Lab 4: Gaussian Processes

Patrick de Kok

October 21, 2012

In this exercise, I have learnt how to use Gaussian processes on sparse data, such as the `chirps.mat` dataset. The data has been split in a train set of 12 data points and a test set of 3 data points. The data points have been randomly divided into either set.

1 The Gaussian distribution

In the first assignment, I have looked at a Gaussian distribution $\mathcal{G}_1 = \mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ over two-dimensional vectors of real numbers. The mean $\boldsymbol{\mu}$ has been set to 0, and the covariance $\boldsymbol{\Sigma}$ to $\begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$. The contour plot of $\mathcal{G}_1$ has been plotted for input $\langle x_1, x_2 \rangle \in \boldsymbol{X}$ where $-4 \leqslant x_1 \leqslant 4$ and $-4 \leqslant x_2 \leqslant 4$ in the upper part of Figure 1. The lower part of Figure 1 presents the marginal distribution over x_1. This has been computed as $\mathcal{N}(x_1; \mu_1, \Sigma_{1,1}) = \mathcal{N}(x_1; 0, 1)$.

In Figure 2, the marginal distribution over x_1 for the follow values of x_2 are given, from top to bottom: $-3, -2, -1, 0, 1, 2, 3$. Not that these seven plots are identical to the subfigure in Figure 1. This is not really surprising, when looking at the computation:

$$\begin{aligned} p(x_1 | x_2 = v) &= \mathcal{N}(x_1; \boldsymbol{\mu}_1 - \Lambda_{1,1}^{-1}\Lambda_{1,2}(x_2 - \mu_2), \Lambda_{1,1}^{-1}) \\ &= \mathcal{N}(x_1; 0 - 1 \times 0 \times v, 1) \\ &= \mathcal{N}(x_1; 0, 1) \\ &= p(x_1) \end{aligned}$$

with $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$.

Another distribution I have looked at, is $\mathcal{G}_2 = \mathcal{N}(\boldsymbol{X}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$. The surface plot is given in Figure 3. Although the marginal distribution of $\mathcal{G}_1$ over x_1 and $\mathcal{G}_2$ are equal, their conditional distributions

Figure 1: Plot of the Gaussian distribution with mean $\boldsymbol{\mu} = 0$ and covariance $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0 \\ 0 & 3 \end{bmatrix}$. In the plot above, one can see the contour plot of $\mathcal{G}_1 = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with isolines for points with probability density $0.01, 0.02, \ldots 0.09$. The lower plot shows the marginal distribution over x_1.

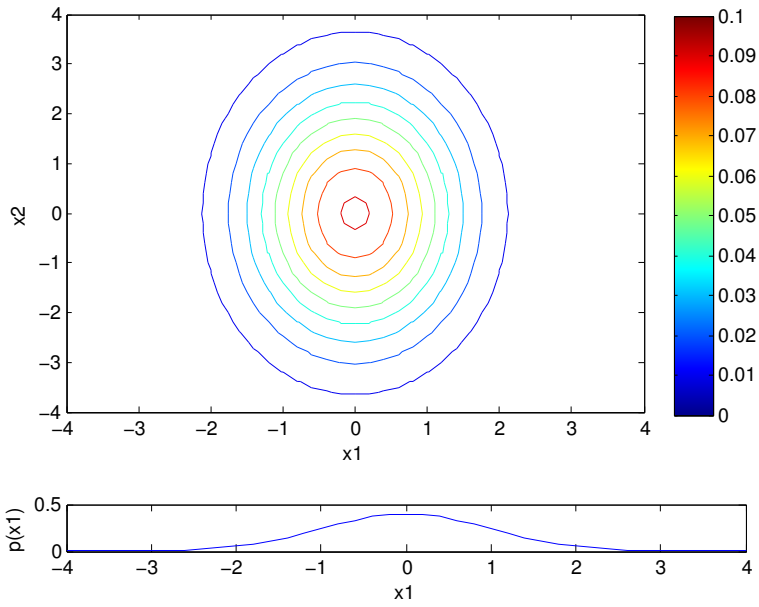


Figure 2: Plots of the conditional distribution over x_1 given various values for x_2 given the Gaussian distribution $\mathcal{G}_1$. The conditional distribution over x_1 is plotted from top to bottom for $x_2 = -3$, $x_2 = -2$, $x_2 = -1$, $x_2 = 0$, $x_2 = 1$, $x_2 = 2$ and $x_2 = 3$.

