# Gaussian Process Regression and Bayesian Optimization

Homework Assignment by Arno Strouwen

This document contains only the answers. For the questions and code used to generate the answers, see [my GitHub](https://github.com/arno-training-material/Master-of-Statistics-and-Data-Science/tree/main/Advanced%20Topics%20in%20Data%20Science).

## a)

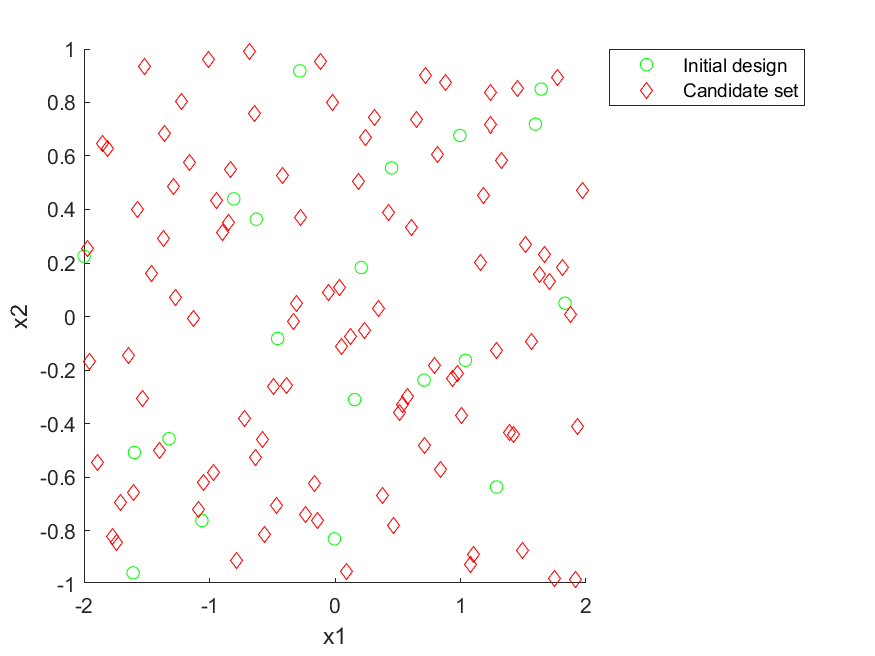


Figure 1) Latin hypercube design for both the initial design and the candidate set.

Per the definition of Latin hypercube sampling, no points in the candidate set have the same x1 value, nor the same x2 value. Moreover, if the x1-axis was divided into 100 pieces of the same length, each piece would contain one sample, and similarly so for the x2-axis.

However, out of all possible Latin hypercubes, there are some good and bad designs. For example, a design where all points lay on a straight line between (-2,-1) and (2,1) can be a Latin hypercube. Such a Latin hypercube would obviously not be a good design to build a regression model, because large parts of the design space would then have no candidate points.

The DACE toolbox seems to construct a random Latin hypercube from the class of possible Latin hypercubes. In my opinion, the candidate set is still clustered quite a bit, as shown in Figure 1, this could be improved by the techniques of [1], where the Audze-Eglais distance is used to spread out the design points more. Furthermore, this technique could also be used to optimize the location of the candidate set, while taking into account the location of the initial design.

## Hyperparameter verification and initial fit

The DACE toolbox requires an initial guess for the parameters of the squared exponential kernel.  
Through trial and error, I found that initial values of 1, as well as lower and upper bounds 10 times larger and smaller than the initial values gave a reasonable initial fit. A reasonable fit here means that the hyperparameters do not lay on the bounds. Of course, taking the bounds very far apart will also cause the hyperparameters to not lay on the bounds. But by doing so, e.g., by making the bounds a factor thousand different from the initial guess, instead of 10, I noticed that the mean squared error of the predicted values in the candidate set started to increase, possibly because the Gaussian process decayed quickly to the mean of the process, when moving away from a point in the initial set.

The mean of the Gaussian process is a constant function, which is also inferred from the data, but no initial value for this fitting parameter is required.

