Assignment V: Gaussian Process 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 27, 2022

In the fifth assignment, I had to describe the Guassian Process Regression and implement it with the library GPy. Furthermore, I had to test different kernel implementations and hyper parameters and verify and compare the results with the results of the last assignment.

1 Gaussian Process Regression

Gaussian processes are a class of nonparametric models for machine learning. They are commonly used for modeling spatial and time series data. (Powell, 2021) The *Gaussian Process Regression* uses the *Multivariate Conditional Distribution*.

$$f(\boldsymbol{x'}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{\sqrt{(2\pi)^{\mathrm{k}}|\boldsymbol{\Sigma}|}} exp^{-\frac{1}{2}(\boldsymbol{x'} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{\text{-}1}(\boldsymbol{x'} - \boldsymbol{\mu})}$$

With $\boldsymbol{x'}$= {x'$_1$, ..., x'_k}, $\boldsymbol{\Sigma}$ the covariance matrix and $|\boldsymbol{\Sigma}|$ the determinant of the covariance matrix. The covariance is determined by the covariance function of the kernel and has to be positive definite. (Rueckert, 2022) I tried three different kernels for the *Gaussian Process Regression*. The *Linear Kernel*, the *RBF*, *Radial Basis Function* and the *Matern 52*.

1.1 Linear Kernel

The *Linear Kernel* is based on linear classification. This classification is based on the linear combination of the characteristics. The decision function can be described with:

$$d(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x}) + b$$

where $\boldsymbol{w}$ is the weight vector, b a biased value and $\phi(\boldsymbol{x})$ a higher dimensional vector of **x**. If $\boldsymbol{w}$ is a linear combination of training data $\boldsymbol{w}$ can be calculated with:

$$\boldsymbol{w} = \sum_{i=1}^{l} \boldsymbol{\alpha}_i\phi(\boldsymbol{x_i})$$

for some $\boldsymbol{\alpha} \in \mathbf{R}^1$ The kernel function can be calculated with

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \phi(\boldsymbol{x}_i))^T\phi(\boldsymbol{x}_j))$$

(Guo-Xun Yuan and Lin, 2021) The linear kernel is good for data with a lot of features. That's because mapping the data to a higher dimensional space does not really improve the performance. (KOWALCZYK, 2014) The implementation in the *GPy* library was the following:

$$K(x, y) = \sum_{i=1}^{D} \sigma_i^2 x_i y_i$$

Where D defines the dimension and σ_i^2 the variance for each dimension.

1.2 RBF

The *Radial Basis Function* is one of the most used kernels. It's similar to the *Gaussian distribution*. The kernel calculates the similarity or how close two points are to each other.

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \ epx(-\frac{||\boldsymbol{x}_i - \boldsymbol{x}_j||}{2\sigma^2})$$

Where $||\boldsymbol{x}_i - \boldsymbol{x}_j||$ is the euclidean (L_2-Norm) and σ the variance and the hyper parameter. (Sreenivasa, 2020)

$$K(r) = \sigma^2 \exp\left(-\frac{1}{2}r^2\right)$$

Implementation in the *GPy* library where $r = |\boldsymbol{x}_i - \boldsymbol{x}_j|$ and σ^2 the variance.

1.3 Matern 52

The *Matern 52* is a generalization of the *RBF* kernel.

$$K(r) = (1 + \frac{\sqrt{5}r}{l} + \frac{5r^2}{3l^2})\ exp(-\frac{\sqrt{5}r}{l})$$

Where $r = |\boldsymbol{x}_i - \boldsymbol{x}_j|$ and l a positive parameter.(C. E. Rasmussen, 2006) However, if I take a look in the code of the *GPy* implementation. It looks a little bit different.

$$K(r) = \sigma^2(1 + \sqrt{5}r + \frac{5}{3}r^2)\exp(-\sqrt{5}r)$$

It looks like, the positive parameter l is set to 1. As additional hyper parameter the variance, σ^2, was introduced.

1.4 Hyper-Parameters

The *GPy* library has for every kernel a variance parameter. This parameter is a hyper parameter to adjust improve the prediction result. If I set the parameter to a lower value the values for the learning are in a smaller gap. If I increase σ^2 the gab increases. So if the variance of my data is high, an increased value for the parameter variance makes sense.

