October 5, 2022

Abstract

Abstract

Contents

1 Introduction **1**

2 Method **2**

2.1 Splitting and scaling of data 4

2.2 Mean squared error and R squared 5

2.3 Bias-variance trade-off 5

2.4 Cross validation 7

2.5 Study of λ dependence for ridge and lasso regression 7

2.6 Study of topography data 8

3 Results **8**

4 Discussion **8**

5 Conclusion **8**

6 **8**

1 Introduction

In this project we will study the Franke function and real topography data by using regression analysis and resampling methods. These are both two dimensional systems which gives us a way of generalizing our functions to work for both. We will use ordinary least squares, ridge and lasso regression in our study of our data and do comparisons to try finding the optimal model for the different datasets.

2 Method

Our fist step is to set up our data for analysis. We start by implementing the Franke function which is a sum of four exponentials:

$$\begin{aligned} f(x,y) = \frac{3}{4}\exp\left(-\frac{(9x-2)^2}{4} - \frac{(9y-2)^2}{4}\right) + \frac{3}{4}\exp\left(-\frac{(9x+1)^2}{49} - \frac{(9y+1)}{10}\right) \\ + \frac{1}{2}\exp\left(-\frac{(9x-7)^2}{4} - \frac{(9y-3)^2}{4}\right) - \frac{1}{5}\exp\left(-(9x-4)^2 - (9y-7)^2\right) \end{aligned}$$

By using this function we generate our data by using $(n{+}1{\times}n{+}1)$ points in a meshgrid for x and y in the interval [0,1]. This gives us a function value of $(n+1\times n+1)$ to which we add a stochastic noise $\epsilon \sim N(0,\sigma^2)$ for a chosen standard deviation σ. This leaves us with our data described by the franke function plus some noise ϵ:

$$\boldsymbol{z} = f(\boldsymbol{x}) + \boldsymbol{\epsilon}$$

This can also be assumed of some real data, for the assumtion of the existence of a continous function f and normal distributed error ϵ.

The ordinary least squares gives us an aproximation to the above equation where we minimize $(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2$ to give us the matrix equation:

$$\tilde{\boldsymbol{z}} = \boldsymbol{X}\boldsymbol{\beta}$$

for a chosen design matrix $\boldsymbol{X}$ of some degree n.

This approximation gives us a new way of describing our dataset in terms of $\tilde{\boldsymbol{z}}$ instead of $f(\boldsymbol{x})$.

$$\boldsymbol{z} = \sum_{j=0}^{n-1} \beta_j x_i^j + \epsilon_i$$

This is the same as the following matrix equation:

$$\boldsymbol{z} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} = \tilde{\boldsymbol{z}} + \boldsymbol{\epsilon}$$

This means that the elemnt i is given by:

$$z_i = \epsilon_i + \sum_j x_{ij}\beta_j$$

and the expectation value of the element i in $\boldsymbol{z}$:

$$E(z_i) = E(\epsilon_i + \sum_j x_{ij}\beta_j)$$

Since we already know or have assumed that $\boldsymbol{\epsilon}$ is normal distibuted with the expectation value 0 $E(\epsilon_i) = 0$, this gives us:

$$E(z_i) = E(\epsilon_i) + E(\sum_j x_{ij}\beta_j) = \sum_j E(x_{ij}\beta_j)$$

x_{ij} and β_j are values and not distrubutions which means they have themselves as expectation values:

$$E(z_i) = \sum_j x_{ij}\beta_j = \boldsymbol{X_{i*}\beta}$$

To find the variance we again recognize that $\boldsymbol{X\beta}$ follow no distribution which leaves us with the variance of $\boldsymbol{z}$ equaling the variance of $\boldsymbol{\epsilon}$ which is given as σ^2:

$$\text{Var}(\boldsymbol{z}) = \text{Var}(\boldsymbol{X\beta} + \boldsymbol{\epsilon}) = \text{Var}(\boldsymbol{\epsilon}) = \sigma^2$$