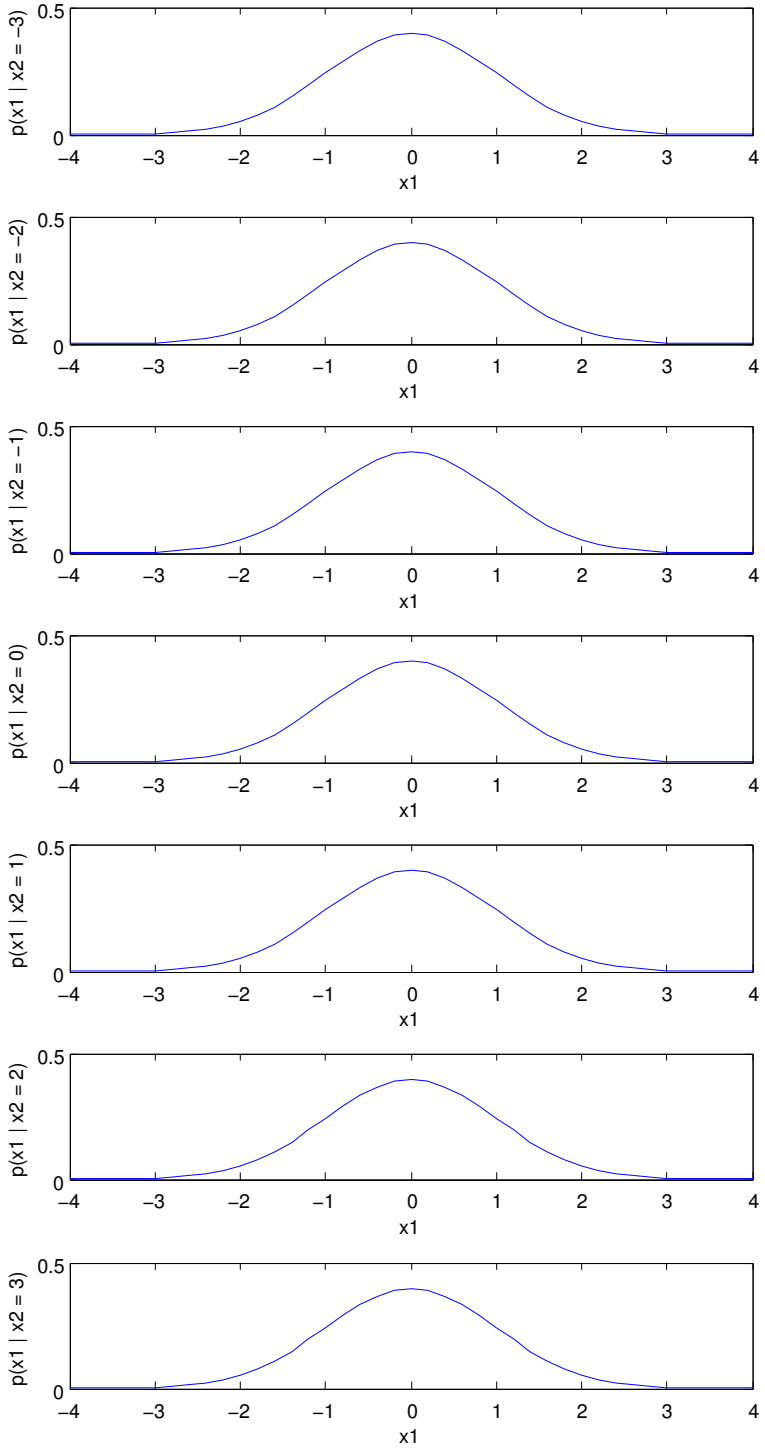
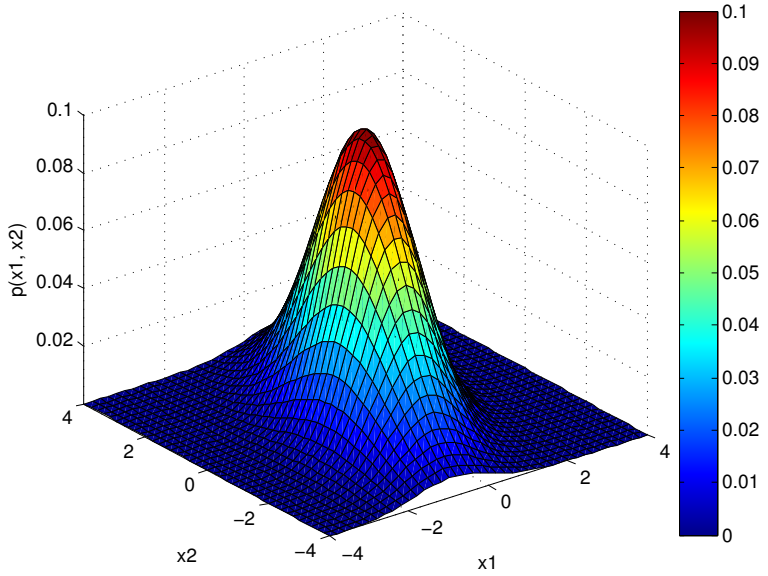


