

where y is the function that I will fit, w_0, w_1 and b are the coefficients that will be settled in polyfit function and x is the input of the function [1].

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \tag{2}$$

where n is the number of data points, y_i is the actual value of the i-th data point, $\hat{y}_i$ is the predicted value of the i-th data point, $f(x_i)$ is the predicted value of y_i based on the model, and the sum is taken over all n data points [2].

3 Methodology

To complete the project Python@3.9 has been used as a programming language and two libraries: NumPy and Matplotlib. The structure is follows, Read the train and test data, fit the polynomial regression models using numpy, evaluate them on test data by calculating sum of squared errors. At last the errors plotted as a function of degree and every fit visualized on the training set.

4 Results

In the figures below each degree of polynomial models fitted on the training data can be seen.

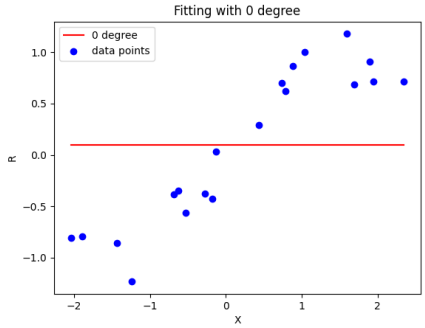

Figure 1: Polynomial fitted with degree of 0

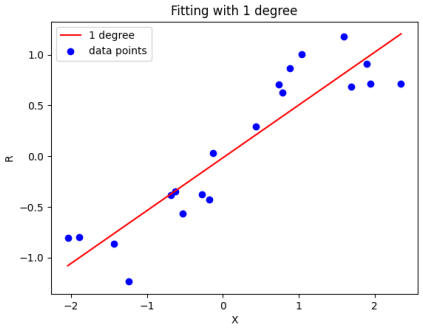

Figure 2: Polynomial fitted with degree of 1

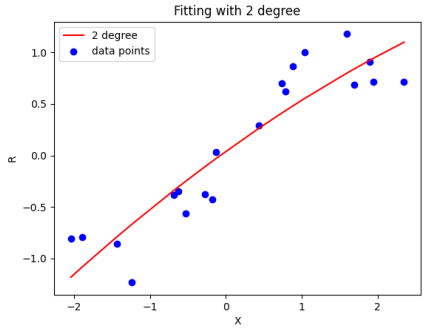

Figure 3: Polynomial fitted with degree of 2

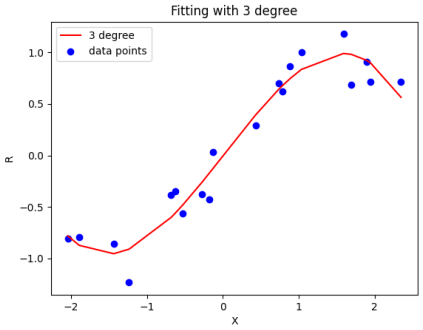

Figure 4: Polynomial fitted with degree of 3

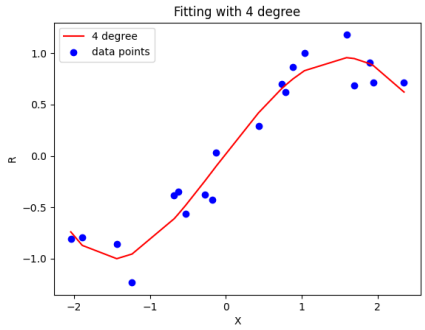

Figure 5: Polynomial fitted with degree of 4

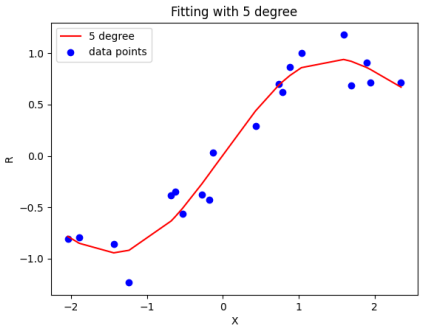

Figure 6: Polynomial fitted with degree of 5

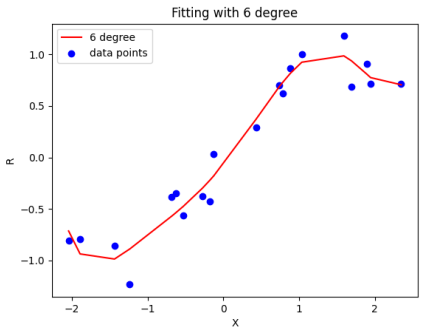

Figure 7: Polynomial fitted with degree of 6

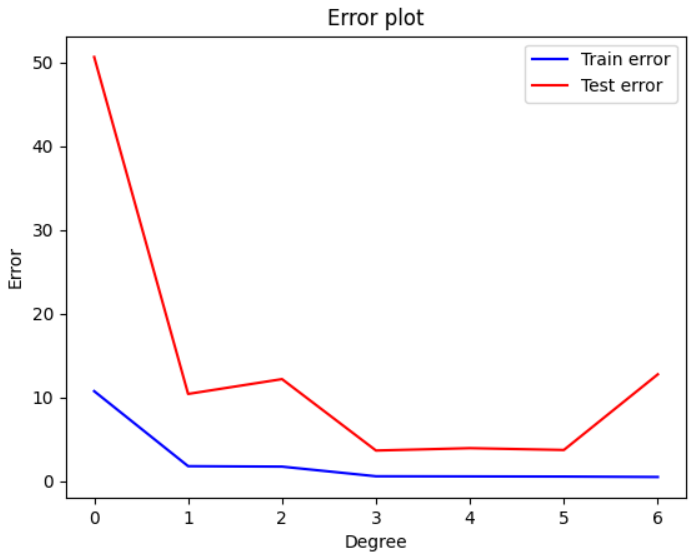

Figure 8: Train and Test errors plotted as a function of polynomial degree

5 Discussion

According the results section it is possible to find an optimum degree for the model within the given dataset. To be able to decide the optimum degree for the polynomial fit it is not enough to just see the drawn plots on the training set figs. 1 to 7. SSE needs to be calculated at each dataset and models with lowest errors should be evaluated to select the most suitable model. As clearly seen in the fig. 8 SSE on each dataset seems to be decreasing but not in every increase in the polynomial degree. Inside the trained models, third degree model is the best performing model because it has one of the lowest test error compared to other models. The test errors in fourth and fifth polynomial models are also very low but they are not suitable models because the model should also has the power to generalize the problem. They have very similar error values so there is no need to increase the complexity of the model. After degree of six test error increases slightly that indicates the model started to overfit the dataset even training error stays the same or decreasing. To solve the overfitting problem training samples may be increased in the training set.

6 Conclusion

In this study several different degree of polynomial models have been fitted to the dataset and evaluated with error functions and multiple datasets. This study indicates and importance of finding the sweet spot between underfitting and overfitting region and select the best model that generalizes the problem.

7 Appendix

