University of Nebraska Lincoln

Introduction to Machine Learning
CSCE 478/878

Programming Assignment 2

Fall 2020

Linear Regression

Part A (Model Code): 478 (68 pts) & 878 (78 pts)
Part B (Data Processing): 478 & 878 (7 pts)
Part C (Model Evaluation): 478 (25 pts) & 878 (35 pts)
Part D (Written Report): 478 & 878 (25 pts)
Extra credit (BONUS) tasks for both 478 & 878: 30 pts

Total: 478 (125 pts) & 878 (145 pts)

Last Name 1: Adamec

First Name 1: Matous

NUID 1: 5001857

Last Name 2:

First Name 2:

NUID 2:

Last Name 3:

First Name 3:

NUID 3:

Obtained Score:

Evaluation of Regression Models On Wine Quality Classification Data

Matous Adamec
University of Nebraska-Lincoln
Email: matous.adamec@huskers.unl.edu
Date: October 6, 2020

***Abstract*—We investigate the usage of regression algorithms on a wine quality dataset. We evaluate the performance of linear regression with and without regularizers, including the L1 and L2 norms, and we investigate polynomial regression up to the fifth degree. After tuning hyperparameters, we propose that the $\lambda = 0, \eta = 0.1$, L2 model works best for batch gradient descent and the $\lambda = 0.001, \eta = 0.01$, L2 model works best for stochastic gradient descent. We also find that the third-degree polynomial model appears to be the best fit, and batch gradient descent generally performs better.**

1. Introduction

Linear regression involves the prediction of a target variable based on values for a set of feature variables. The relationship between these two is unknown, but be approximated by a linear (or higher-polynomial) curve, where each feature is weighted by a θ array. The values of these weights are also unknown, but can be empirically derived through basic linear algebra. The general method for this is referred to as gradient descent, and, in particular, we will employ and optimize two such algorithms in this paper: batch gradient descent and stochastic gradient descent.

2. Data Summary

The wine data is presented as a set of eleven features and one target variable. The basic distributions of these variables are shown in Table 1. The data was standardized to unit variance about the mean. This is an important step, because we have relatively high-dimensional data (eleven features), and differences in variance between the dimensions could make certain dimensions have more weight in the regression model. This would skew data and lead to poorer results, whereas unit variance will result in all dimensions being considered equally, and, thus, better predictions.

The data was then evaluated for correlations. The graph of correlations can be found in Appendix Fig. 1. Briefly, it was decided that citric acid, fixed acidity, free sulfur dioxide, and density could be dropped, as they all had high correlations ($|r| > 0.5$) between each other, or with another feature. This left us with seven features, but the reintroduction of a bias term (equal to 1 for all rows) to define the y-intercept of our loss function brings us back up to a nominal eight.

3. Methods

In approaching the linear regression problem, there are several options for possible solutions. The so-called closed-form solution is the most direct answer to our problem, but is computationally inefficient, as it requires complex array operations, including array inversion, which in and of itself is O(n^3). Because we have a large number of features and rows, the closed-form solution is not ideal, and the alternative iterative approach of gradient descent is preferred.

Gradient descent functions by defining a cost function. It takes the derivative of the cost function with respect to each weight, and then moves "down" the gradient, in the hopes that we will reach a global minimum. This process is roughly O(n^2), which is far better than the closed-form approach's complexity. In batch gradient descent, we can perform the gradient operation on the whole array efficiently, and get our new cost function in an easy mathematical step. This can, however, run into problems with local minima.

The solution to this problem is stochastic gradient descent. Stochastic gradient descent randomly selects a number of points in each epoch, computes their gradient, and alters the cost function accordingly. This leads to more randomness, which can pop the algorithm out of local minima, and makes it more likely to converge to a global minimum. The downside to stochastic gradient descent is that it is approximate - it will bounce around the point it converges to, rather than giving us an exact value.