Figure 3: Surface plot of the Gaussian distribution $\mathcal{G}_2 = \mathcal{N}(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu} = 0$ and covariance $\boldsymbol{\Sigma} = \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$.



over x_1 given x_2 are not. For $x_2 = v$, the conditional distribution is:

$$\begin{aligned} p(x_1|x_2 = v) &= \mathcal{N}(x_1; \boldsymbol{\mu}_1 - \Lambda_{1,1}^{-1}\Lambda_{1,2}(x_2 - \mu_2), \Lambda_{1,1}^{-1}) \\ &= \mathcal{N}(x_1; 0 - 1.1952 \times -0.2789 \times x_2, 1.1952) \\ &= \mathcal{N}(x_1; 0.3333x_2, 1.1952) \end{aligned}$$

The conditional distribution over x_1 for $x_2 \in \{-3, -2, -1, 0, 1, 2, 3\}$ is presented in Figure 4.

2 Gaussian Processes

In this exercise, the temperature must be predicted from the number of chirps per time unit crickets produce during summer. The dataset describing the data to learn from, `chirps.mat`, contains only 15 chirp-temperature pairs. Because of this small dataset size, the test set contains 1 data point, and 14-fold cross-validation is applied to acquire the best learner. A plot of the dataset is given in Figure 5.

I have written an implementation of Gaussian process according to Algorithm 1 from the assignment in MATLAB in `gaussian_process.m`. The implementation has been done as close as possible to the pseudocode. The `gaussian_process` function has the following inputs:

Figure 4: Plots of the conditional distribution over x_1 given various values for x_2 given the Gaussian distribution $\mathcal{G}_2$. The conditional distribution over x_1 is plotted from top to bottom for $x_2 = -3$, $x_2 = -2$, $x_2 = -1$, $x_2 = 0$, $x_2 = 1$, $x_2 = 2$ and $x_2 = 3$.

