UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR ROBOTICS
AND COGNITIVE SYSTEMS

Course: Probabilistic Machine Learning (WS2020/21nils.r)
Prof. Dr. Elmar Rueckert & Nils Rottmann, M.Sc.
Assignment Report Nr.II
Date: 19. February 2021

Team Nr. 1
Rakesh Reddy Kondeti
Contact: rakesh.kondeti@student.uni-luebeck.de

Introduction

In this project we use probabilistic non-linear regression to predict unknown temperature values at a coastline, based on a real world data set, 'the California Coastal Regional Temp Field Dataset'. We will use our given Training-Data to construct a variety of different parameters such as hyper-parameters, Covariance-Matrices, and mean values.

Solution Method

0.1 Multivariate Conditional Distribution

From the lecture we obtain the conditional expectation (E) and the covariance Matrix (Σ) of a Multivariate Conditional Distribution as:

$$E(x \mid y) = \mu x + CB^{-1}\,(y - \mu y) \tag{1}$$

$$\Sigma x \mid y = A - CB^{-1}C^{T} \tag{2}$$

0.2 Gaussian Processes

To solve the above described problem we utilize Gaussian Processes. A Gaussian process can be described as a collection of random variables, each of which have a joint Gaussian distribution and is specified by its mean function $\mu(x)$ and its covariance function $k(x, x')$, and they can be used to estimate mean and variance for an unknown data point, based on known data (our training subset).

Consider our Training subset T = (x_i , y_i) = (**X**, y) with x_i (our input vector of dimension D=2 e.g. a position at the coastline, latitude and longitude), and y_i (e.g. the temperature at position x_i). And similarly our Testing subset $T* = (x*_i , y*_i) = (X*, y*)$

We are now able to rewrite our Equation (1) with our training data, but furthermore we expect our training set T to be corrupted with noise, which can be taken into account by adding σ 2 n I to the covariance matrix K

$$\mathrm{E}(y_* \mid y, X, X^*) = \mu_* + K^{T}_{*}(K + \sigma_n^2 I)^{-1}\,(y - \mu) \tag{3}$$

With the expectation value $\mu(.)$ of a certain input set and $K(., .)$ a covariance matrix determined using two input sets. As introduced in the previous chapter, for this task we are mainly interested in the expectation values of unknown values (y∗) consider: $\mu(X) = \mu$, $\mu(X^*) = \mu^*$, $K(X, X) = K$, $K(X, X^*) = K^*$

0.3 Kernel Functions

Generally, the most preferable way to group similar data is linear. But this isn't always possible to do in the non-linear data. We can project the non-linear data into higher dimensional space to make it linearly separable. But mapping of data into higher dimensional space is computationally expensive, as we have to perform calculations on data with new features included. Therefore by using Kernels we only need to compute the inner product of the data points. In this assignment only three type of kernels are used:

$$K = \sigma^2 \exp\left(\frac{-||x - x'||^2}{2l^2}\right) \tag{4}$$

$$K = \sigma^2 \exp\left(\frac{-2\sin(|x - x'|)^2/p}{l^2}\right) \tag{5}$$

$$K = \sigma_b^2 + \sigma^2(x - c)(x' - c) \quad (6)$$

0.4 Parameter Adjustment

In the previous subsection, the kernel functions have different parameters which are to be adjusted to the optimized value, as these parameters have an impact on the resulting predictions (effects covariance matrix and hence our conditional expectations). This fact makes it obvious that we want to find the best possible pair of parameters. To reach this goal we utilize Bayesian optimization to find values for the parameters σ and l which maximize the log-likelihood function. Luckily the needed derivation was already given as:

$$\frac{\partial \log(f(y|X,\Theta))}{\partial \Theta_j} = \frac{1}{2}((\propto \propto^T - K)^{-1}\frac{\partial K}{\partial \Theta_j})$$

Results

The given data set is already divided into training and test dataset. Fig. 1 shows the given data points. Here to reduce the computation time, the training data is limited to 1624 data points and testing data set is limited to 7488 data points. Figure 1 shows the training set and testing set after the reduction of data size.