We implement both batch gradient descent and stochastic gradient descent, and compare both algorithm's results across various hyperparameters. We then select the ideal hyperparameters, and do further comparisons for each

TABLE 1. SUMMARY STATISTICS OF ALL VARIABLES IN WINE DATASET

Index	Fixed Acid.	Vol. Acid.	Citric Acid	Res. Sugar	Chlorides	Free SO_2	Total SO_2	Density	pH	Sulphates	Alcohol	Quality
Count	1599	1599	1599	1599	1599	1599	1599	1599	1599	1599	1599	1599
Mean	8.320	0.528	0.271	2.539	0.088	15.875	46.468	0.997	3.311	0.658	10.423	5.636
StDev	1.741	0.179	0.195	1.410	0.047	10.460	32.895	0.002	0.154	0.170	1.066	0.808
Min	4.6	0.12	0	0.9	0.012	1	6	0.990	2.74	0.330	8.4	3
25%	7.1	0.39	0.09	1.9	0.07	7	22	0.996	3.21	0.550	9.5	5
50%	7.9	0.52	0.26	2.2	0.079	14	38	0.997	3.31	0.620	10.2	6
75%	9.2	0.64	0.42	2.6	0.09	21	62	0.998	3.40	0.730	11.1	6
Max	15.9	1.58	1	15.5	0.611	72	289	1.004	4.01	2.00	14.9	8

method. We will also test polynomial regression up to the fifth degree with batch gradient descent, to address problems of over/under-fitting in our linear regression model.

4. Results

The exhaustive list of RMSE values for various combinations of hyperparameters is given in Table 2 for batch gradient descent, and in Table 3 for stochastic gradient descent.

In the case of batch gradient descent, our validation RMSEs were relatively stable for all parameters, except for the case where the learning rate was quite small ($\eta < 0.01$). The best model was with $\lambda = 0, \eta = 0.1$ and L2, though this should be identical to the case of L1, since $\lambda = 0$ eliminates the regularization term, which means that the model with $\lambda = 0.1$ and L2 also has identical performance.

For the case of stochastic gradient descent, we have a little bit more diversity in the error. The best models appear to be those with smaller λ and L2 norm, and with $\eta = 0.01$. The actual best model had $\lambda = 0.001$ to be specific, but $\lambda = 0.0001$ was not far behind in second place. Some values for error were reported as very large, and even infinite, and this typically corresponded with a learning rate that was too large (i.e., $\eta = 0.1$). This would happen in the case of a large gradient, which stochastic gradient descent would be more prone to, since it computes the gradient of a random instance rather than the entire array at every step. Stochastic gradient descent thus seems to require a smaller learning rate to dilute the importance of the gradient at each step, otherwise anomalies dominate and error explodes.

A part of the clustering of RMSE values could be due to a relatively high training size of 0.1, which subdivided our error measurements into 10 iterations. Decreasing training size, say, to 0.01, might lead to better results and help in discriminating between these models more effectively, but would also take ten times longer to run, because we have to iterate through 100 subsets instead of 10.

The weights for stochastic and batch gradient descent are certainly different, especially in the case of a large learning rate, where the former may diverge due to anomalous gradients. But they are different even in the general case, because batch descent gets the gradients of all rows while stochastic descent gets them for single rows, and thus the weights of stochastic descent are merely approximations of the batch descent weights at any point. Furthermore, in the case where batch descent finds a local minimum, the weights may be drastically different than when stochastic descent is capable of finding the global minimum, because they have converged to two different targets.

Taking the best models from each set, we can plot the learning curves. Since this only requires running on two models, rather than all combinations of hyperparameters, I have made these plots with $\eta = 0.01$, so that we can see more details of the process. These curves are shown in Figure 1, with RMSE plotted on the vertical axes. For both models, the validation fold has significant error, and there is a gap between the training and validation folds. This typically indicates overfitting, possibly because our model is running for too long. A larger learning rate might help resolve this for batch gradient descent, but we'd need to find another solution for the stochastic descent, since a large learning rate caused issues (a cursory search suggested methods such as early stopping). Running our models on the test data shows an RMSE of 0.42483 for batch gradient descent, and an RMSE of 1.43426 for stochastic gradient descent.

We next ran our model for polynomial degrees 1, 2, 3, 4, and 5. I have plotted the curves for each of these in the appendix, with $\lambda = 0.1$ and the L2 regularizer. These all require different learning rates for results to not be NaNs, and for results to be visible, and so I have plotted each additional degree separately, with the values that gave results being $0.1, 0.01, 1e4, 1e6, 1e9$, respectively. The best model appears to be degree 3, as error increases after degree 4 and 5, as the result of greater overfitting.