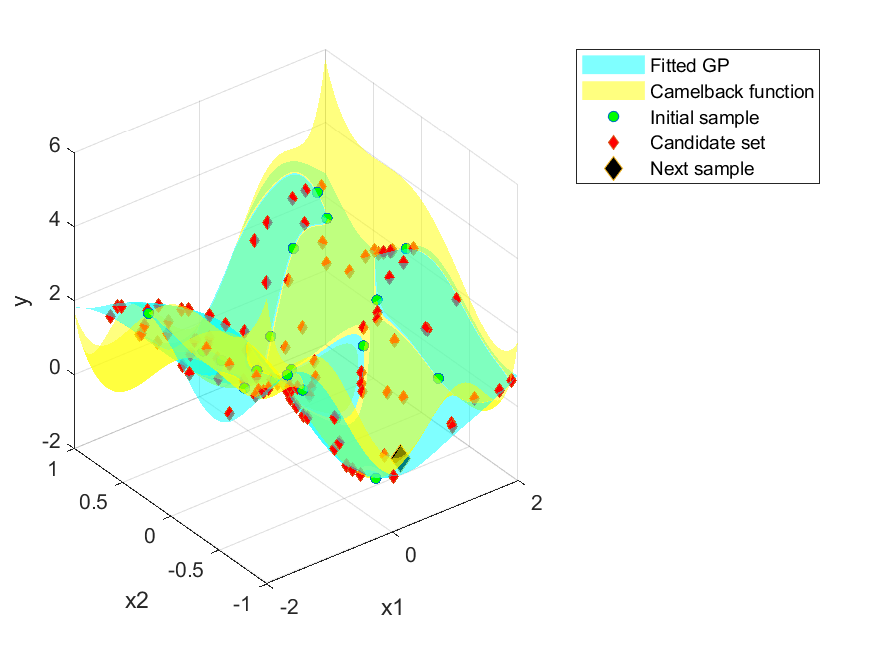


Figure 2) Gaussian Process fit using the Initial sample, as well as the first sample taken from the candidate set in the Bayesian optimization process. The squared exponential kernel has hyperparameters, theta: [1.0905 0.7384],

The static Figure 2 is somewhat hard to read, running the accompanying MATLAB code will give an interactive figure.

The green dots lay on both surfaces, the diamonds lay on the fitted Gaussian Process surface. We can visually verify that the first point from the candidate set chosen by the Bayesian optimization algorithm is a reasonable choice.

## b+c)

## 

Figure 3) Points chosen by Bayesian optimization. True minimum compared to minimum found by Bayesian optimization.

Figure 3 shows that the points chosen by the Bayesian optimization algorithm are mostly located in the bottom-right quarter and top-left quarter, while staying away from the top-right corner and bottom-left corner. This is reasonable since the camelback function has high function values in these two corners. The location of the minimum found by the algorithm is located quite far away from the true minimum. This is because it is in the other “hump” of the camel, than the hump that the true minimum is located in. However, the point chosen by Bayesian optimization is the minimum over the candidate set and initial set, it could thus not have found a lower function value. Table 1 shows that the found minimum is still 9% worse than the true optimum.

Table 1) Summary of minimum location and value.

|  |  |  |  |
| --- | --- | --- | --- |
|  | x1 | X2 | y |
| True min | 0.0977 | -0.6973 | -1.0294 |
| Candidate + Initial min | -0.0183 | 0.7990 | -0.9367 |
| Bayesian optimization min | -0.0183 | 0.7990 | -0.9367 |

## d)

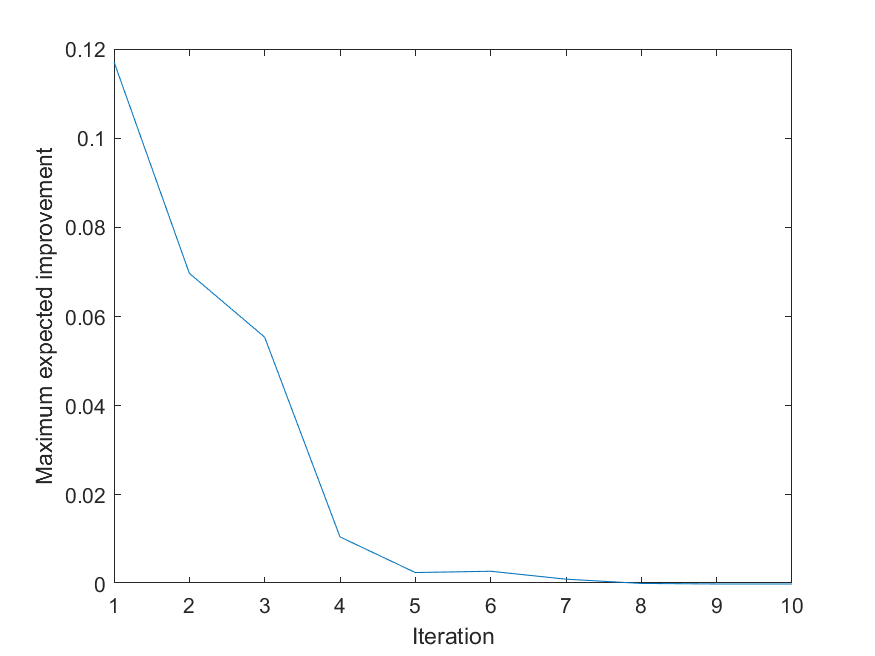


Figure 4) Expected improvement each iteration of the Bayesian optimization algorithm.

The Expected improvement generally goes down as the iteration number increases. However, this decrease is not monotonic, e.g., between iteration 5 and 6 there is an increase. This is because between each iteration, a new model is fitted. The hyper-parameters of the kernel are fine-tuned, and even if these values were to remain the same, the posterior of the Gaussian Process would be updated. This newly fitted model might mark certain regions in the design space as interesting, which the previous model might not have fitted correctly (particularly if the previous model also misjudged its confidence in the fit, by assigning a low MSE to the region).