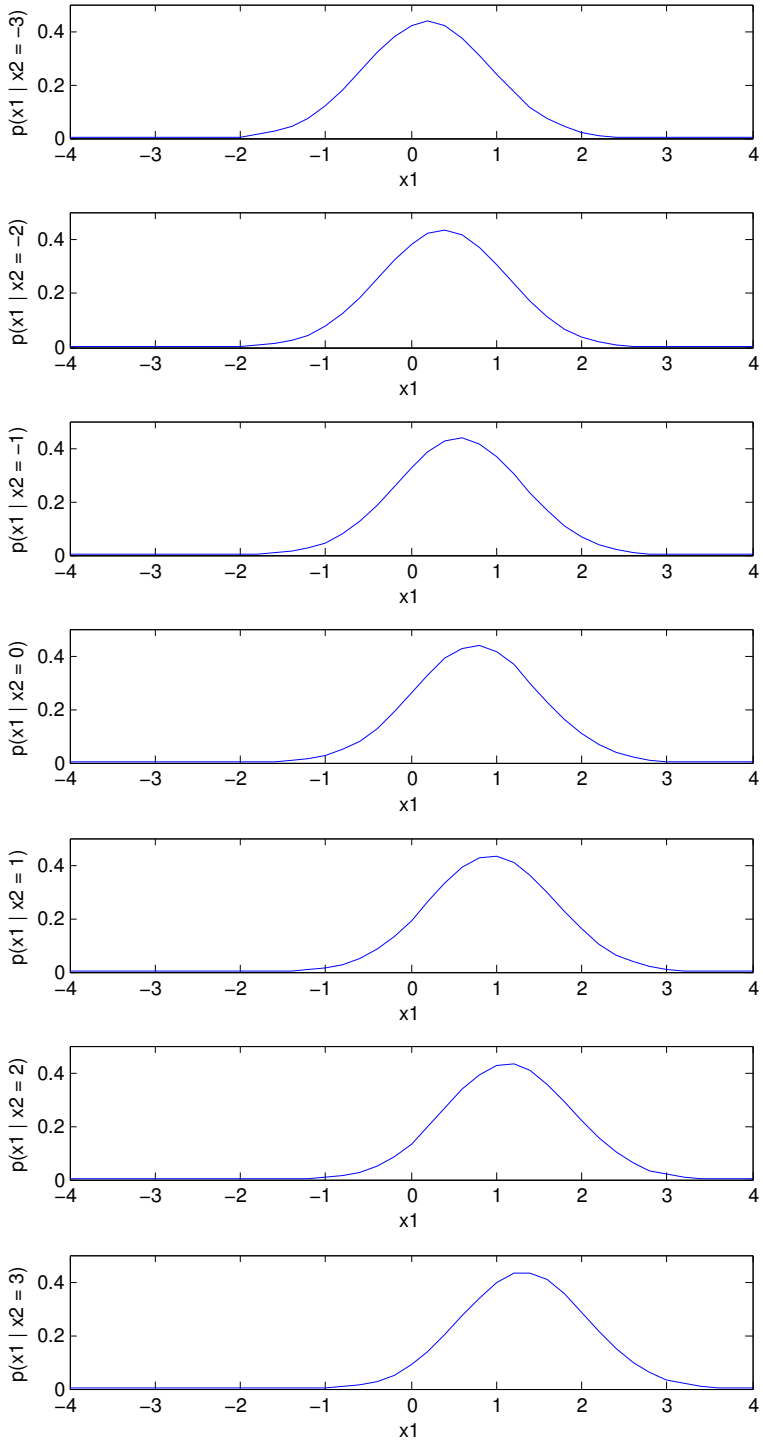
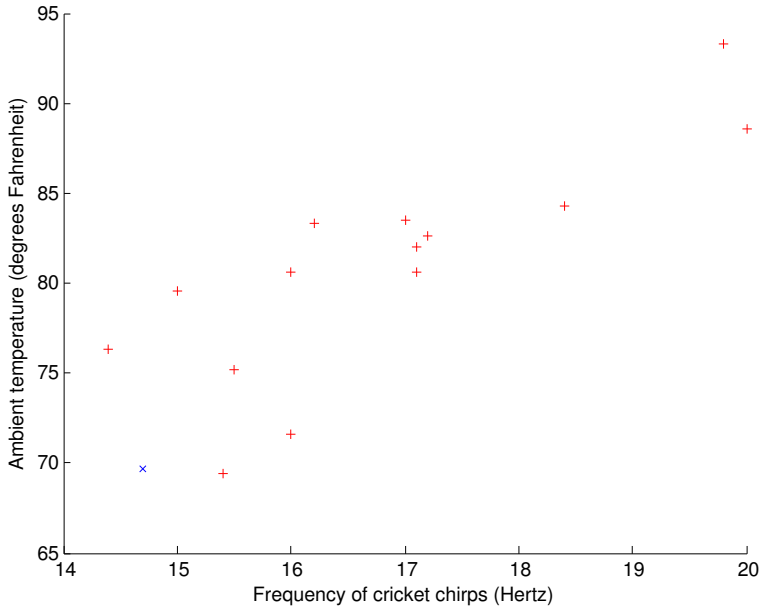


