CUED - Engineering Tripos Part IIB 2019-2020 **Module Coursework**

Module	4F13	Title of report	Gaussian Processes

Type text here Date submitted: 06/11/2020	Assessment for this module is [✓ 100%] / ☐ 25% coursework of which this assignment forms 33.33 %

UNDERGRADUATE STUDENTS ONLY		POST GRADUATE STUDENTS ONLY			
Candidate number:	5673D	Name:		College:	

Feedback to the student

☐ **See also comments in the text**

		Very good	**Good**	Needs improvmt
C O N T E N T	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
P R E S E N T A T I O N	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Overall assessment (circle grade)		A*	**A**	B	C	D
Guideline standard		>75%	**65-75%**	55-65%	40-55%	<40%
Penalty for lateness:		*20% of marks per week or part week that the work is late.*				

Marker: Date:

4F13: Probabilistic Machine Learning
Gaussian Processes

Candidate number: 5673D
Words count: approximetly 1150

November 6, 2020

a GP with a squared exponential SE covariance function

We want to fit the 1-D data loaded from `cw1a.mat` using a 0-mean Gaussian Process with Squared Exponential (SE) covariance function. Assuming our observations have WGN with variance σ_n^2, we can compute the covariance between targets as:

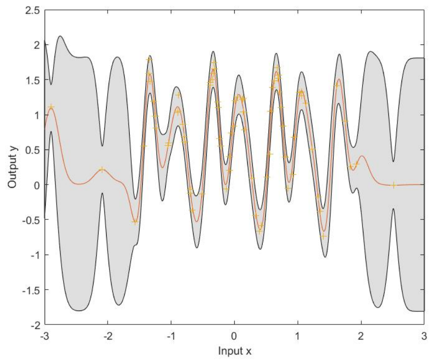
$$k(x_i, x_j) = \sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{2l^2}\right) + \delta_{ij}\sigma_n^2 \tag{1}$$

where $\sigma^2 > 0$ is the signal variance, $l > 0$ is the length scale and δ_{ij} is Kronecker delta. The hyper-parameters of this model are $\theta = (l, \sigma, \sigma_n)$. The length scale ($l$) determines the smoothness of the function. A low-valued l means that our data is weakly correlated (equivalent of a low signal to noise ratio) and our model will have troubles extrapolating further away from out data points and will tend to fluctuate more quickly. A high-valued l means a strong correlation between the points, therefore better approximations for point further away from our data.

The optimal values for our hyper-parameters are calculated by minimizing the negative marginal log-likelihood $-\log(p(y|x, \theta))$ wrt. the hyper-parameters θ using the code below.

```
S = load('cw1a.mat')
hyp = struct('mean', [], 'cov', [0 0], 'lik', -1);
covfunc = @covSEiso; %set the Covariance fuction as SE
meanfunc = [];
likfunc = @likGauss;
hyp.cov = [-1, 0]; hyp.lik = 1; %initialise hyperparameters
%minimise negative log marginal likelyhood
hyp2 = minimize(hyp, @gp, -100, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
%set targets
xs = linspace(-3, 3, 100)';
%train GP
[mu s2] = gp(hyp2, @infGaussLik, meanfunc, covfunc, likfunc, x, y, xs);
```

We get the following minimum $log(p(y|x, \theta) = -11.9$, corresponding to $\theta = (0.128, 0.897, 0.118))$. The predictions are shown in Figure 1, on the left, in orange and the 95% error bars in grey. The small length scale effect can be clearly seen. The error bars are tight around the data and are larger in regions with few data points. In the regions with many data points the lower bound of the error bars is $2\sigma_n$, while in the other regions increases to 2σ. This makes the model good at interpolation and while not being efficient for extrapolation. Moreover this optimization prevents the model from over-reducing the values for length scale and noise variance, thus avoiding overfitting.