$\tilde{\boldsymbol{z}}$ is defined through the minimization of the mean square error $(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2$ which for $\tilde{\boldsymbol{z}} = \boldsymbol{X\beta}$ translates to the minimization of the cost function:

$$C(\boldsymbol{\beta}) = \frac{1}{n}\{(\boldsymbol{z} - \boldsymbol{X\beta}^T\boldsymbol{z} - \boldsymbol{X\beta})\}$$

Where we take the derivative with respect to $\boldsymbol{\beta}$ and solve where the derivative is 0 to find the minimum:

$$\frac{\partial C(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{X}^T(\boldsymbol{z} - \boldsymbol{X\beta}) = 0$$

$$\boldsymbol{X}^T\boldsymbol{z} = \boldsymbol{X}^T\boldsymbol{X\beta}$$

We assume that $\boldsymbol{X}^T\boldsymbol{X}$ is invertible which gives us the the optimal $\boldsymbol{\beta}$:

$$\tilde{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{z}$$

We can now find the expectation value for the optimal $\tilde{\boldsymbol{\beta}}$:

$$\begin{aligned}
E(\tilde{\boldsymbol{\beta}}) &= E[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{z}] = E[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\epsilon})] \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T E[(\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\epsilon})] \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{\beta} \\
&= \boldsymbol{\beta}
\end{aligned}$$

Here we see that the expectation value of our optimal parameter is the parameter $\boldsymbol{\beta}$.

We now find the variance of the optimal parameter:

$$\begin{aligned}
\mathrm{Var}(\tilde{\boldsymbol{\beta}}) &= \mathrm{Var}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T(\boldsymbol{X}\boldsymbol{\beta}+\boldsymbol{\epsilon})] \\
&= \mathrm{Var}[(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{\epsilon}]
\end{aligned}$$

For a matrix A we have $\mathrm{Var}(\boldsymbol{A}\boldsymbol{X}+b) = \boldsymbol{A}\mathrm{Var}(\boldsymbol{X})\boldsymbol{A}^T$ which gives us:

$$\begin{aligned}
\mathrm{Var}(\tilde{\boldsymbol{\beta}}) &= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathrm{Var}[\boldsymbol{\epsilon}]((\boldsymbol{X}^T\boldsymbol{X})^{-1})^T\boldsymbol{X} \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\mathrm{Var}[\boldsymbol{\epsilon}](\boldsymbol{X}^T)^{-1}\boldsymbol{X}^{-1}\boldsymbol{X} \\
&= (\boldsymbol{X}^T\boldsymbol{X})^{-1}\mathrm{Var}[\boldsymbol{\epsilon}]
\end{aligned}$$

This we can use to compute the optimal β and in turn give us a OLS prediction for our real function f.

2.1 Splitting and scaling of data

A central part of regression analysis is to scale and split our data to avoid both overfitting and avoid the MSE to contain the intercept. In our case of the Franke function we have $x, y \in [0, 1]$ and follwing z data on the same order which may indicate that a scaling is unessecary. This expectation comes from that our data does not include any extreme values of any other order that may greatly negativly incluence our predictors. In other words scaling of data is normaly what we do to get the data as equal to each other as our Franke data already is. To test if this expectation is valid, we test this and compare our non-scaled data to mean scaled data where we have subtracted the mean.

To avoid overfitting we perform a 80-20 percent split into train and test data. We use the train data to train the model or in other words find the β parameters. To predict we use the trained model together with the test data which leaves us with a more generalized prediction not only working for the data we have at hand, but also other data coming from the same source. This can be explained by for example a model trained so well with a design matrix $\boldsymbol{X}$ of an infinite degree that its prediction follows every noisy outlier. This woulds match the data at hand very well and give us an extremly low MSE, but give us high MSE for new data which were not used to train the model. To test this we look at both MSE and R squared for predictions for $\boldsymbol{X}$ of different degrees using only train data compared to predictions made using both train and test data.