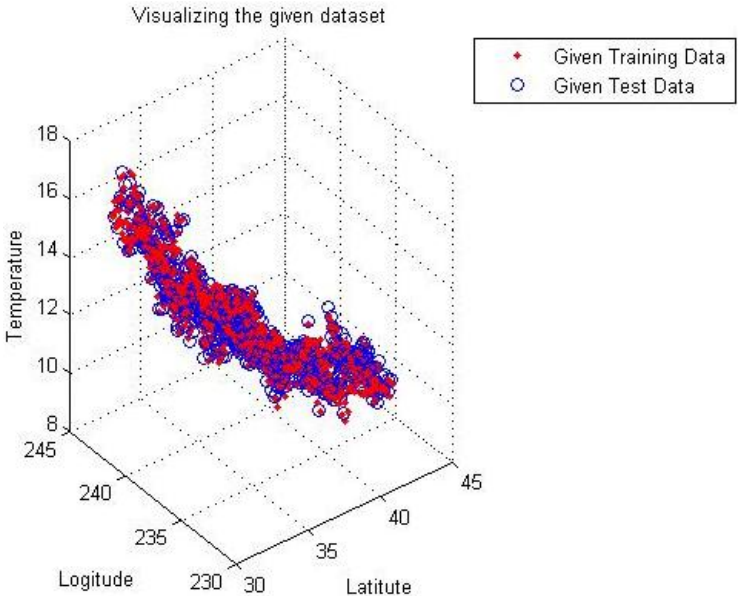

Figure 1: Visualization of the data dataset

Here, the GP using different kernel functions are applied on the data. Fig 2 shows the Gaussian Process with RBF kernel and mean = 0, having the MSE of 17.99, whereas when applied Periodic Kernel and mean = 0, it has MSE = 25.29, demonstrated in Fig 3. And Fig 4 shows, Gaussian process with Linear Kernel and mean = 0 produces MSE = 21.175