Figure 5: A plot of the dataset `chirps.mat`. The red "+"'s represent the train data points. The blue "×"' represent the test data point.



- `X`: an `N` by `D` matrix, representing `N` data points of dimensionality `D`, which are used for training.
- `t`: an `N` by `1` vector. These are the target values, which the Gaussian process should approximate.
- `k`: the kernel function. This function should accept two arguments, both `1` by `D` dimensional vectors.
- `sigma2`: the squared variance on the target value. This represents the amount of noise in the data.
- `x_`: a data point, for which the Gaussian process shall predict the target value.

As in the pseudocode, the provided MATLAB implementation returns three values:

- `f_`: the most likely target value, given the input properties.
- `sigma_2`: the squared variance on the prediction.
- `LLog`: the log-likelihood of the provided train data under the model. This can also be seen as a measure how well the model can "explain" the data.

The exercise tells that `sigma2` should have a small value. Therefore it is set to 10^{-7}, unless stated differently.

The kernel function $k(\boldsymbol{x}, \boldsymbol{x}')$ as defined by Equation 9 is returned by calling `squared_exponential` with specific values for θ and l, called the hyperparameters. These parameters represent a quantity associated with the spread of the temperature and chirp frequency in the train data set.

Learning from data in terms of Gaussian processes means to find the Gaussian process under which the dataset is most likely to occur. This means that the program needs to find values for the hyperparameters of the kernel function k which maximize the (log-)likelihood. In this case, the software has to optimize the log-likelihood by finding the correct values of θ and l. This can be done in two ways:

- Gradient ascent. For this, I need to compute the gradient with respect to θ and l of the following expression:

$$-\frac{1}{2}\boldsymbol{t}^\top(\boldsymbol{K}\backslash\boldsymbol{t}) - \sum_i \log \operatorname{cholesky}(\boldsymbol{K}) - \frac{N}{2}\log 2\pi$$

  where $\boldsymbol{t}$ are the labels of the train data, N the size of the train data set, and $\boldsymbol{K}$ a matrix with values $K_{ij} = k(x_i, x_j)$ for elements x_i, x_j in the dataset $\boldsymbol{X}$.

  Although this will give the best result, finding the gradient of this expression is very hard.

- Grid search. With a grid search, I will look for the highest log-likelihood of the Gaussian process for values of θ and l in a certain range and step size.

  When applying this method, the best combination of the hyperparameters might not be found, because that optimum might be in between two steps, or outside the searching range.

I have also searched for the hyperparameters which minimize the estimated error, as well as maximize the variance.

The estimated error of the Gaussian process will be measured by the mean square error of the predicted mean temperature, associated with the input chirp frequency. For n targets $\boldsymbol{t}$ and their corresponding predictions $\boldsymbol{t}^*$, the error is given by:

$$\mathrm{E}(\boldsymbol{t}, \boldsymbol{t}^*) = \frac{1}{n}\sum_i^n (t_i - t_i^*)^2$$

A performance measure should increase when the inspected subject behaves (in this case, generalizes) more to our likings. Therefore, I will define the performance measure as the negative of the estimated error.