2.2 Mean squared error and R squared

For different design matrices X with different polynomial degrees we can calculate the mean squared error and R squared. This is done using the following equations where we define z as our test data containing $n+1$ points.

$$MSE(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = \frac{1}{n}\sum_{i=0}^{n}(z_i - \tilde{z}_i)^2$$

$$R^2(\boldsymbol{z}, \tilde{\boldsymbol{z}}) = 1 - \frac{\sum_{i=0}^{n}(z_i - \tilde{z}_i)^2}{\sum_{i=0}^{n}(z_i - \bar{z}_i)^2}$$

For the mean value:

$$\bar{z} = \frac{1}{n}\sum_{i=0}^{n} z_i$$

2.3 Bias-variance trade-off

To study the bias-variance trade-off we first implement the bootstrap resampling technique. This is done by resampling our train data n_B times with replacement. That means we gain n_B new samples that may be unique but still only contain values from the original sample. This in turn is used to compute our predictions leaving us with a total of n_B different predictions. This can be used to study the bias and variance of the predictions.

By looking back at the cost function which is minimized in order to find our optimal parameters $\boldsymbol{\beta}$ we can derive an expression for the bias and variance.

$$C(\boldsymbol{X},\boldsymbol{\beta}) = \frac{1}{n}\sum_{i=0}^{n-1}(z_i - \tilde{z}_i)^2 = \mathbb{E}[(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2]$$

By the definition of our data $z = f + \epsilon$ we rewrite the costfunction:

$$\begin{aligned}
\mathbb{E}[(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2] &= \mathbb{E}[(f + \epsilon - \tilde{z} + \mathbb{E}[\tilde{z}] - \mathbb{E}[\tilde{z}])^2] \\
&= \mathbb{E}[f^2 + \epsilon^2 + \tilde{z}^2 + 2\,\mathbb{E}[\tilde{z}]^2 \\
&+ 2f\epsilon + 2f\tilde{z} + 2f\,\mathbb{E}[\tilde{z}] - 2f\,\mathbb{E}[\tilde{z}] \\
&+ 2\epsilon\tilde{z} + 2\epsilon\,\mathbb{E}[\tilde{z}] - 2\epsilon\,\mathbb{E}[\tilde{z}] \\
&+ 2\tilde{z}\,\mathbb{E}[\tilde{z}] - 2\tilde{z}\,\mathbb{E}[\tilde{z}] - 2\,\mathbb{E}[\tilde{z}]^2] \\
&= \mathbb{E}[(f - \mathbb{E}[\tilde{z}])^2] + \mathbb{E}[\epsilon^2] + \mathbb{E}[(\mathbb{E}[\tilde{z}] - \tilde{z})^2] \\
&+ 2(f - \mathbb{E}[\tilde{z}])\,\mathbb{E}[\epsilon] + 2\,\mathbb{E}[\epsilon]\,\mathbb{E}[(\mathbb{E}[\tilde{z}] - \tilde{z})] \\
&+ 2(f - \mathbb{E}[\tilde{z}])\,\mathbb{E}[(\mathbb{E}[\tilde{z}] - \tilde{z})]
\end{aligned}$$

By using that $\epsilon \sim N(0,1)$ we have $\mathbb{E}[\epsilon] = 0$ and $\mathrm{Var}[\epsilon] = \mathbb{E}[\epsilon^2] - \mathbb{E}[\epsilon]^2 = \mathbb{E}[\epsilon^2] = \sigma^2$ from this we get:

$$\begin{aligned}
\mathbb{E}[(\boldsymbol{z} - \tilde{\boldsymbol{z}})^2] &= \mathbb{E}[(f - \mathbb{E}[\tilde{z}])^2] + \mathbb{E}[(\tilde{z} - \mathbb{E}[\tilde{z}])^2] + \sigma^2 + 2(f - \mathbb{E}[\tilde{z}])\,\mathbb{E}[\mathbb{E}[\tilde{z}] - \tilde{z}] \\
&= \mathbb{E}[(f - \mathbb{E}[\tilde{z}])^2] + \mathbb{E}[(\tilde{z} - \mathbb{E}[\tilde{z}])^2] + \sigma^2 \\
&= \mathbb{E}[(z - \epsilon - \mathbb{E}[\tilde{z}])^2] + \mathbb{E}[(\tilde{z} - \mathbb{E}[\tilde{z}])^2] + \sigma^2 \\
&= \mathbb{E}[(z - \mathbb{E}[\tilde{z}])^2] + \mathbb{E}[\epsilon^2 - 2z\epsilon + 2\epsilon\,\mathbb{E}[\tilde{z}]] + \mathbb{E}[(\tilde{z} - \mathbb{E}[\tilde{z}])^2] + \sigma^2 \\
&= \mathbb{E}[(\mathrm{Bias}[\tilde{y}])^2] + \mathbb{E}[(\tilde{z} - \mathbb{E}[\tilde{z}])^2] + \sigma^2
\end{aligned}$$

Here we have the expectation value of the bias which is the bias itself, while we rewrite the second term in terms of a sum:

$$\begin{aligned}
&(\mathrm{Bias}[\tilde{y}])^2 + \frac{1}{n}\sum_i(\tilde{z} - \mathbb{E}[\tilde{z}])^2 + \sigma^2 \\
&= (\mathrm{Bias}[\tilde{y}])^2 \quad + \quad \mathrm{var}[\tilde{f}] \quad + \sigma^2
\end{aligned}$$

2.4 Cross validation

To futher investigate our model we implement the cross validation resampling method. Togther with the already implemented bootstrap method we now have a possibility to optimize our predictions by chosing a design matrix that reduces our models mean squared errors. The cross validation method takes in data and splits it up in k equaly sized sets where one set is used as test data while the rest is used as train data. For every fold a mean squared error is then computed before the model reruns with another set as test data. This process happens over and over until all k sets have been used as test data. This gives a total of k folds. Since our data is ordered and not chaotic we shuffle it before the splits are performed. We can see why this is neccesary in the k-fold visualization in figure 1 where we would without a shuffle have x, y an z not representative of the whole dataset but rather small intervals of values. We can imagine how not shuffling the data would have consequences on a extreme case where the first split of the first k-fold in the figure below only contained values around 0 while the rest contained values around 1. That would leave us with high computed mean squared erros for especially some of the folds which would in turn give us bad predictions for the MSE.

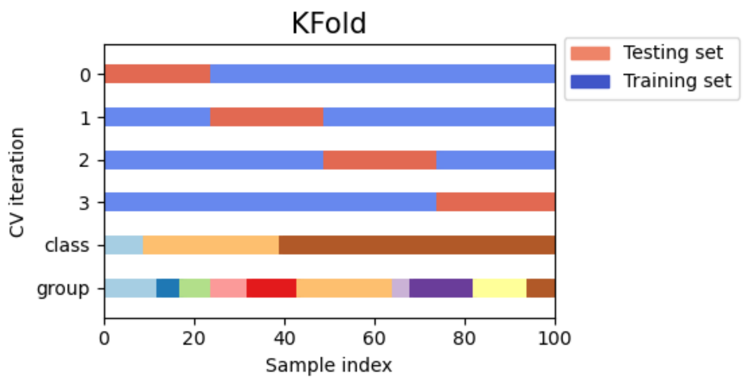

Figure 1: A visualization of the splits of the data performed in cross validation. Here we see 4 k-folds which gives a total of 4 splits and iterations.
https://scikit-learn.org/stable/modules/cross_validation.html

2.5 Study of λ dependence for ridge and lasso regression

To get the best possible predictions of our datasets we also implement lasso and ridge regression. These two methods are unlike OLS, dependent on a regularization parameter λ where we have to compute which value that optimizes these two models. Similar to OLS we perform cross validation to find which degree of design matrix reduces the mean squared error of our predictions, only that we for ridge and lasso find the best λ at the same time. This is because we for one degree would have one optimal λ and for another degree have another. We therefore compute one cross validation for different values of λ parameters for each degree. The degree and λ giving the smallest MSE is then used for futher predictions

2.6 Study of topography data

3 Results

4 Discussion

5 Conclusion

6