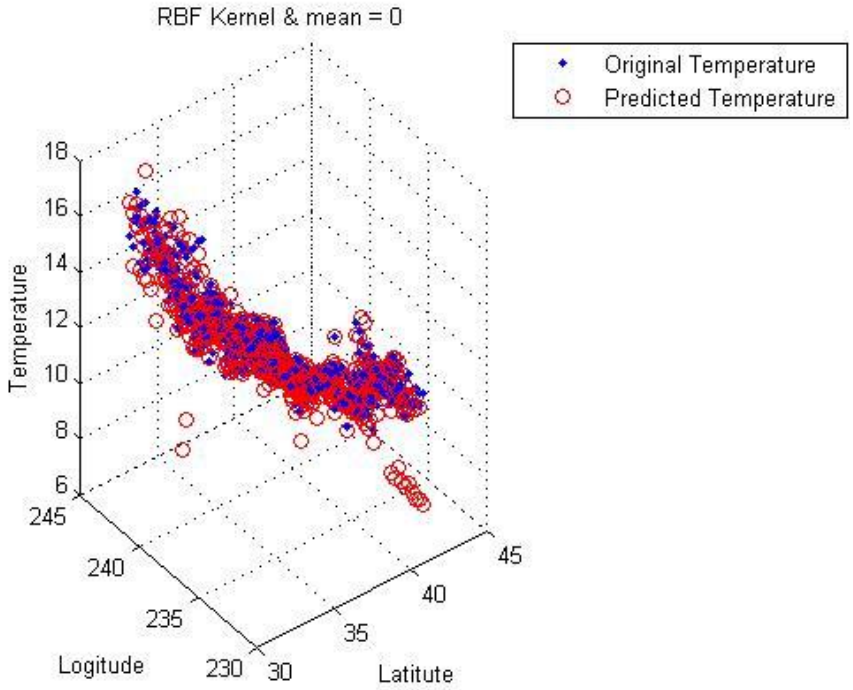

Figure 2: RBF Kernel and mean = 0

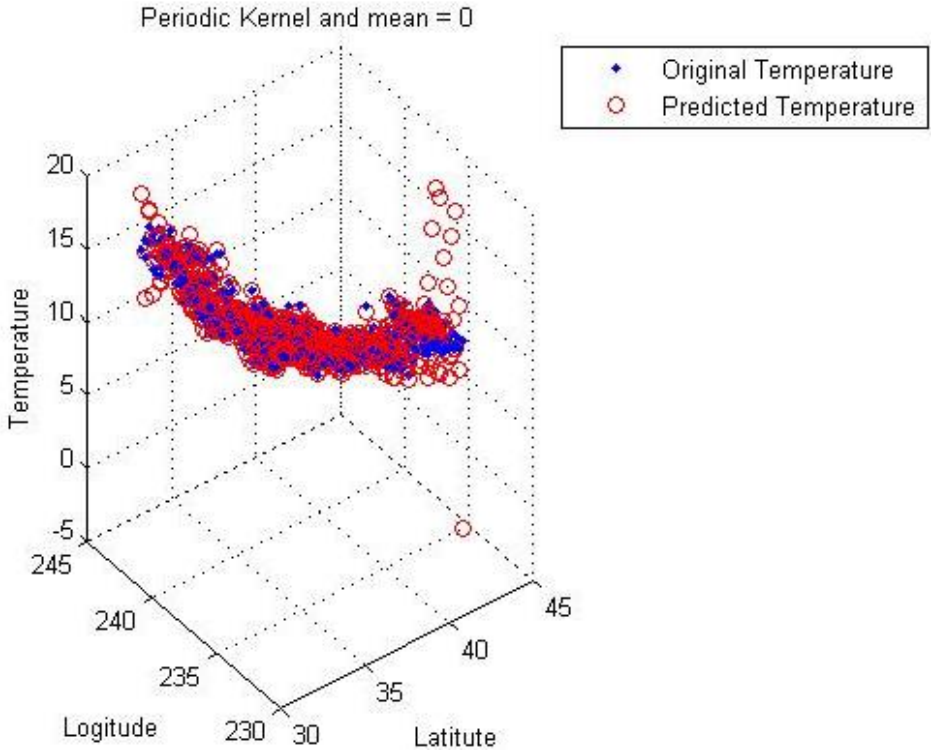

Figure 3: Periodic Kernel and mean = 0

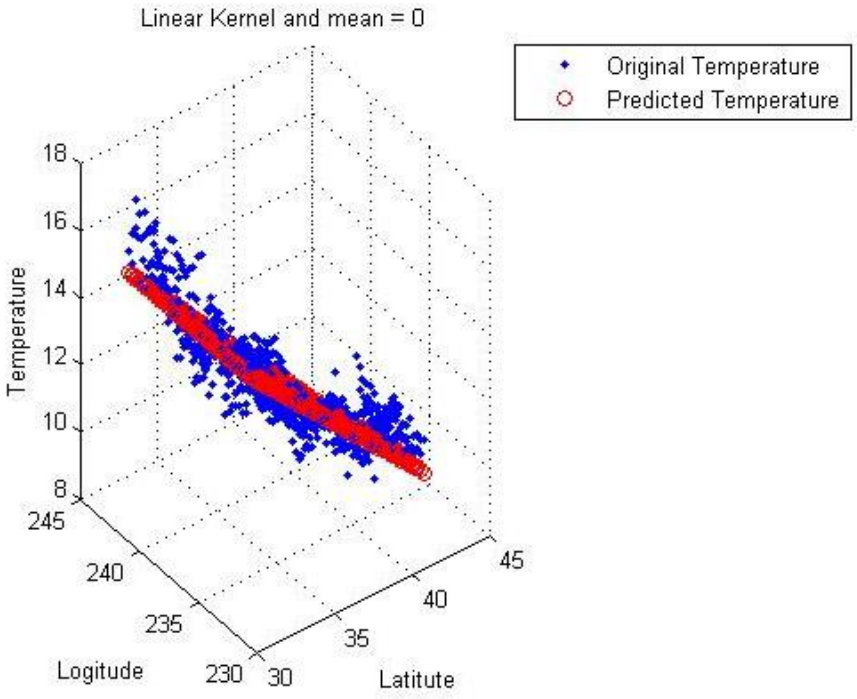

Figure 4: Linear Kernel and mean = 0

In the above the mean is assumed to be zero. Now the mean function is being changed to find its influence of the mean on predictions. After having the prior knowledge and mean is assigned the value of mean of the y's in the training set. The Gaussian process with RBF kernel is applied again and it produces an error of 5.501, which is shown in Fig5. Later, the mean function is chosen as the output from the linear or ridge regression, which produces MSE or 5.509, shown in Fig 6

