**Part1: Visualizing performance on the 1D dataset:**

**Linear kernel:**

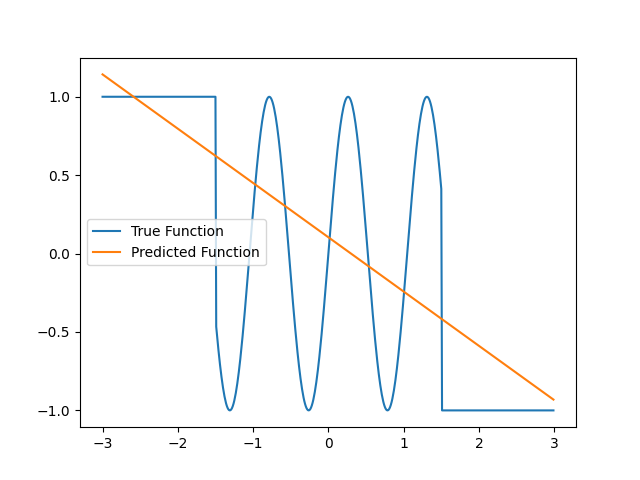
****

Figure 1: 1D dataset true function and predicted function using linear kernel

**RBF kernel:**

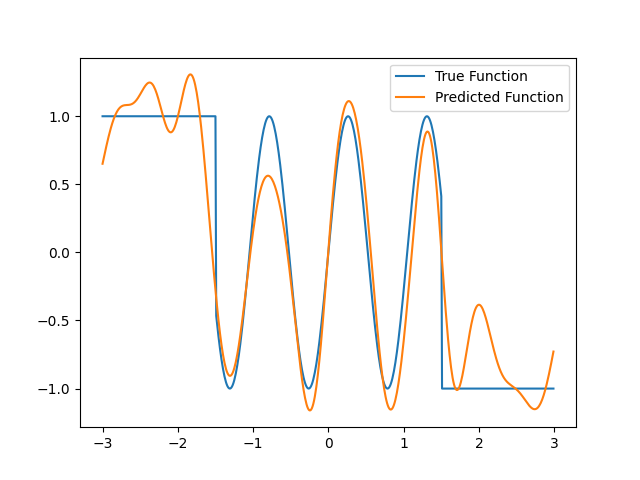
****

Figure 2: 1D dataset true function and predicted function using RBF kernel

**Part 2: Performance as a function of iterations:**

**Linear kernel:**

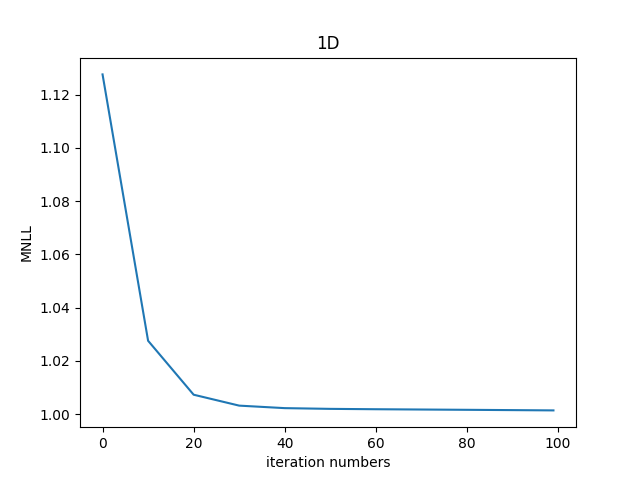
****

Figure 3: 1D dataset MNLL performance as a function of the number of the training iterations (using linear kernel)

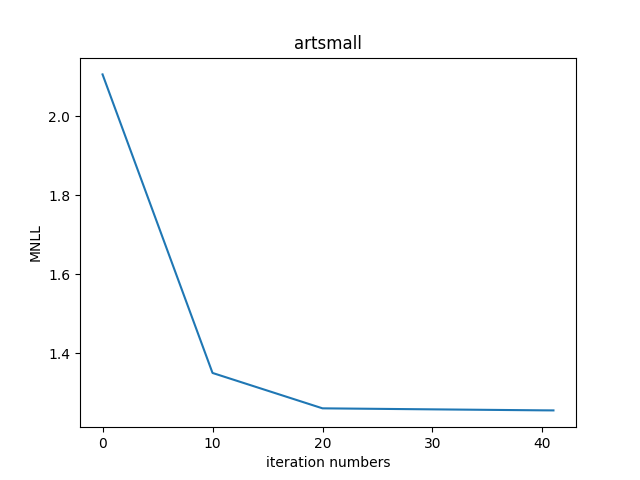


Figure 4: artsmall dataset MNLL performance as a function of the number of the training iterations (using linear kernel)