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

# function to fit and predict the model
def fit_predict(X, Y, degree):
    z = np.polyfit(X, Y, degree)
    y_hat = np.polyval(z, X)
    return z, y_hat

# function to predict the model
def predict(model, X):
    y_hat = np.polyval(model, X)
    return y_hat

# function to plot the model
def plot(y_hat, X, Y, ax, degree):
    ax.plot(X, y_hat, color="r", label=str(degree) + " degree")
    ax.scatter(X, Y, color="b", label="data points")
    ax.set_title("Fitting with " + str(degree) + " degree")
    ax.set_xlabel("X")
    ax.set_ylabel("R")


# function to create a figure
def create_fig():
    fig = plt.figure()
    ax = fig.subplots()
    return ax


# function to plot errors on each degree
def error_plot(errors, degrees, ax, dataset):
    if dataset == "Train":
        ax.plot(degrees, errors, color="b", label=dataset + " error")
    else:
        ax.plot(degrees, errors, color="r", label=dataset + " error")

    ax.set_title("Error plot")
    ax.set_xlabel("Degree")
    ax.set_ylabel("Error")

# function to calculate SSE
def calculate_SSE(y_hat, y):
    errors = []
    for i in range(len(y_hat)):
        errors.append(np.sum((y_hat[i] - y) ** 2))
    return errors


def main():
    # load the data
    train_data = np.loadtxt(
        "data/train.csv", delimiter=",", skiprows=1, dtype=np.float32
    )
```

```
    test_data = np.loadtxt("data/test.csv", delimiter=",",
                           skiprows=1, dtype=np.float32)
    # sort the data
    train_data = train_data[train_data[:, 0].argsort()]
    test_data = test_data[test_data[:, 0].argsort()]

    train_x, train_y = train_data[:, 0], train_data[:, 1]
    test_x, test_y = test_data[:, 0], test_data[:, 1]

    # list of degrees
    degrees = [0, 1, 2, 3, 4, 5, 6]

    # list of figures
    figures = []

    # create the figures for each degree
    for i in range(len(degrees)):
        figures.append(create_fig())

    # error figure
    error_fig = create_fig()

    # list of results
    train_y_hats = []
    test_y_hats = []

    # Fitting
    for degree in degrees:
        # Fitting and predicting
        model, train_y_hat = fit_predict(train_x, train_y, degree)
        train_y_hats.append(train_y_hat)

        # predicting on test data
        test_y_hat = predict(model, test_x)
        test_y_hats.append(test_y_hat)

    # Calculating and plotting train errors for each degree
    train_errors = calculate_SSE(train_y_hats, train_y)
    error_plot(train_errors, degrees, error_fig, "Train")

    # Calculating and plotting test errors for each degree
    test_errors = calculate_SSE(test_y_hats, test_y)
    error_plot(test_errors, degrees, error_fig, "Test")

    # Plotting the models separately
    for i in range(len(degrees)):
        plot(train_y_hats[i], train_x,
             train_y, figures[i], degree=i)
        figures[i].legend()

    error_fig.legend()
    plt.show()


if __name__ == "__main__":
    main()
```

References

[1] Wikipedia. *Polynomial Regression — Wikipedia, The Free Encyclopedia*. 2023. URL: https://en.wikipedia.org/wiki/Polynomial_regression.

[2] Wikipedia. *Residual sum of squares — Wikipedia, The Free Encyclopedia*. 2023. URL: https://en.wikipedia.org/wiki/Residual_sum_of_squares.