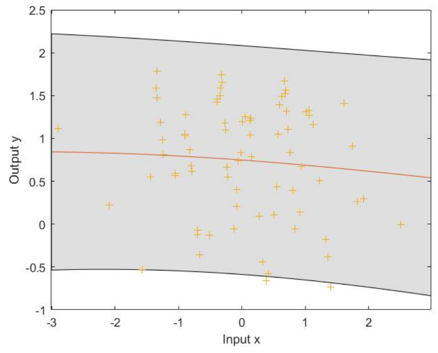


Figure 1: GP trained with SE covariance function with different local minima.
On the left, $\theta = (l, \sigma, \sigma_n) = (0.128, 0.897, 0.118)$ and $-\log(p(y|x, \theta)) = 11.9$.
On the right, $\theta = (l, \sigma, \sigma_n) = (8.042, 0.696, 0.663)$ and $-\log(p(y|x, \theta)) = 78.2$)

b Local minima

Our optimization function minimize a differentiable multivariate function using conjugate gradients. This method find the nearest local minimum to the initialization point. Therefore, by changing the initial hyper-parameters we expect to achieve other minima. For instance, by initializing (hyp.cov = [1, 0]) gives us another minimum for the marginal likelihood $\log(p(y|x, \theta)) = -78.2$. This corresponds to $\theta = (8.042, 0.696, 0.663)$. From this, we expect our predictions to have error bars similar size.

From the right side of Figure 1, we can see the our data is misinterpreted as noise. A large length scale implies an equivalent lower signal-to-noise ratio.

Looking at the value of the marginal likelihoods, we can clearly see that the first model has a better value without even taking the exponential values. From this reasons, we can safely say that our first model is indeed the better one.

c GP with a periodic covariance function

In this section, we take a look at the same process implemented using a periodic covariance function. This covariance has the following form

$$k(x_i, x_j) = \sigma^2 \exp\left(-\frac{2\sin^2\left(\frac{\pi(x_i - x_j)}{p}\right)}{l^2}\right) + \delta_{ij}\sigma_n^2 \tag{2}$$

where , σ, l, σ_n have the same meanings as in section a. The new hyper-parameter is the period of the functions p.

We used the following initialization:

```
covfunc = @covPeriodic; hyp.cov = [0, 0, 0]; hyp.lik = 1; %initialise hyperparameters
```

and after following the same steps as before we get $\log p(y|x, \theta) = 35.2$ and $\theta = (l, p, \sigma, \sigma_n) =$
$= (0.936, 0.999, 1.008, 0.110)$. We expect the error bars to be narrow as the signal-to-noise ratio is high($\sigma >> \sigma_n$). The lower bound for the error bars is $2\sigma_n$. Moreover, both the length scale and the period are approximately 1. Therefore the signal should not change very much during one period and the functions should have a smoothly behaviour similar to sinusoidal. The periodicity of the model makes confident extrapolation making use of the highly dense data region repeatedly over the entire space.

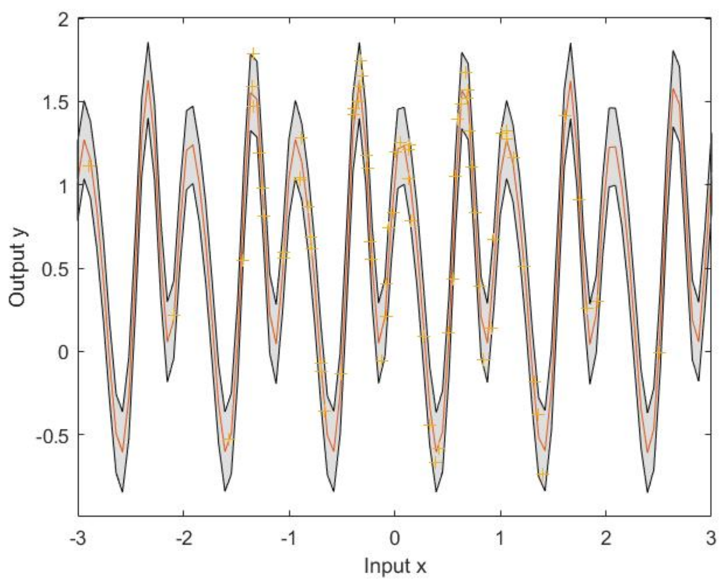


Figure 2: Heat map plot of the the approximation of the model evidence obtained in the grid search.

This periodic model not only fits the high number of data points in the central region, but also the isolated points. And on a first look we would choose this model although it is much restrictive. Indeed, if we have a look at the marginal likelihood has a better value than our first model.

d Data generation

For this task we will be using a combination of the SE covariation and periodic covariance.

$$k(x_i, x_j) = \sigma^2 \exp\left(-\frac{(x_i - x_j)^2}{2l_{SE}^2}\right) \exp\left(-\frac{2\sin^2\left(\frac{\pi(x_i-x_j)}{p}\right)}{l_{per}^2}\right) \tag{3}$$

By using the product of these two covariances we want to achieve a behaviour that is periodical in shape but its the shape is scaled by a mean component.