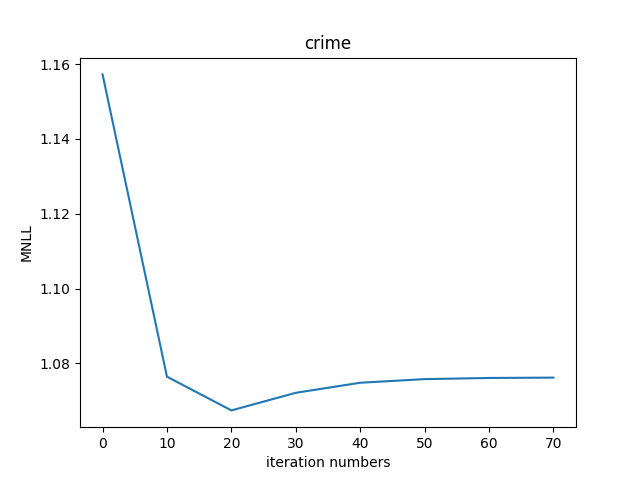


Figure 5: crime dataset MNLL performance as a function of the number of the training iterations (using linear kernel)

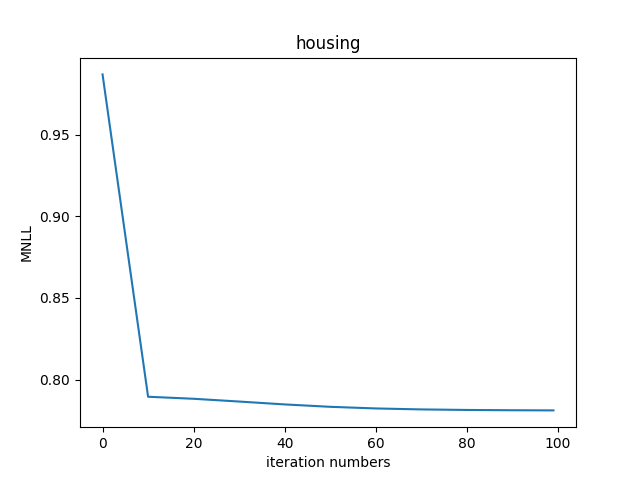


Figure 6: housing dataset MNLL performance as a function of the number of the training iterations (using linear kernel)

**RBF Kernel:**

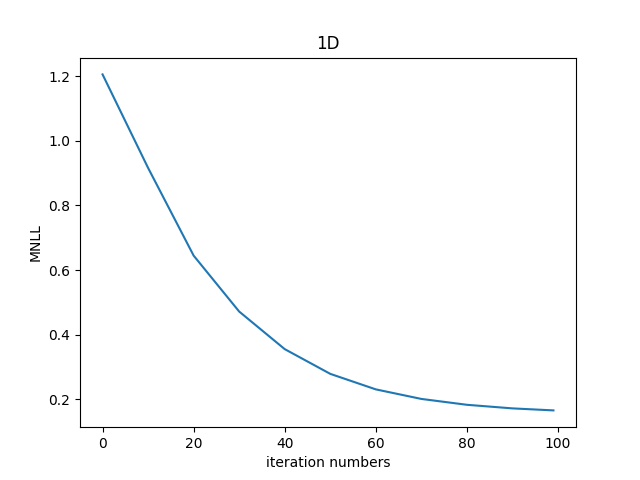
****

Figure 7: 1D dataset MNLL performance as a function of the number of the training iterations (using RBF kernel)

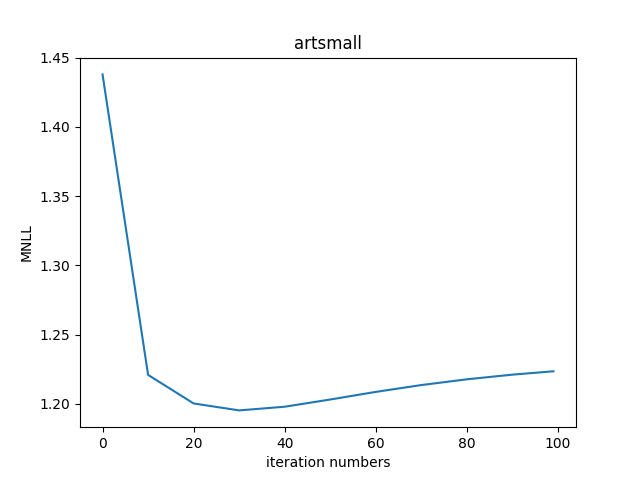


Figure 8: artsmall dataset MNLL performance as a function of the number of the training iterations (using RBF kernel)

