# Assignment IV: Bayes'Theorem and Ridge Regression

**Exercises in Machine Learning (190.013), SS2022**
**Stefan Nehl**[1]

[1]*stefan-christopher.nehl@stud.unileoben.ac.at, MNr: 00935188*, *Montanuniversität Leoben, Austria*

May 9, 2022

**In the fourth assignment, I had to solve three different task. First, calculate the probability for a positive test results with the Bayes' Theorem, next describe the ridge regression and derive the weight update for with the least squares regression and last implement the ridge regression. The implementation of the ridge regression also includes testing the model and plotting it's results.**

## 1 Task 1: Bayes'Theorem

The Bayes'Theorem is a mathematical formula which describes the probability of an event. Furthermore, it is used for calculating conditional probabilities. (Joyce, 2021)

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

(Rueckert, 2022)

I used this formula to calculate the probability to be infected with SARS CoV2 and having a positive test result of an antigen test. Let A ∈ [infected, non-infected] the event, which defines if a person is infected or not, and B ∈ [+,-] the event, which defines the result of the antigen test.

### 1.1 Implementation

First, I created the following variables with the values.

**Table 1:** *Variables and Values*

| name | value |
|---|---|
| *populationAustria* | 9095538 |
| *activeCases* | 441098 |
| *covTestSensitivity* | 0.971 |
| *covTestSpecific* | 0.995 |

First, I set the variable for *p(+|inf)* to the value of *covTestSensitivity* and the variable for *p(-|nInf)* tp the value of *covTestSpecific*. Next, I calculated the value for *p(inf)*, *p(nInf* and stored the values in the variables *pInfected* and *pNotInfected*.

$$p(inf) = \textit{activeCases}/\textit{populationAustria}$$

$$p(nInf) = 1 - p(inf)$$

The variable *p(inf)* defines the value for the probability to be infected with covid and *p(nInf)* not. Furthermore, the abbreviation for infected is *inf* and for non infected *nInf*.The abbreviation for having a positive test result is $+$ and for a negative test result $-$. Next, I initialized the following variables and calculated there values with the following formulas.

$$p(nInf\&-) = p(nInf) * p(-|non - infected)$$

$$p(-) = p(inf\&-) + p(nInf\&-)$$

$$p(nInf\&+) = p(nInf) - p(nInf\&-)$$

$$p(+) = p(inf\&+) + p(nInf\&+)$$

$$p(-|inf) = \frac{p(inf\&-)}{p(-)}$$

$$p(+|nInf) = \frac{p(nInf\&+)}{p(nInf)}$$

Last, I used the Bayes'Theorem to calculate the $p(infected|+)$ value.

$$p(inf|+) = \frac{p(+|inf) * p(inf)}{p(nInf)}$$

### 1.2 Result and Conclusion

The results of the calculation is displayed in Table 2. The result for *p(inf|+)* is $0.630525 \approx 63.05\%$. Which means there is a $63.05\%$ chance to be infected with covid and get a positive test result. The implemented code can be found in the appendix of this paper.

**Table 2:** *Results*

| name | value |
|---|---|
| *p(-\|inf)* | 00.15% |
| *p(+\|nInf)* | 02.90% |
| *p(inf)* | 04.85% |
| *p(nInf)* | 95.15% |
| *p(+)* | 07.47% |
| *p(inf\|+)* | 63.05% |

## 2 Task 2: Ridge Regression

Ridge regression is used for parameter estimation to address the collinearity problem in multiple linear regression. (McDonald, 2009) The Ridge Regression adds the quadratic regularization term $\frac{\lambda}{2}\left(\boldsymbol{\omega}^{\mathrm{T}}\boldsymbol{\omega}\right)$ to the objective $\mathbf{J}_{LS}$. (Rueckert, 2022)

### 2.1 Derivation of the Least Squares Solution

For the derivation of the least squares I defined the vectors $\mathbf{y} \in \mathbb{R}$, $\mathbf{A} \in \mathbb{R}^{nxM}$ and $\omega \in \mathbb{R}^{M}$ where M is the dimension and n the number of samples.

$$\frac{\partial J_{LS}}{\partial \boldsymbol{\omega}} = \frac{\partial}{\partial \boldsymbol{\omega}}\{1/2\sigma^{-2}(\mathbf{y}-\mathbf{A}\boldsymbol{\omega})^{T}(\mathbf{y}-\mathbf{A}\boldsymbol{\omega})\}$$