Another noteworthy thing is that the hyperparameterization which optimize the log-likelihood, are quite often similar to that which optimizes the variance. Because this parameterization characterizes the dataset best, the Gaussian process is fairly sure about its predictions. I have decided to apply the grid search approach. The lower and upper bounds on the search range have been based on the data in terms of the average temperature and chirps among the data points. Let $\bar{\theta}$ and $\bar{l}$ be the average temperature and average chirp frequency among the test-and-validation set. For ease of reading, I will use $\bar{\boldsymbol{\theta}} = \begin{bmatrix} \bar{\theta} \\ \bar{l} \end{bmatrix}$ and $\boldsymbol{\theta} = \begin{bmatrix} \theta \\ l \end{bmatrix}$ as shorthands. Initially, I have looked at the joined ranges $[0.1\bar{\boldsymbol{\theta}}, \bar{\boldsymbol{\theta}}]$ with step size $\mathrm{d}\boldsymbol{\theta} = 0.1\bar{\boldsymbol{\theta}}$, and $(\bar{\boldsymbol{\theta}}, 100\bar{\boldsymbol{\theta}}]$ with step size $\mathrm{d}\boldsymbol{\theta} = \bar{\boldsymbol{\theta}}$[1].

The results of the grid search, for noise levels $\sigma^2 = 10^{-7}$ and $\sigma^2 = 10^{-4}$, and tests sets of 1 up to 5 items, are presented in Table 1 and Table 2. In the first column, the noise level is indicated. The second column contains data on how many test items have been used. In the third and fourth column can be found which heuristic has been optimized, and what its optimal value is. Optimization has been done as described above; log-likelihood and variance are maximized, while the expected error is minimized. In the sixth and seventh column are recorded the optimal values $\hat{\boldsymbol{\theta}}$ for the heuristic. The last column presents the performance.

In Figure 6, one can find the expected mean and its variance for input values 12 through 22 for various values of $\hat{\boldsymbol{\theta}}$. The expected mean and variance for the parameterization which optimizes the log-likelihood is drawn as a red line. The expected mean and variance generated by $\hat{\boldsymbol{\theta}}$ for a minimal error is supplied in green. The blue line is generated by $\hat{\boldsymbol{\theta}}$ which maximize the variance. The corresponding values for $\hat{\boldsymbol{\theta}} = \begin{bmatrix} \hat{\theta} \\ \hat{l} \end{bmatrix}$ can be found in Table 1.

It is notable that, although the variance is so low that it cannot be distinguished from the mean for most part of the lines in Figure 6, it (even visually) increases for inputs which are further from train data points. This is not surprising; the further you are from known data, the less precise your estimation will be. Because variance is a measure of confidence, this is only expected.

However, I cannot explain well why the variance in general is close to zero. This can only be because the noise level is very low, but even then I would expect the model to be less confident about its predictions.

The kernel function's parameters are of a big importance on the expected

[1] At first, I wanted to look at the range $[0.1\bar{\boldsymbol{\theta}}, 100\bar{\boldsymbol{\theta}}]$ with step size $0.1\bar{\boldsymbol{\theta}}$. Combining this with the cross validation, I would test 14 million Gaussian processes. After I have implemented the cross-validation part in a multithreading way and ran it overnight on a quadcore pc with no other big tasks, and the algorithm had not finished in the afternoon, I came to the conclusion this must not have been the idea.

Table 1: The performance of Gaussian processes based on differently selected hyperparameters $\hat{\theta}$ and $\hat{l}$ with noise level $\sigma^2 = 10^{-7}$.