We thus focus specifically on the degree 3 polynomial, and again apply our batch gradient descent algorithm and evaluate its performance. The learning curve of this particular degree is plotted in Figure 2, with $\eta = 1e4$ as above and $\lambda = 0.1$ and L2. Again, the curves are far apart, with training error being quite low. It thus still appears to suffer from overfitting, but we also haven't tuned the hyperparemeters as precisely as we did for the 1st-degree polynomial, and the validation RMSE is consistent with what we found in the polynomial graphs, where 1st-degree was twice as high for these parameters. We can thus infer that this model is better, even if it does not solve the

TABLE 2. BGD PARAMETERS, ORDERED BY VALIDATION FOLD RMSE

Parameters	Training RMSE	Validation RMSE
$\lambda = 0$, $\eta = 0.1$ l2	0.1869	0.6511
$\lambda = 0.1$, $\eta = 0.1$ l2	0.1789	0.6513
$\lambda = 0$, $\eta = .01$ l1	0.1928	0.6515
$\lambda = 1$, $\eta = 0.1$ l1	0.1847	0.6522
$\lambda = 1$, $\eta = 0.1$ l2	0.1879	0.6522
$\lambda = 0$, $\eta = 0.1$ l1	0.1883	0.6522
$\lambda = 0.0001$, $\eta = 0.1$ l2	0.1828	0.6523
$\lambda = 0.001$, $\eta = 0.1$ l2	0.1868	0.6524
$\lambda = 0.0001$, $\eta = .01$ l1	0.1913	0.6525
$\lambda = 1$, $\eta = .01$ l2	0.1928	0.6527
$\lambda = 0.01$, $\eta = .01$ l1	0.1798	0.6528
$\lambda = 0.0001$, $\eta = .01$ l2	0.1836	0.6529
$\lambda = 0$, $\eta = .01$ l2	0.1859	0.6529
$\lambda = 0.01$, $\eta = 0.1$ l2	0.1867	0.6529
$\lambda = 0.001$, $\eta = .01$ l1	0.1926	0.6529
$\lambda = 0.001$, $\eta = .01$ l2	0.1778	0.653
$\lambda = 0.01$, $\eta = .01$ l2	0.18	0.653
$\lambda = 0.001$, $\eta = 0.1$ l1	0.1849	0.653
$\lambda = 0.01$, $\eta = 0.1$ l1	0.1809	0.6533
$\lambda = 0.1$, $\eta = .01$ l2	0.1866	0.6535
$\lambda = 1$, $\eta = .01$ l1	0.1884	0.6539
$\lambda = 0.1$, $\eta = .01$ l1	0.188	0.653
$\lambda = 0.1$, $\eta = 0.1$ l1	0.1811	0.654
$\lambda = 0.0001$, $\eta = 0.1$ l1	0.1849	0.654
$\lambda = 0.1$, $\eta = .001$ l2	0.1497	1.0106
$\lambda = 0.1$, $\eta = .001$ l1	0.153	1.011
$\lambda = 0.01$, $\eta = .001$ l2	0.161	1.011
$\lambda = 0.01$, $\eta = .001$ l1	0.162	1.0116
$\lambda = 0$, $\eta = .001$ l2	0.1494	1.0117
$\lambda = 0.0001$, $\eta = .001$ l2	0.1585	1.0117
$\lambda = 1$, $\eta = .001$ l1	0.1546	1.0122
$\lambda = 0.001$, $\eta = .001$ l1	0.1638	1.0126
$\lambda = 0.0001$, $\eta = .001$ l1	0.1522	1.0127
$\lambda = 0$, $\eta = .001$ l1	0.16	1.0127
$\lambda = 1$, $\eta = .001$ l2	0.1608	1.0127
$\lambda = 0.001$, $\eta = .001$ l2	0.1665	1.0143
$\lambda = 0.1$, $\eta = .0001$ l2	0.0354	4.6792
$\lambda = 0.0001$, $\eta = .0001$ l1	0.0321	4.6793
$\lambda = 0.01$, $\eta = .0001$ l1	0.0313	4.6797
$\lambda = 0.01$, $\eta = .0001$ l2	0.0322	4.6797
$\lambda = 1$, $\eta = .0001$ l1	0.0334	4.6799
$\lambda = 0$, $\eta = .0001$ l2	0.0337	4.6803
$\lambda = 0.001$, $\eta = .0001$ l1	0.0332	4.6804
$\lambda = 0.001$, $\eta = .0001$ l2	0.031	4.6806
$\lambda = 1$, $\eta = .0001$ l2	0.0311	4.6806
$\lambda = 0$, $\eta = .0001$ l1	0.0336	4.6808
$\lambda = 0.1$, $\eta = .0001$ l1	0.0347	4.6809
$\lambda = 0.0001$, $\eta = .0001$ l2	0.034	4.681