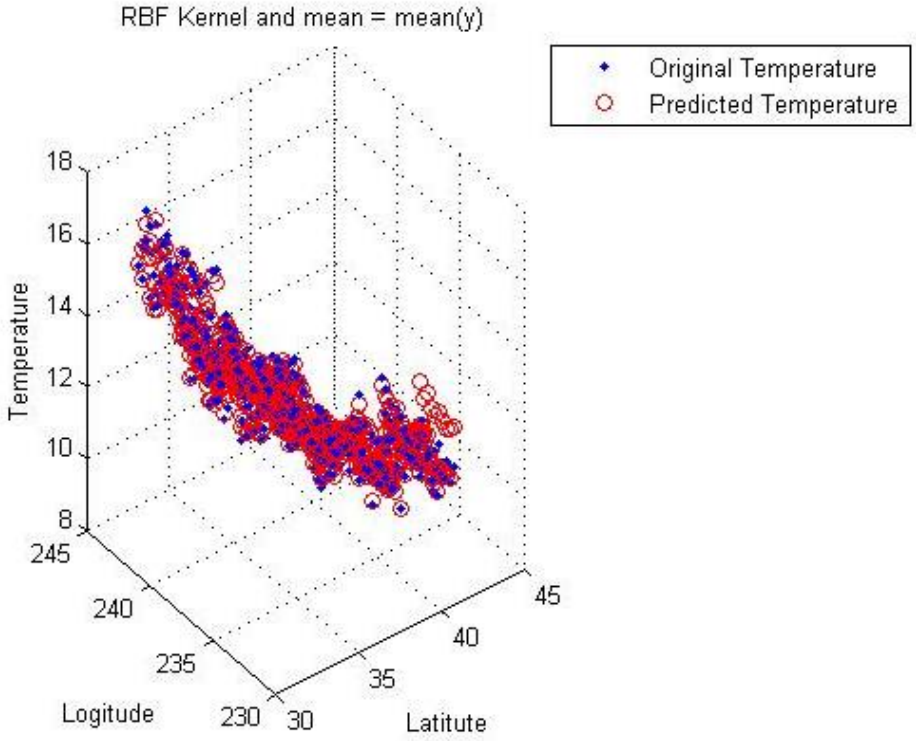

Figure 5: GP with RBF kernel and mean = mean(y)