σ^2	Test items	Criterion	Cr. value	$\hat{\theta}$	$\hat{l}$	Performance
10^{-7}	1	Log-likelihood	-177771595.768096	$0.3\bar{\theta}$	$95\bar{l}$	-75.353693
		Expected error	3.067816	$0.7\bar{\theta}$	$\bar{l}$	-112.484638
		Variance	1.050682	$0.1\bar{\theta}$	$98\bar{l}$	-75.563232
	2	Log-likelihood	-175492431.801194	$0.3\bar{\theta}$	$97\bar{l}$	-79.122924
		Expected error	7.863844	$0.1\bar{\theta}$	$\bar{l}$	-101.478213
		Variance	8.8462×10^{-8}	$0.2\bar{\theta}$	$50\bar{l}$	-79.137309
	3	Log-likelihood	-4083462.049829	$0.4\bar{\theta}$	$98\bar{l}$	-78.930534
		Expected error	7.441296	$0.1\bar{\theta}$	$0.9\bar{l}$	-90.614656
		Variance	25.881577	$0.1\bar{\theta}$	$0.9\bar{l}$	-78.980332
	4	Log-likelihood	-4009203.062108	$0.3\bar{\theta}$	$98\bar{l}$	-79.020887
		Expected error	06459058	$68\bar{\theta}$	$93\bar{l}$	-95.228785
		Variance	11.443654	$0.1\bar{\theta}$	$98\bar{l}$	-78.417271
	5	Log-likelihood	-3920114.918473	$0.3\bar{\theta}$	$98\bar{l}$	-71.076326
		Expected error	1.664945	$1.8\bar{\theta}$	$0.4\bar{l}$	-65.334042
		Variance	13.491816	$0.1\bar{\theta}$	$98\bar{l}$	-73.171125

error. A plot of the mean square error as a function of both θ and l is provided in Figure 7.

3 Conclusion

The `chirps.mat` dataset has been analyzed by means of a Gaussian process. Optimal values for the hyperparameters have been selected with a grid search and leave-one-out cross-validation according to three criteria:

1. Maximizing the log-likelihood of the train set.
2. Minimizing the mean square error of the validation set.
3. Maximizing the variance of the validation set.

The hyperparameters have then been evaluated for a noise level of 10^{-7} and 10^{-4}, as well as test data sets of size 1 up to 5.

In Table 1 and Table 2 one can see that the performance of the Gaussian process almost everywhere is increased with more noise. In the experiments where the performance increases by adding noise, this might signify that the model otherwise is overfitted. Adding noise worsens the performance when the train set has been reduced to 10 items, and the test set contains 5. This might be due the lack of train data.

Reducing the number of train data point (and so increasing the number of test data points) does increase performance in some cases. However, the

Figure 6: The expected means together with the variance generated by the Gaussian process, with $\sigma^2 = 10^{-7}$ and 14 train data points. The red line represents the means generated with the hyperparameters which optimize the log-likelihood $\hat{\theta} = 0.2\bar{\theta}$, $\hat{l} = 93\bar{l}$. The blue line is parameterized with $\hat{\theta} = 0.7\bar{\theta}, \hat{l} = \bar{l}$, which minimize the expected mean square error. The green line is generated with $\hat{\theta} = 0.1\bar{\theta}, \hat{l} = 98\bar{l}$. The variance is on most places not visible, as it is really close to the mean.