TABLE 3. SGD PARAMETERS, ORDERED BY VALIDATION FOLD RMSE

Parameters	Training RMSE	Validation RMSE
λ = 0.001, η = .01 l2	0.1719	1.5346
λ = 0.0001, η = .01 l2	0.1856	1.5854
λ = 0.01, η = .01 l1	0.1743	1.5993
λ = 0.001, η = .01 l1	0.1686	1.6411
λ = 1, η = .01 l2	0.1697	1.6577
λ = 0.1, η = .01 l2	0.1624	1.6703
λ = 0, η = .01 l2	0.188	1.7068
λ = 0.01, η = .01 l2	0.1777	1.7089
λ = 0.0001, η = .01 l1	0.1757	1.771
λ = 1, η = .01 l1	0.1826	1.8956
λ = 0.1, η = .01 l1	0.1862	1.9295
λ = 0, η = .01 l1	0.1779	1.9646
λ = 0.1, η = .001 l1	0.1575	2.4476
λ = 0.0001, η = .001 l1	0.1774	2.4739
λ = 0, η = .001 l2	0.164	2.5868
λ = 0.1, η = .001 l2	0.1574	2.6043
λ = 1, η = .001 l2	0.1549	2.6222
λ = 0, η = .001 l1	0.1467	2.6556
λ = 0.001, η = .001 l1	0.1541	2.7628
λ = 0.01, η = .001 l2	0.15	2.8397
λ = 0.0001, η = .001 l2	0.1501	2.8764
λ = 0.01, η = .001 l1	0.1757	2.8882
λ = 1, η = .001 l1	0.1514	2.941
λ = 0.001, η = .001 l2	0.1452	2.9807
λ = 1, η = .0001 l2	0.0535	4.8056
λ = 0, η = .0001 l1	0.0485	4.8178
λ = 0.01, η = .0001 l2	0.046	4.8199
λ = 0.1, η = .0001 l2	0.0509	4.8261
λ = 0.001, η = .0001 l2	0.047	4.8558
λ = 0, η = .0001 l2	0.0466	4.8563
λ = 0.1, η = .0001 l1	0.0402	4.8574
λ = 0.0001, η = .0001 l1	0.0414	4.8608
λ = 0.0001, η = .0001 l2	0.044	4.8645
λ = 1, η = .0001 l1	0.0416	4.8731
λ = 0.01, η = .0001 l1	0.0511	4.8944
λ = 0.001, η = .0001 l1	0.0419	4.9191
λ = 1, η = 0.1 l2	1.8541e+110	1.4316e+112
λ = 0.0001, η = 0.1 l2	2.4460e+118	2.2568e+121
λ = 0, η = 0.1 l1	4.8150e+122	1.7873e+126
λ = 0.1, η = 0.1 l1	4.5386e+129	6.9984e+130
λ = 0.0001, η = 0.1 l1	3.8269e+131	5.6990e+133
λ = 0.001, η = 0.1 l2	6.0443e+132	2.5284e+135
λ = 0.01, η = 0.1 l1	6.5217e+148	3.8890e+150
λ = 1, η = 0.1 l1	inf	inf
λ = 0, η = 0.1 l2	inf	inf
λ = 0.1, η = 0.1 l2	inf	inf
λ = 0.01, η = 0.1 l2	inf	inf
λ = 0.001, η = 0.1 l1	inf	inf

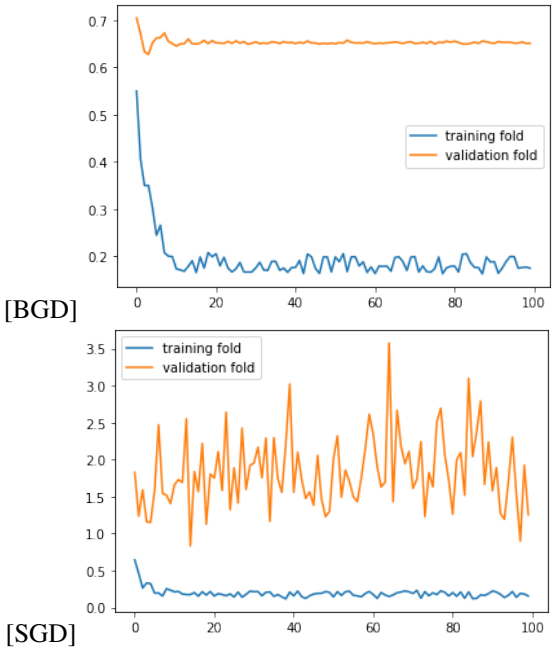

Figure 1. Learning curves of batch and stochastic gradient descent methods.