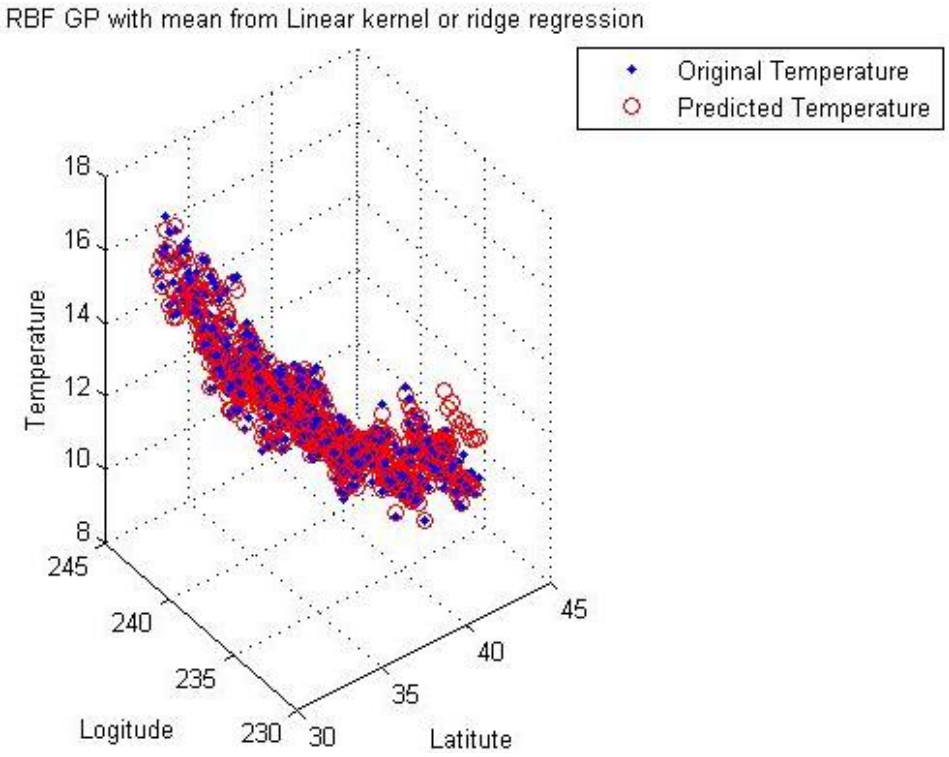

Figure 6: GP with RBF kernel and mean as the output from linear regression

Now as discussed in the solution method, the kernel functions have different parameters which are to be adjusted to the optimized value. Using Meta-Learning, we would now optimize by maximizing the loglikelihood of training data set. The output or optimized parameters are then plugged into the kernel functions to get the best fit and the least mse. After computing the optimized parameters, these values are plugged into RBF Kernel, which produced mse of 1.122 and is shown in Fig 7

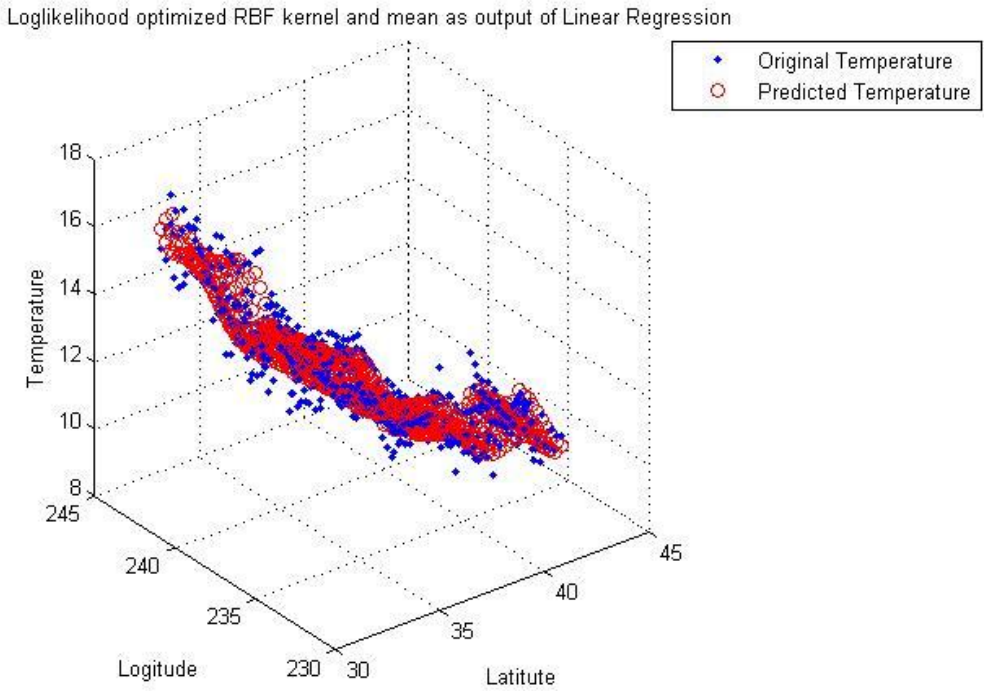

Figure 7 ; Loglikelihood optimized parameters - RBF Kernel and mean as output of linear regression

Conclusion

In conclusion, most of our GP's predictions match well to targets y of the test data set, as seen in the Graphs above. Also, it can be seen that mse produced by RBF kernel is 17.998, whereas mse in case of periodic kernel is 25.298 and in case of linear kernel is 21.117. However, it is to be noted that the initial parameters were chosen randomly and hence a strict comparison cannot be made. But, it is to be noted that the error produced with RBF kernel is less when compared to other kernels. Therefore RBF kernel fits well when compared to other two kernels.