After the partial deviation we receive the following equation.

$$\frac{\partial J_{LS}}{\partial \boldsymbol{\omega}} = 1/2\sigma^{-2}(-2\mathbf{y}^{T}\mathbf{A} + 2\boldsymbol{\omega}^{T}\mathbf{A}^{T}\mathbf{A})$$

I set this equation to zero to calculate the least square solution.

$$\begin{gathered}\frac{\partial J_{LS}}{\partial \boldsymbol{\omega}} = 0, \\ 1/2\sigma^{-2}(-2\mathbf{y}^{T}\mathbf{A} + 2\boldsymbol{\omega}^{T}\mathbf{A}^{T}\mathbf{A}) = 0, \\ 1/2\sigma^{-2}(-2\mathbf{y}^{T}\mathbf{A} + 2\boldsymbol{\omega}^{T}\mathbf{A}^{T}\mathbf{A}) = 0, \\ -\mathbf{y}^{T}\mathbf{A} + \boldsymbol{\omega}^{T}\mathbf{A}^{T}\mathbf{A}\boldsymbol{w} = 0, \\ \boldsymbol{\omega} = (\mathbf{A}^{T}\mathbf{A})^{-1}\mathbf{A}^{T}\mathbf{y}.\end{gathered}$$

(Rueckert, 2022) Important here is, that the matrix $\mathbf{A}$ has a full rank and is invertible. If this is not the case I would use the Moore–Penrose inverse which is described in the following formula.

$$\mathbf{A}^{+} = (\mathbf{A}^{T}\mathbf{A})^{-1}\mathbf{A}T.$$

(Rueckert, 2022)

## 3 Task 3: Implementation of Ridge Regression

The last task was to implement the ridge regression for a given dataset. The dataset includes longitude and latitude of a map with the corresponding temperature data.

### 3.1 Import Data and Implementation

For the implementation I first created the abstract class *Regression* which includes the abstract methods *importData*, *generateTrainingSubset*, *computeLinearRidgeRegression*, *testModel*, *computeError*, *plotError*, *plotHeatMap*, *computMeanOfError* Then i created the class *RidgeRegression* which implements those methods and has the parameter *trainStep* which indicates the size of the training data. For importing the data I used the scipy.io library which is able to read *MatLab* files and load this data. I used only the first dataset of the time series data to create the model. The method *generateTrainingSubset* creates the training data with the parameter *trainStep*. The parameter *trainStep* defines the size of the training set. For example, if the value is set to 4 every 4. values is used for the calculation of the weight values.

### 3.2 Ridge Regression

For the Implementation of the Ridge Regression calculation I added the method *computeLinearRidgeRegression* and made the calculation. I followed the formula from chapter 2 and used the *numpy* library to do the matrix calculations. I added a helper method which I used to create the feature vector for the calculation. The method is name *createFeatureVector* and takes the parameter *x*. The parameter *x* is the vector of the current y-value. In our case it's a two dimensional vector with the longitude and latitude. I augmented this vector and created the new vector *featureVector* which is a three dimensional vector. The first dimension contains a 1, the second the latitude and the third the longitude. This vector was then returned from the method. After the creation of the feature vectors, the method *computeLinearRidgeRegression* finishes it's calculations and returns the weight vector.

### 3.3 Plotting

### 3.4 Results

Figure 1 shows the gauss distribution for one dimension. It displays the values distribution and the frequencies of those values. The orange line displays the gauss distribution itself.

### 3.5 Conclusion

The implementation of the abstract class was straightforward. For the other classes I made some changes. I moved the classes to separate files to handle them better. Unfortunately, I was not able to create a satisfying plot for the gauss distribution for two dimensions. The whole code is in the appendix of this paper.

## Bibliography

Joyce, James (2021). "Bayes' Theorem". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2021. Metaphysics Research Lab, Stanford University.

McDonald, Gary C. (2009). "Ridge regression". In: *WIREs Computational Statistics* 1.1, pp. 93–100. DOI: `https://doi.org/10.1002/wics.14`. eprint: `https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/wics.14`. URL: `https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/wics.14`.

Rueckert, Elmar (2022). *An Introduction to Probabilistic Machine Learning*. Elmar Rueckert.

## APPENDIX