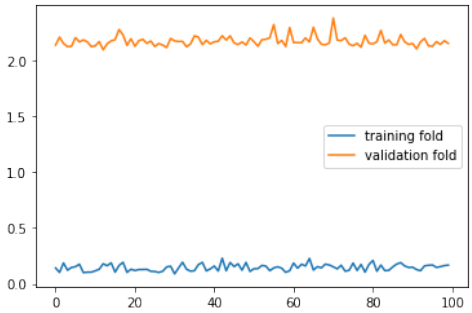

Figure 2. 3rd-degree polynomial learning curve.

overfitting problem entirely. The RMSE on the test data was 2.013274.

5. Conclusions

In conclusion, on the wine dataset that we tested the linear regression model on, the third-degree polynomial fit appears to work best, though it still suffers from some problems of overfitting. For the case of a 1st-degree fit, batch gradient descent performs better than stochastic gradient descent, with the best models having $\lambda = 0, \eta = 0.1$, L2 and $\lambda = 0.001, \eta = 0.01$, L2 respectively. Stochastic gradient descent appears to run into problems with a high learning rate, while a low learning rate results in higher error for batch gradient descent.

Appendix

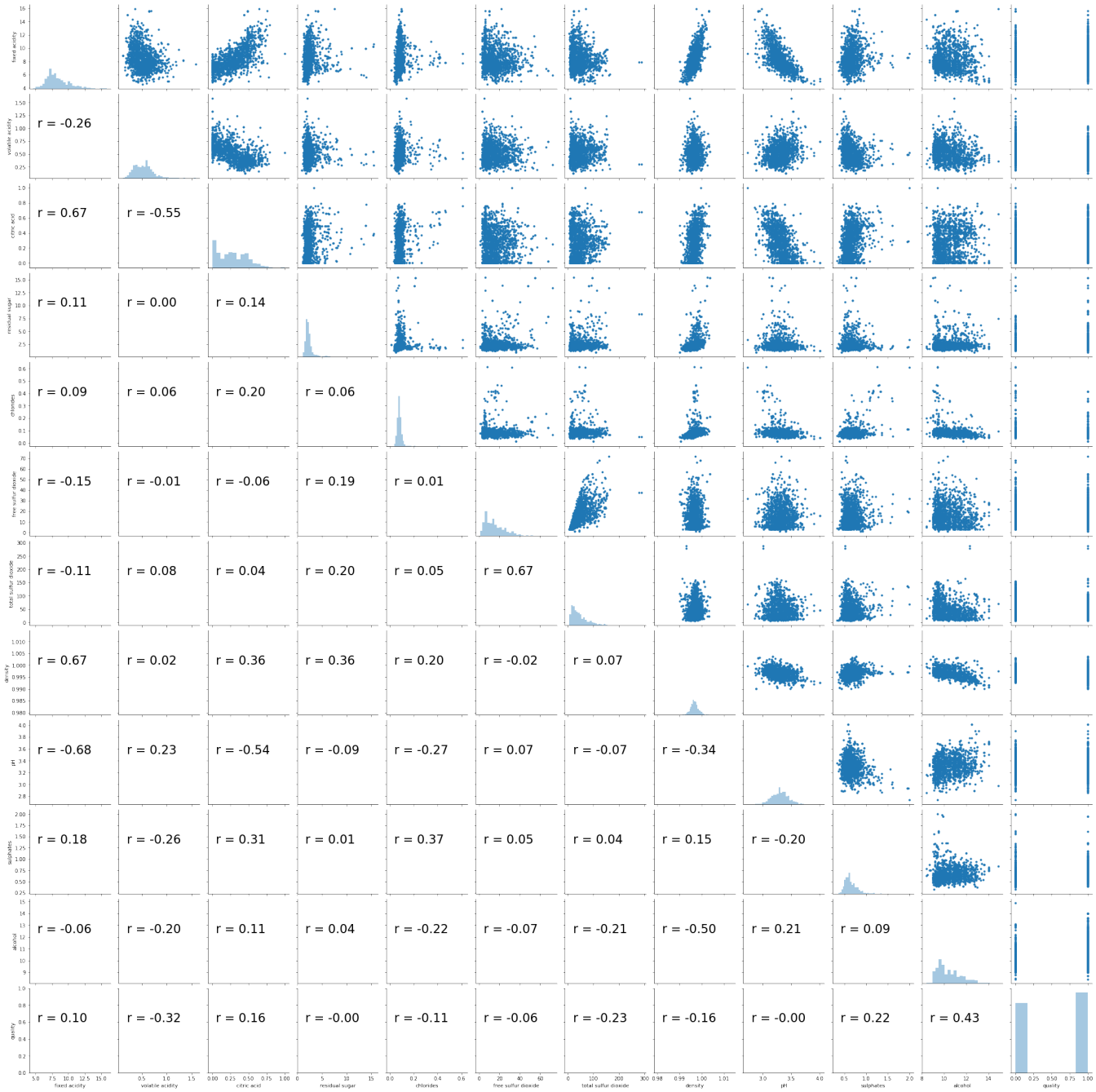

Figure A.1. Correlation plots for features of the wine quality dataset.

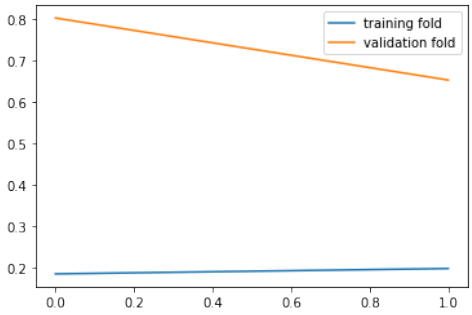

Figure A.2. Curve for polynomial degree 1.

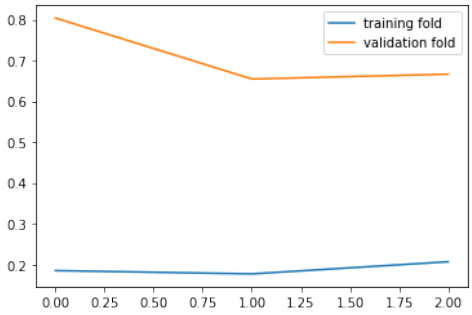

Figure A.3. Curve for polynomial degree 2.

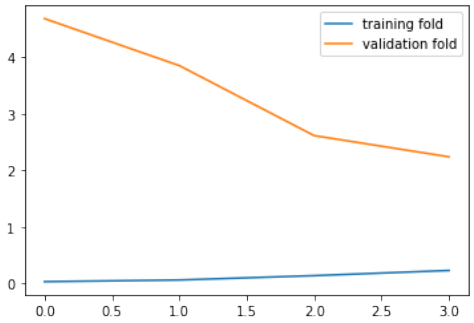

Figure A.4. Curve for polynomial degree 3.

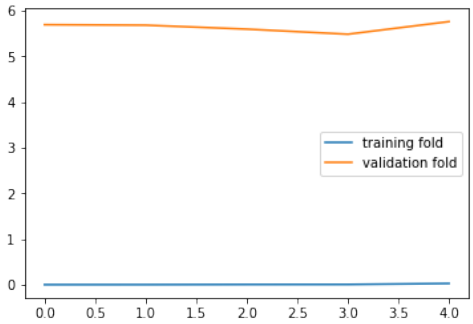

Figure A.5. Curve for polynomial degree 4.

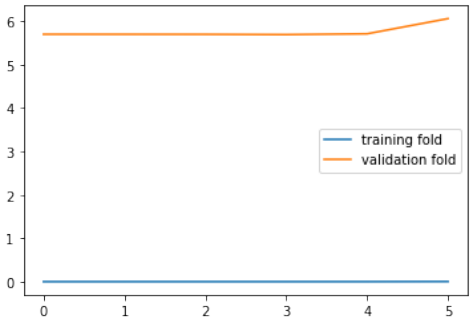

Figure A.6. Curve for polynomial degree 5.