## e)

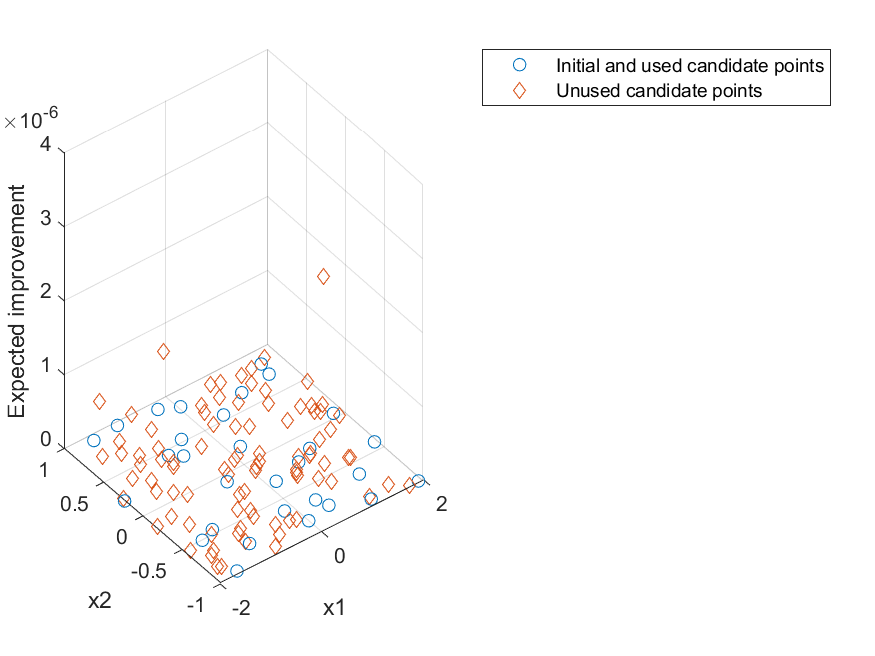


Figure 5) Expected improvement over points in initial and candidate set.

Figure 5 shows that all the points have an expected improvement almost equal to zero. The points of the initial sample and the points chosen during Bayesian optimization have an expected improvement equal to zero. Figures 4 and 5 show there is little reason to continue the Bayesian optimization algorithm with the current candidate set. However, as discussed in a) increasing the simulation budget by considering a better (or even just larger) candidate set might lead to improvement.

## Final fit)

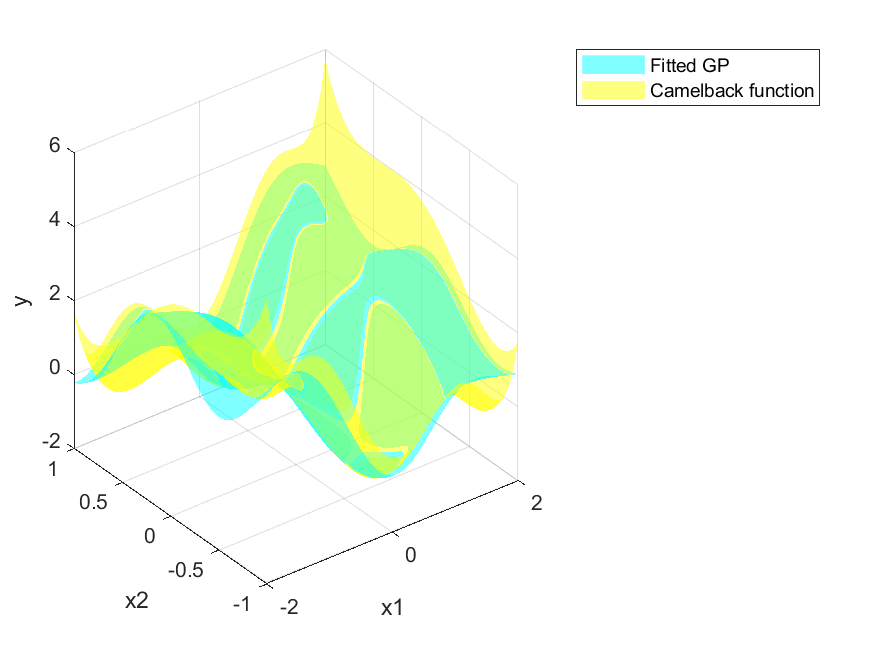


Figure 6) Fitted Gaussian process at the end of the Bayesian optimization algorithm.

Compared to Figure 2, the fit in Figure 6 is much better, particularly around the two humps of the camel, but less so at the top-right and bottom-left corner of the design space. This is reasonable, since the Bayesian optimization algorithm will focus on accurately fitting the regions where there is hope for improvement, and not regions where there is no expectation of improvement.

## References

[1]: Stuart Bates, Johann Sienz, and Vassili Toropov. "Formulation of the Optimal Latin Hypercube Design of Experiments Using a Permutation Genetic Algorithm", 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics & Materials Conference, Structures, Structural Dynamics, and Materials and Co-located Conferences, <https://doi.org/10.2514/6.2004-2011>