University of Michigan – Dearborn
Department of Computer and Information Science
CIS 481: Computational Learning
(Winter 2023)
Instructor: Luis E. Ortiz

Project 1 Report: Polynomial Curve-Fitting Regression for Working-Age Data

Jordan S-H

[0] Overview

The aim of this project is to implement all components necessary to perform generalized linear regression on the data found in the '**data/**' directory.

The implementations for all models and transformers can be found in the '**src/impls.py**' file, and the usage of implementations is done in the '**src/main.py**' file which can be run with:

```
$ python3 ./main.py
```

Note that the main file can be run from any location as it can locate the positions of supporting files.

All figures used in this document and data collected from testing the model are also stored in the '**out/**' directory with names indicative of their content.

The model to be tested (in '**src/main.py**') is given as follows:

```
model = OutputScalingWrapper(
    Pipeline(
        StandardScaler(),
        PolynomialFeatures(degree=d),
        StandardScaler(),
        LinearRegressor()
    )
)
```

A rough outline of how the model seeks to perform polynomial curve fitting is given as follows:

1. The target vector is scaled prior to fitting the model (for prediction accuracy).
2. The input vector is scaled and transformed with Polynomial Features.
3. The transformed input vector is scaled to create interpretable weights.
4. A linear regressor is fit using the transformed input and target vectors.
5. When creating predictions, the model applies the weights found when predicting to the new transformed input (using the same transformations as the training data) and un-scales the resulting vector (using the same transformations as the training target vector).

[1] 6-Fold Cross Validation for varying degrees

We'll be using 6-fold cross validation to select an optimal degree (d*) by evaluating the average RMSE[†] across each fold for each d ∈ [0, 12]. The splits made are done in order (fold 1 validation set uses indices [0, 7), fold 2 uses indices [7, 14), ...) without shuffling the data as was specified in the project description.

The data for each degree and its associated average RMSE can be found in '**out/cv_errors_data.dat**', and the error curve to select d* is given below:

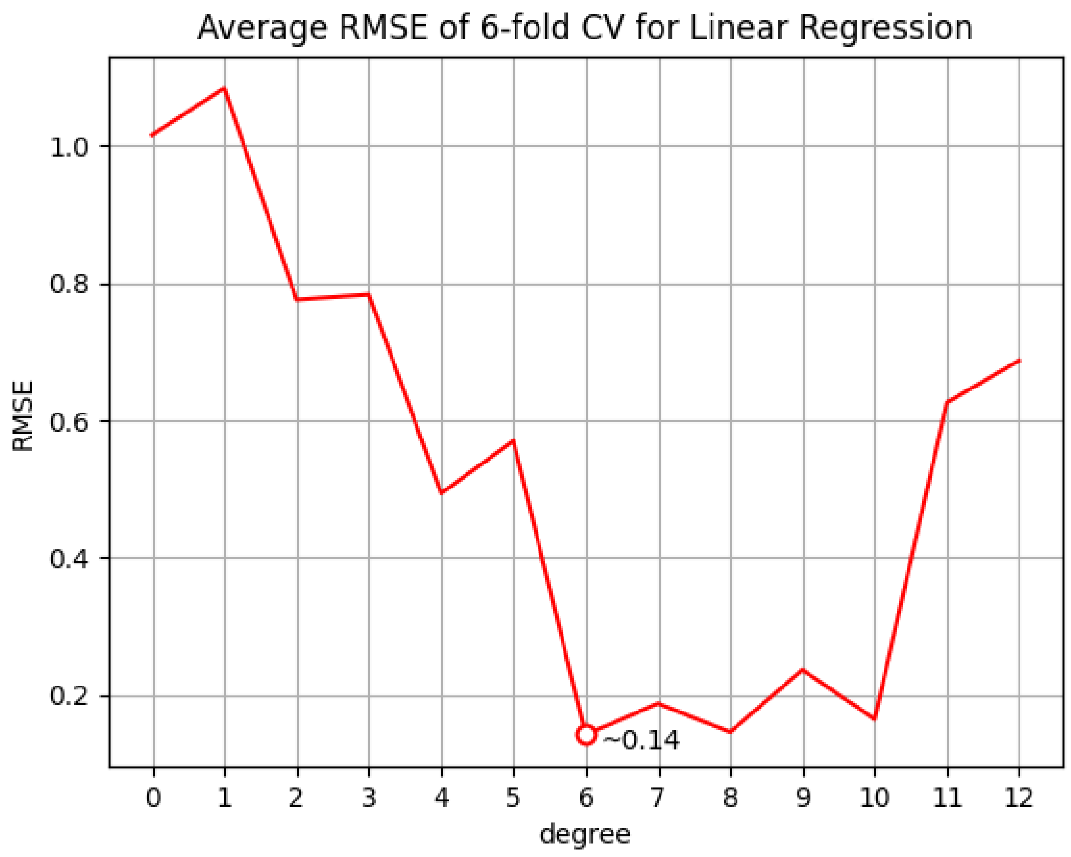

Figure 1.1

[2] Optimal degree of model - d*

Figure 1.1 shows the error decline as model complexity increases, until it begins to rise again at higher degrees (which suggests overfitting). Also shown in the figure is the degree with minimal error, which is degree 6 with an error of roughly 0.14. Degree 6 will be used for d* moving forward.