RBF Kernel projects data into infinite dimensional space, (which is hard to visualize) works very well for non-linear regression and linear kernel works well only if the data is linearly separable (as expected, linear kernel produced a linear model in our graphs)

Keeping the parameters same for RBF Kernel and altering mean functions, it is found that the mse produced for mean = 0 is 17.99 and mse for mean = mean(y) is 5.501. For the same kernel and same parameters, when is mean is chosen as the output of linear regression, the mse is 5.509. However, it is to be noted that the mean mse has reduced by choosing mean function from the prior knowledge. Hence it is ideal to choose any mean functions (using prior knowledge) instead zero mean functions. Functions varied between the simplest case(computationally inexpensive) in which μ is simply set to 0, and μ being the true mean of our input training set is little expensive but yields better results by decreasing the prediction error

Meta Learning produces parameters, which decreases the error on predictions. In our case the error is reduced from mse of 5.5 (when used RBF Kernel and the mean function as the prior knowledge) to mse of 1.12 when used optimized parameters (with RBF Kernel and the mean function as the output from Linear regression). In general, when used optimized parameters, the prediction error is reduced when applied different kernels.

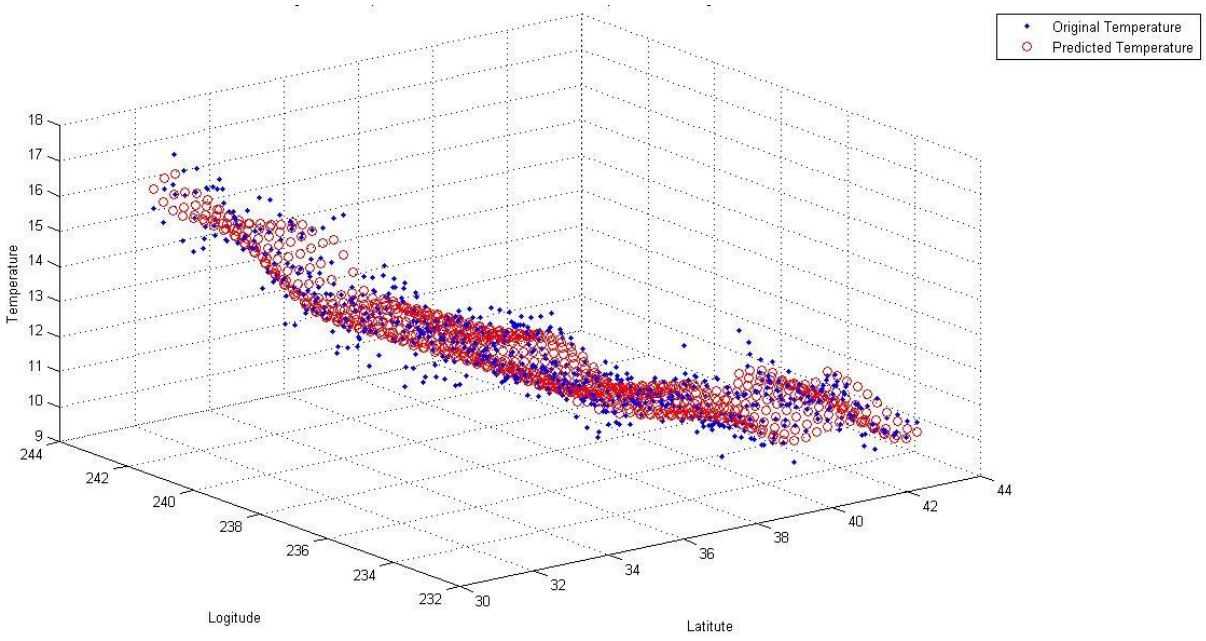

Figure 8: 3D view of RBF Kernel with optimized parameters