1.5 Implementation

For the implementation of the *Gaussian Process Regression* i created a the class *GaussianProcess* which derived from the class *Regression* and reused the functions, *importData*, *generateTrainingSubset*, *createFeatureVector*, *testModel*, *computMeanOfError*, *getMeanError*, *plotError* and *plotHeatMap* from the last assignment about *Ridge Regression*. I implemented than the function *computeGaussianProcessRegression* which takes the parameters *kernelSetting* and *variance*. The parameter *kernelSetting* is an enum with the following values:

Table 1: *Values for the kernel settings*

Value	String
LinearKernel	Linear Kernel
RBFWithGpu	RBF with GPU
RBF	RBF
Matern52	Matern 52

I checked those values with an if and set the kernel to the corresponding value. The kernel function, *GPY.kern.KernelName* got 2 more parameters. One was the dimension of the data and the other one the variance, which is the hyper parameter. The dimension was set to two and for the variance I tried different values. After the kernel I created the model with the function *GPY.models.GPRegression* which takes the x and y values and the kernel as a parameter. The y values where normalized to have a mean of zero in the data and I used only a subset of the training data to overcome the performance issues. The size of the subset can be set with the parameter *trainStep* which has the same implementation from assignment 4. After the model creation I optimized the model with the *model.optimize* function. This function takes the max iterations as a parameter. I set this value to 1000. After the optimization I used the *model.predict* function and passed the y test data to the trained model. With the returned y values and the y test data from our data set, I calculated the error and the mean of the error. The last step was the plotting of the descending error, the heat map and compared the error with the error of the*Ridge Regression*.

1.6 Result

First I tested which kernel performs the best. For this I created a test with the train steps of 20 and a hyper parameter of 1 and tested all three kernels and compared the error with the error of the *Ridge Regression*. Figure 1 displays the different values. Figure 1 shows, that

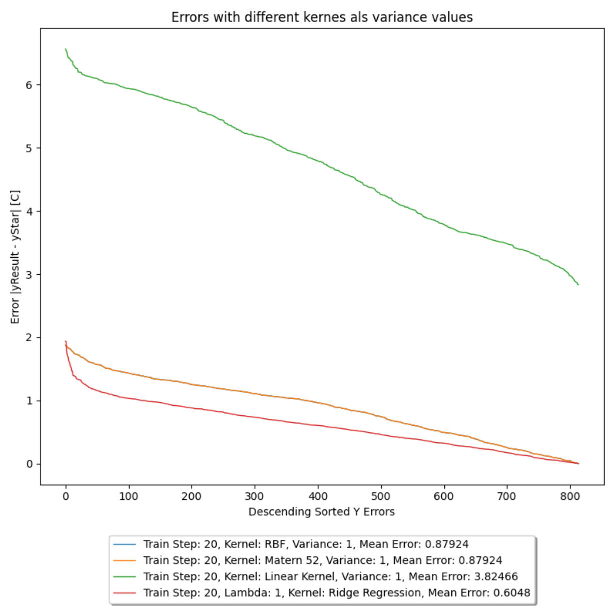

Figure 1: *Error with different kernels and variance:1*

in this case the *Ridge Regression* performs better than the *Gaussian Processes* . From the *Gaussian Processes* the *RBF* and the *Matern 52* performs better than the *Linear Kernel*. Also the time for the learning process took some time. Table 2 displays the learning time from the different kernels.

Table 2: *Learning time for the different kernels*

Kernel	Time
RBF	1m 53s
Matern 52	2m 13s
Linear Kernel	1m 8s

Because of the results and the performance I used the *Matern 52* kernel for the next tests. I tested the *Matern 52* with 4 different values for the variance. The values were 0.1, 0.5, 1 and 2.

1.7 Conclusion

Bibliography

C. E. Rasmussen, C. K. I. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press, p. 85.

Guo-Xun Yuan, Chia-Hua Ho and Chih-Jen Lin (2021). "Recent Advances of Large-Scale Linear Classification". In: *Proceedings of the IEEE* 100, pp. 2584–2603.

KOWALCZYK, Alexandre (2014). *Linear Kernel: Why is it recommended for text classification*. URL: `https://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification/`.

Powell, Alex (2021). *Multivariate normal distribution*. URL: `https://towardsdatascience.com/intro-to-gaussian-process-regression-14f7c647d74d`.

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

Sreenivasa, Sushanth (2020). *Radial Basis Function (RBF) Kernel: The Go-To Kernel*. URL: `https://www.svm-tutorial.com/2014/10/svm-linear-kernel-good-text-classification/`.

APPENDIX