703642, Advanced Machine Learning course 29th October, 2018

Assignment 1

Programming Project: Ridge Regression and Model Selection

Kamal Zakieldin, Yusuf Ipek

Overview:

In this report we are describing our implementation for the ridge regression model and our experiments we have made, first we introduce the used technology and the libraries, then we discuss our tasks illustrating what we have made then showing the results and the values we have got.

Technologies:

Python as the programming language.

Libraries:

This section will give a short overview of the libraries we used.

- **numpy** provide help working with matrices.
- **matplotlib** for all ploting and drawing the figures to visualize our results.

Discussion:

We start our implementation by generating the data set of N points, choosing a quadratic function f(x) using the following equation:

$$t_n = f(x_n) + \epsilon_i \quad \text{where} \quad \epsilon_i \sim^{iid} N(0, \sigma^2)$$

We have used epsilon to add a normal distributed uniform noise calculated using the mean and the variance of points.

in the beginning we have created dataset with high X values that create a very high Y values, and depending on high Y values were really difficult and increased the error, that's why we used a smaller X values that subsequently produced better Y values, generating the data set was one of the most underestimated step that affect the performance of our model.

Then we built our model of the linear regression model using matrix formulation.
Using the **Erms** to reduce the error between the actual values and the predicted values, one way to compute the minimum of a function is to set the partial derivatives to zero

$$\frac{\partial}{\partial\theta_i}\left[\frac{1}{2N}\parallel \boldsymbol{y}-Q\boldsymbol{\theta}\parallel^2\right]=0$$

We then use this derivation for all the weights.
We then can use them to solve our matrices to get the **f(x) =** $\sum_{j=0} w_j\phi_j(\mathbf{x})=\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$

We then get the design matrix;

$$\boldsymbol{\Phi}=\begin{bmatrix}\phi_0(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1)\\ \vdots & \ddots & \vdots\\ \phi_0(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N)\end{bmatrix}$$

and finally, we get the $\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{w}=\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}.$ as **AX = b**

and we can then calculate the weights using this equation and adding the regularization term to penalize the parameters.

$$\mathbf{w}=\left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}+\lambda\mathbf{I}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}$$

Results:

In this section we are trying to describe our model using the achieved results and insights.

Figure 1 illustrates the relationship between the lambda values and the position of the points and how regularization affect the predicted values.
By trying out different regularization parameters we figured out that values of lambda below -3 is a great choice. Taking smaller values for lambda didn't make such a great change, which can be seen on Figure 1.
However, choosing a good polynomial order is crucial for a good model. Low polynomial order can lead to under-fitting of the predicted model, when the actual data can only be generated with a higher polynomial order. [Figure 2] depicts the under-fitting where the predicted function is a 1-order polynomial and the actual data was a 2-order polynomial, we can easily notice how poorly the predictive power was, even with all lambda values, and it totally make sense, as we are trying to fit the model with less features than it should have.

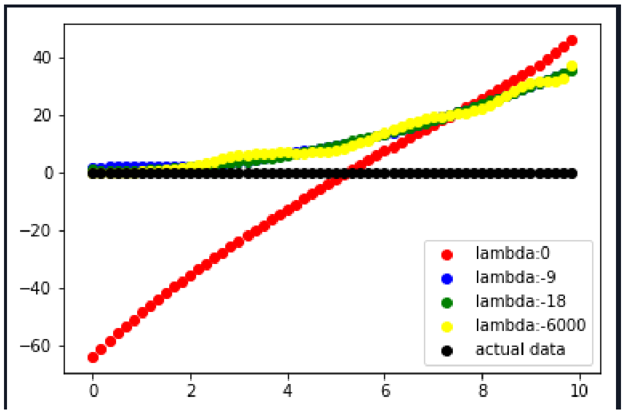

Figure 1 *For the visualization of a data set of 60 samples and a polynomial order of 12 is used. Red line is the output of the predicted function without any regularization, as we set lambda to 0. Other lines are prediction with regularization lambda. The blue line is the plot of the original data set*