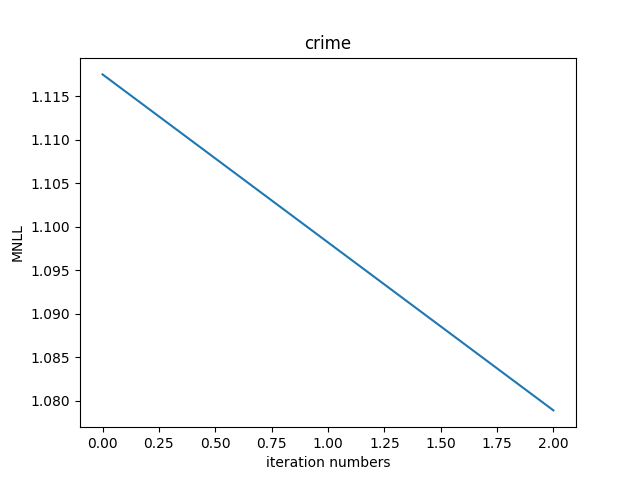
****

Figure 9: crime dataset MNLL performance as a function of the number of the training iterations (using RBF kernel)

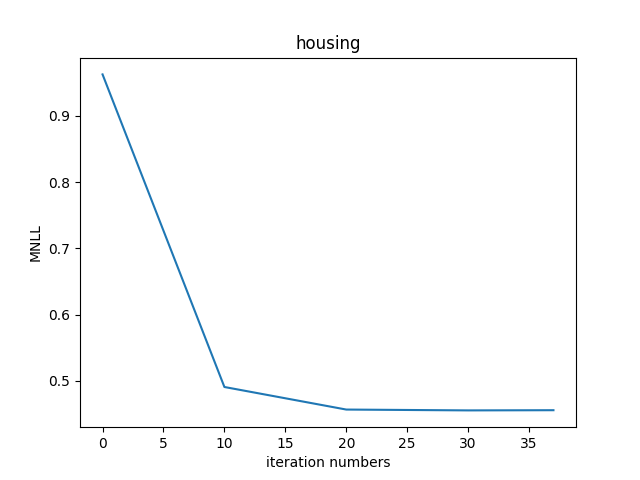


Figure 10: housing dataset MNLL performance as a function of the number of the training iterations (using RBF kernel)

**Part 3: Comparison to Bayesian Linear Regression**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **dataset** | **BLR MSE** | **BLR alpha** | **BLR beta** | **Linear kernel MSE** | **Linear kernel alpha** | **Linear kernel beta** | **RBF kernel MSE** | **RBF kernel alpha** | **RBF kernel beta** | **RBF kernel s** |
| **crime** | 0.5 | 357.5 | 2.6 | 0.503 | 368.833 | 2.602 | 0.515 | 1.371 | 2.699 | 15.560 |
| **artsmall** | 0.716 | 141.4 | 4.23 | 0.705 | 146.866 | 4.072 | 0.678 | 0.424 | 6.493 | 16.214 |
| **housing** | 0.288 | 20.4 | 4 | 0.288 | 21.362 | 3.995 | 0.178 | 0.372 | 12.993 | 4.494 |
| **1D** | 0.39 | 7.5 | 1.9 | 0.411 | 2.384 | 1.926 | 0.071 | 1.258 | 16.674 | 0.222 |

**Part 4: Discuss the results**

**With all these results recorded, what can you observe w.r.t. the performance of the algorithms?**

Performance of BLR and GP with linear kernel are the same but GP with RBF kernel is better than the other too for all of the data sets respect to MSE and MNLL.

**Are the BLR and GP with linear kernel behaving similarly w.r.t. α, β, MSE as expected?**

Yes, they both are the same with respect to alpha, beta and MSE

**How does the performance of GP compare when changing RBF vs. linear kernel?**

GP is better using RBF kernel rather than linear kernel because it fit’s the data better

**What are potential advantages or disadvantages of each method?**

The advantages of Gaussian processes:

* The prediction interpolates the observations (at least for regular kernels).
* The prediction is probabilistic (Gaussian) so that one can compute empirical confidence intervals and decide based on those if one should refit (online fitting, adaptive fitting) the prediction in some region of interest.
* Versatile: different kernels can be specified. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of Gaussian processes:

* They are not sparse, i.e., they use the whole samples/features information to perform the prediction.
* They lose efficiency in high dimensional spaces – namely when the number of features exceeds a few dozens.

Comparison of RBF and linear kernel: in general RBF is more accurate than linear kernel but linear kernel is faster.

Cite: https://scikit-learn.org/stable/modules/gaussian\_process.html