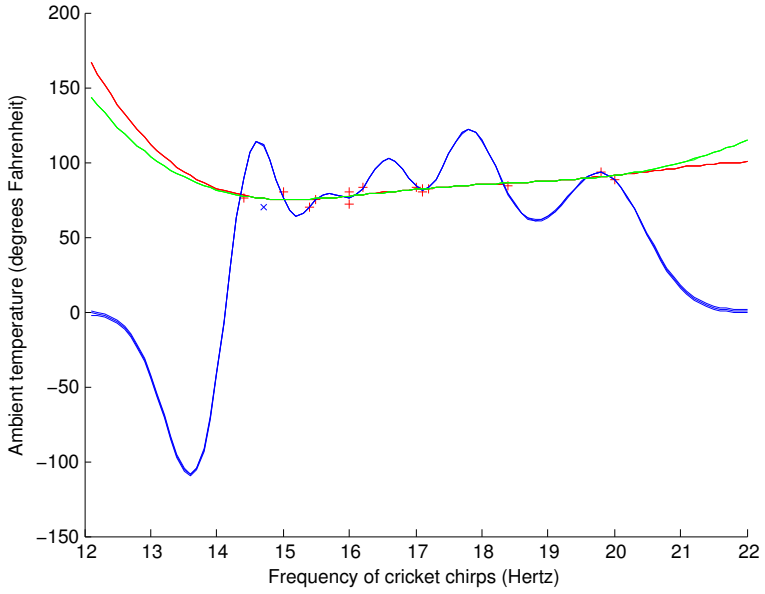


Figure 7: The error of the Gaussian process has been plotted against hyperparameters $\boldsymbol{\theta} = \begin{bmatrix} \theta \\ l \end{bmatrix}$. The process has a noise level of $\sigma^2 = 1^{-7}$. The error is measured as the mean square error among the test samples. Lower, blue values for the mean square error are more desirable. **Left:** The error plotted for $0.1\bar{\boldsymbol{\theta}} \leqslant \boldsymbol{\theta} \leqslant \bar{\boldsymbol{\theta}}$. **Right:** The error plotted for $\bar{\boldsymbol{\theta}} < \boldsymbol{\theta} \leqslant 100\bar{\boldsymbol{\theta}}$.

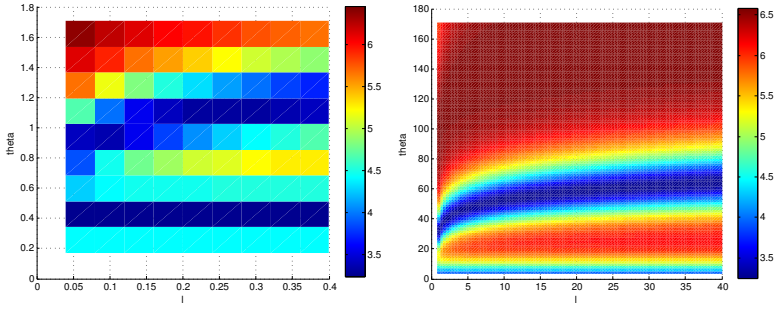


Table 2: The performance of Gaussian processes based on differently selected hyperparameters $\hat{\theta}$ and $\hat{l}$ with noise level $\sigma^2 = 10^{-4}$.

σ^2	Test items	Criterion	Cr. value	$\hat{\theta}$	$\hat{l}$	Performance
10^{-4}	1	Log-likelihood	−177937.199801	$0.4\bar{\theta}$	$98\bar{l}$	−75.745288
		Expected error	3.043957	$0.8\bar{\theta}$	$2\bar{l}$	−84.724637
		Variance	1.050777	$0.1\bar{\theta}$	$98\bar{l}$	−75.628623
	2	Log-likelihood	−175634.144192	$0.3\bar{\theta}$	$98\bar{l}$	−78.665675
		Expected error	7.861724	$0.1\bar{\theta}$	$\bar{l}$	−84.308779
		Variance	0.000088	$0.1\bar{\theta}$	$98\bar{l}$	−78.574643
	3	Log-likelihood	−4215.151426	$0.4\bar{\theta}$	$98\bar{l}$	−78.953028
		Expected error	8.514720	$0.9\bar{\theta}$	$0.1\bar{l}$	−82.342356
		Variance	25.881656	$0.1\bar{\theta}$	$98\bar{l}$	−78.965517
	4	Log-likelihood	−4123.566594	$0.3\bar{\theta}$	$98\bar{l}$	−77.410747
		Expected error	0.621619	$2\bar{\theta}$	$85\bar{l}$	−77.931735
		Variance	11.443740	$0.1\bar{\theta}$	$98\bar{l}$	−77.452683
	5	Log-likelihood	−4037.452951	$0.3\bar{\theta}$	$98\bar{l}$	−77.516761
		Expected error	1.525246	$0.6\bar{\theta}$	$1.2\bar{l}$	−69.502743
		Variance	13.491900	$0.1\bar{\theta}$	$98\bar{l}$	−77.912734

performance difference for the hyperparameters chosen to maximize the log-likelihood or variance is not significantly changed after reducing the train data set. In contrast, the hyperparameters optimized for a low expected error do improve from a small set of train data. This might be because this might be overfitting.

As can be seen in Figure 7, changing the hyperparameters can be of a big influence on the performance of the Gaussian process. This is not surprising, as these parameters are usually learnt by a process such as gradient ascent (Bishop, sec. 6.4.3, “Learning the hyperparameters”).