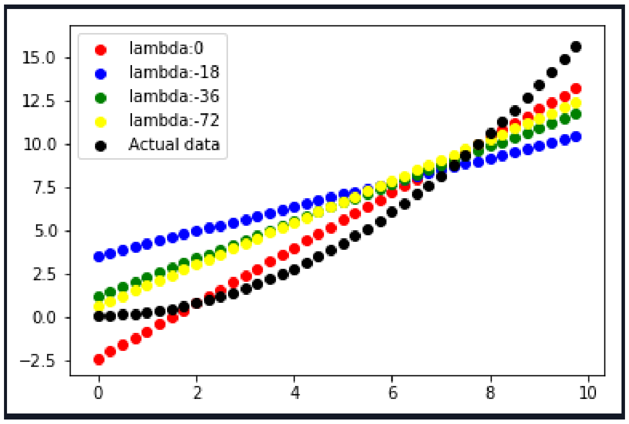

Figure 2 *Fitting a 1-order polynomial function to a quadratic function*

Figure 3 depicts the impact of the regularization parameter to the model. The predicted model is fitting the training set very well as the **ERMS** is nearly at 0, and for the test set the ERMS is higher, which tells us that the predicted model is over-fitted. We can see that the actual model was already an accurate model as the ERMS at lambda *0* is low. For the test set a jump can be seen on the Figure 3 for lambda *-1*, but as lambda decreases the ERMS also decreases and at -9 there is a turning point as the ERMS increases again.

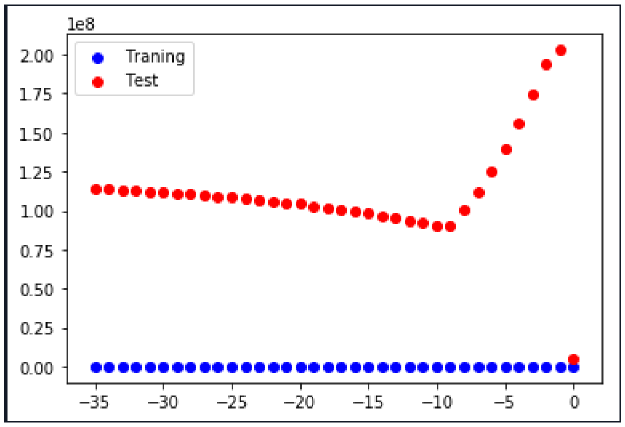

Figure 3 *ERMS according to chosen lambda, also by using K-Folds. Red points are the test set and blue points are the training set. Sample data N = 100, with k = 3 folds and polynomial order of 9.*

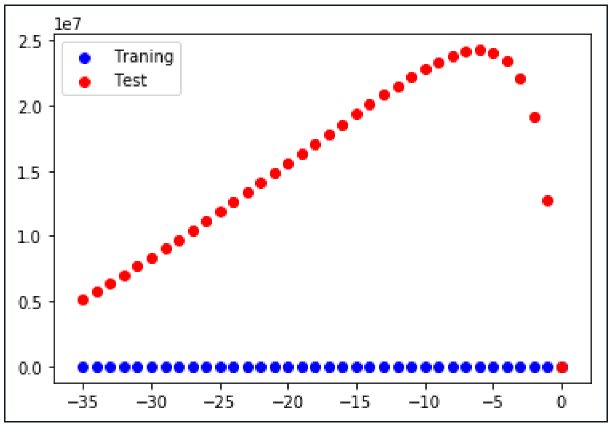

Figure 4 *Impact of K-Fold to the ERMS. Sample data N = 100, with 3 folds and polynomial order of 9.*

We can see by comparing Figure 3 and Figure 4, which have the same configuration except the K, that determining K by K-Fold does affect the predicted model, as the ERMS in Figure 4 is greater for lambdas between -3 and –20 but then the ERMS is smaller for lower lambdas compared to a higher K in Figure 3.