The hyper-parameters have the same meaning as in previous sections, but this time we have different length scales for periodicity and squared exponential. While l_{per} looks at the smoothness inside one period, l_{SE} acts on the smoothness of the mean modulator. As before, low l_{per} corresponds to high oscillation inside one period and a high value corresponds to a smoother transition between the prediction. The value of l_{SE} also acts as an indicator for the weighting of the models. A high value l_{SE} result in the exponential tending to 1. Thus, our covariance tends to be only periodical. Otherwise, for low-valued l_{SE}, our model will have high fluctuations outside the period and the SE covariance over-weights the periodic covariance.

This implementation can be done using a new covariance matrix:

```
K = feval( covfunc, hyp2.cov , xs);% find covariance matrix K
K = K + 1e-6*eye(200); ;% make K positive definite ( stability )
y = chol(K)'* randn( 200 , 1);% generate y from x
```

The Cholesky decomposition works only on positive definite matrices. Therefore, we make sure that our matrix has positive eigenvalues by adding a diagonal matrix. Otherwise, due to finite precision of the computer the eigenvalues would be so small that they would be approximated to zero and the Cholesky method would fail.

e 2-D fitting

This time we will load the 2-D data from `cw1e.mat`. The only difference from previous sections is that our inputs for the `gp` function are 2-D. As we did before, we must first choose a covariance function for our model. This time with chose for our first model a Squared Exponential covariance function with Automatic Relevance Detemination (ARD) distance measure. This has the following form:

$$k(x_i, x_j) = \sigma^2 \exp\left(-\frac{(x_i^{(1)} - x_j^{(1)})^2}{2l_1^2} - \frac{(x_i^{(2)} - x_j^{(2)})^2}{2l_2^2}\right) \tag{4}$$

As we can see the only difference from our first model is the, now, we have to hyper-parameters for the length scale l_1 and l_2, one for each dimension. The model predictions are shown in Figure 3 and the corresponding hyper-parameters are $\theta = (l_1, l_2, \sigma, \sigma_n) = (1.5116, 1.2859, 1.1073, 0.1026)$ with a marginal likelihood of $\log p(y|x, \theta) = 19.2$. As before, the length scales determine the smoothness in each direction. The small difference in magnitude shows a similar structure on both x1-axis and x2-axis.

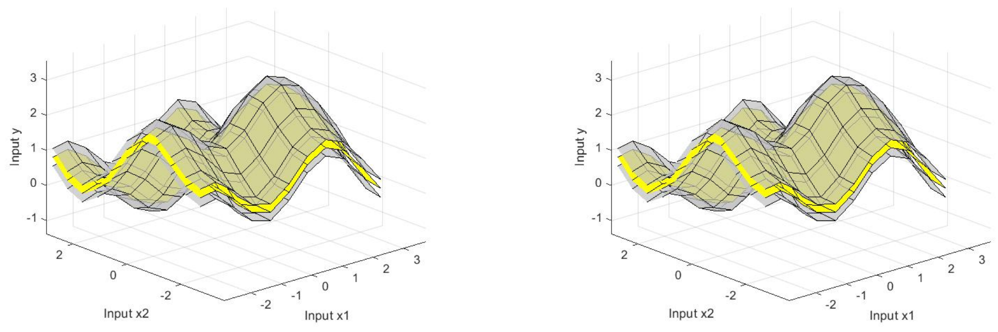


Figure 3: GP trained in 2-D with ARD. On the left, SEard. On the right, Sum of covariances

Our second model wants to implement the sum of two ARD covariance functions as follows:

$$k(x_i, x_j) = \sigma_1^2 \exp\left(-\frac{(x_i^{(1)} - x_j^{(1)})^2}{2l_{11}^2} - \frac{(x_i^{(2)} - x_j^{(2)})^2}{2l_{12}^2}\right) + \sigma_2^2 \exp\left(-\frac{(x_i^{(1)} - x_j^{(1)})^2}{2l_{21}^2} - \frac{(x_i^{(2)} - x_j^{(2)})^2}{2l_{22}^2}\right) \tag{5}$$

This model offers more flexibility as it account for different variances and dependencies on each input-axis. Moreover, its worst case scenario seems to be when one of the σs collapses to zero and it ends up in the previous model. This flexible implementation guarantees a better marginal likelihood as long as we break the symmetry. If symmetry exits, our optimizer will see one σ and one l for each axis. Thus, this model should be almost always better, although it takes a bit more time to implement.