[†] See appendix 8.1 for details

[3] Weights obtained using d*

The weights obtained fitting a model with degree 6 on all training data are stored in '**out/d6_weights.dat**', and the formula for predicting output values (with rounded weights, and assuming $x \in R$) is given below:

$$f(x) \approx -0.113 + 0.357x + 3.370x^2 + 0.136x^3 - 2.946x^4 + 0.008x^5 + 0.538x^6$$

Since the training data was scaled after transforming with polynomial features, the generated weights be compared relatively[†]. We can see that the weights for the x^2 and x^4 have a larger effect on the output, while the weight for x^5 has a very small effect.

[4] Scores of the selected model

When evaluating the model with d* on the training and testing datasets, the following RMSE values are produced (these values can be seen when running '**src/main.py**'):

Training RMSE: 0.10540106673270466

Testing RMSE: 0.11432570919500114

While the testing error is still greater than the training error (which is to be expected), it's surprisingly close[‡] (which was unexpected).

[†] I am not entirely confident in this statement 😊.

[‡] See addendum 7.1 for details and further analysis.

[5] Prediction Curve

Below is the graph of the model fit using all training data transformed with degree 6 polynomial features superimposed on all data in the project.

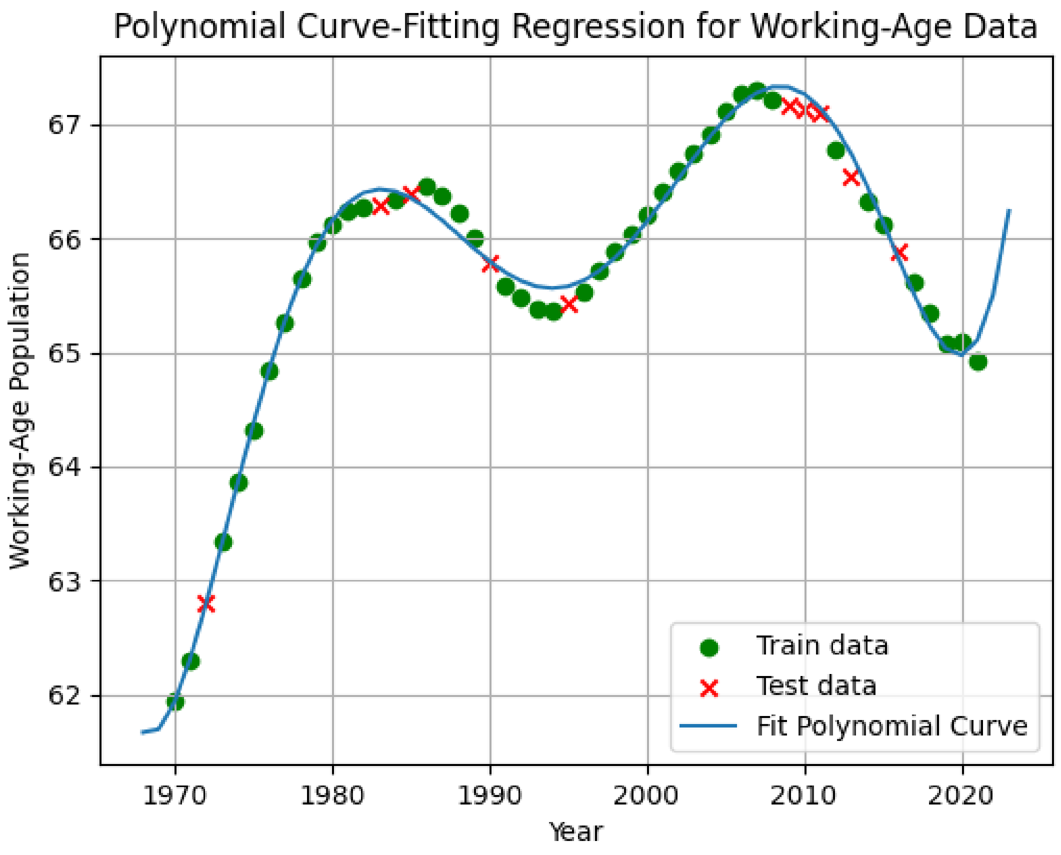

Figure 5.1

We can see in figure 5.1 that the model is able to fit a polynomial curve well to the data. There are places in the curve where it may have benefitted from a slightly higher degree (namely the peaks of the data). This can be supported by figure 1.1 where degree 8 is comparable in accuracy to degree 6 in cross validation, and further exploration could be conducted to test that claim[†].

[†] See addendum 7.2 for details

[6] Reflection

I'm happy with the curve my program was able to fit as seen in figure 5.1. The results discussed in section 4 also indicate that the program was able to generalize (roughly) to the training data with how low and close the training and testing losses were, though this could be due to the size of the dataset. I might want to try this model on a larger dataset to see if my program can perform as well on other forms of data and to see if my implementation of polynomial features works on data with more than one feature[†].

I would also like to try testing a ridge regression model, as linear regression is equivalent to ridge regression parameterized by $\lambda=0$, meaning the results might be improved with different values of λ[‡].

[†] See addendum 7.3 for further analysis on datasets, and appendix 8.2 for details on the implementation of polynomial feature creation.

[‡] See addendum 7.4 for details and further analysis.

[7] Addendum

** Code for this section can be found in '**src/addendum.py**' which is run similarly to '**src/main.py**'*

7.1 – Test error across degrees

# TODO: test err on cv graph

7.2 – Changes in architecture

# TODO: remove initial scaling

7.3 – Larger datasets

# TODO: this

7.4 – Ridge regression

# TODO: this

[8] Appendix

8.1 – RMSE implementation

# TODO: this

8.2 – Polynomial features implementation

# TODO: this