Figure 5 depicts a model which is over-fitted. When looking to the trainings data which is marked as blue, we see that the ERMS is low and slightly increase at order 8. In contrast the ERMS for the test set increases starts with a slightly higher ERMS than the ERMS of the trainings set, but at order 8 the ERMS increases dramatically.

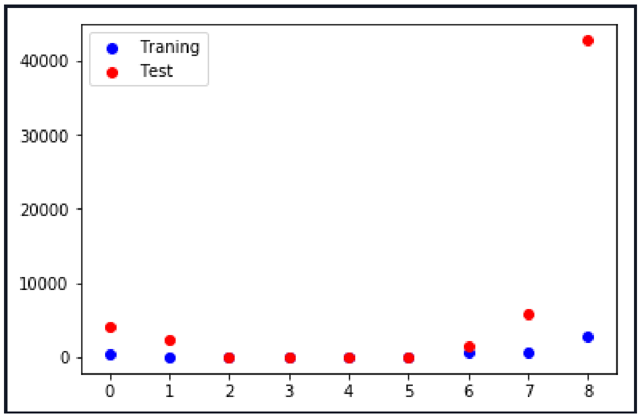

***Figure 5** Over-fitted model illustrate overfitting problem*

Finally, comparing Figure 6 and Figure 7, it is clear the difference between the overfitted model, and how it completely follow the actual data that trained on them, without any miss-matching in Figure 6, but in Figure 7 as the polynomial order has decreased to a reasonable polynomial number, so it doesn't have a high overfitted results.

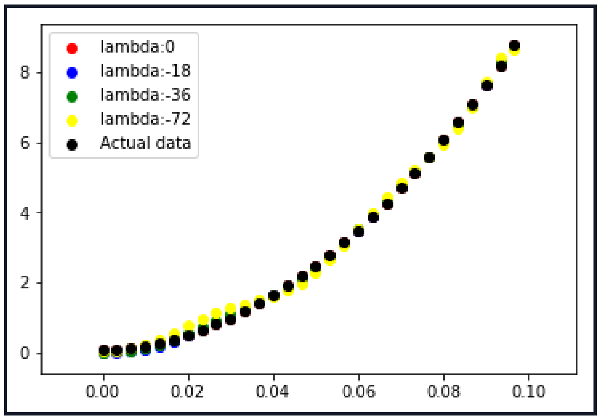

***Figure 6** Over-fitted model using polynomial order of 15*

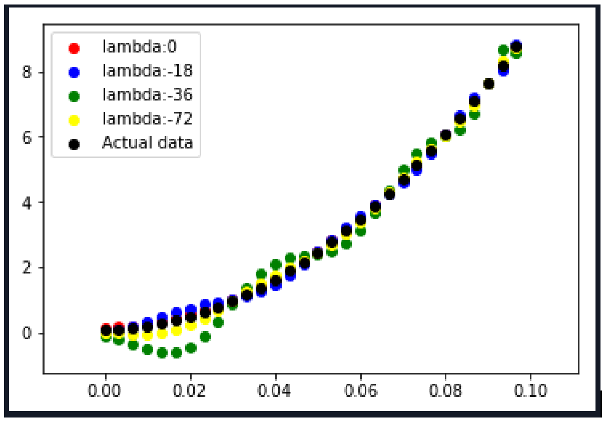

***Figure 7** neutral fitting model using polynomial order of 8*

References:

- M. Schmidt. " Least Squares Optimization with L1-Norm Regularization", December 2005.
- Risi Kondor. "Regression by linear combination of basis functions" , February 5, 2004.
- "Introduction to Linear Regression Analysis, Fifth Edition" . Douglas C. Montgomery, Elizabeth A. Peck, G. Geoffrey Vining. Chapter 5, 2012.
- Bishop, Christopher M. (2006). Pattern recognition and machine learning. New York :Springer, Chapter 3.