CUED - Engineering Tripos Part IIB 2023-2024 **Module Coursework**

Module	4F13	Title of report	Coursework 1 - Gaussian Processes

Date submitted: 05/11/2023	Assessment for this module is [✓ 100%] / ☐ 25% coursework of which this assignment forms 33.3 %

UNDERGRADUATE and POST GRADUATE STUDENTS

Candidate number:	5553C	[✓ Undergraduate] ☐ Post graduate

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?			

Marker: Date:

1 Exercise A

```
meanfunc = []; hyp.mean = []; likfunc = @likGauss; hyp.lik = 0;
covfunc = @covSEiso; hyp.cov = [-1 0]; % Squared Exponential Covariance Function
hyp2 = minimize(hyp, @gp, -100, @infGaussLik, meanfunc, covfunc, likfunc, x, y)
[mu s2] = gp(hyp2, @infGaussLik, meanfunc, covfunc, likfunc, x, y, xs);
```

Figure 1: Code for a GP with the @covSEiso covariance function.

After loading the training data from cw1a.mat, that we will fit our model to, the code in Figure 1 was used to train a Gaussian Process with a squared exponential covariance function, shown in Equation 1. In this model we see the hyper-parameters are: the characteristic length scale, l , the standard deviation of the signal, σ_f, and lastly to the standard deviation of the output noise, σ_y, as shown in full in Equation 2.

$$k_{se}(x, x') = \sigma_f^2 \exp\left(\frac{-(x-x')^2}{2l^2}\right) \tag{1}$$

$$cov[y, y'] = k_{se}(x, x') + \sigma_y^2 \delta_{xx'} \tag{2}$$

We observe from Table 1 that the minimisation of the hyper-parameters has considerable impact in increasing the performance, as indicated by the marginal likelihood.

	l	σ_f	σ_y	ML
Initialised	0.368	1.00	1.00	4.46e-41
Minimised	0.128	0.897	0.118	6.80e-6

Table 1: Shows the initialised and minimised hyper-parameters, as well as resulting marginal likelihood of the GP with a squared exponential covariance function.

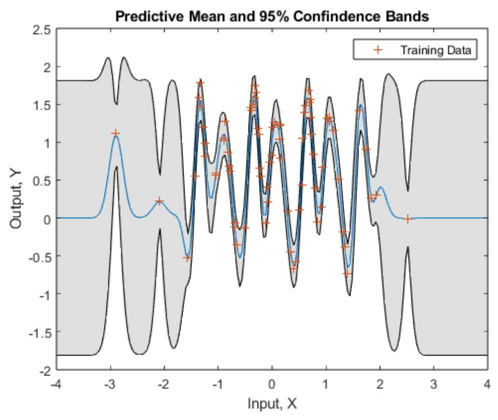

Figure 2: Predictive mean and 95% confidence bands from a Gaussian process with a squared exponential function trained on data from cw1a.mat. The minimised hyper-parameters are shown in Table 1.

From Figure 2 we observe that predictions by our model have smaller confidence bands in areas with a higher density of training data: when x^*, the test input, is located near more training inputs, x, our model has a greater confidence in its predicted output. Furthermore when x^* has little training data nearby, we see the uncertainty increases, but not indefinitely, so the confidence bands have a maximum size. These observations can be explained using Equation 3 [1], which allows us calculate the uncertainty for our predicted outputs.

$$\sigma_{y^*}^2 = \sigma_f^2 + \sigma_y^2 - k(x^*, \mathbf{x})^T (k(\mathbf{x}, \mathbf{x}) + \sigma_y^2 \mathbf{I})^{-1} k(x, \mathbf{x}^*) \tag{3}$$

We note that both σ_f and σ_y are independent from x and x^*, therefore when the third-term, which is dependent on x and x^*, goes to zero we are left with the maximum uncertainty that our model will output with predictions. This occurs when x^* is not correlated to $\mathbf{x}$, hence $k(x, \mathbf{x}^*) = 0$. So the maximum size of the 95% confidence bands is $4\sigma_{y^*} = 4\sqrt{\sigma_y^2 + \sigma_f^2} = 3.62$. This is confirmed in Figure 2.

When x^* is closer to the training data, we know that $k(x, \mathbf{x}^*)$ increases in value, hence reducing σ_{y^*}; from Equation 1 we can see this variation is due to $\sigma_f = 0.897$ and $l = 0.128$. The former dictates the vertical aspect of the covariance functions. The second scales the distance between two points, $x - x'$, and hence determines how quickly the function should vary depending on the amount of data in the area. Characteristic length scales that are too large lead to smoother functions that under-fit the data and those too small leads to a function that over-fits and has high uncertainty found in regions with sufficient data.

2 Exercise B

```
for i = 1:N
    for j = 1:N
        hyp = struct('mean', [], 'cov', [Xs(i) 0], 'lik', Ys(j));
        nlml = gp(hyp, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
        Zs(j, i) = log(nlml);
    end
end
```

Figure 3: Code for local optimum searches (data presented in Figure 4).

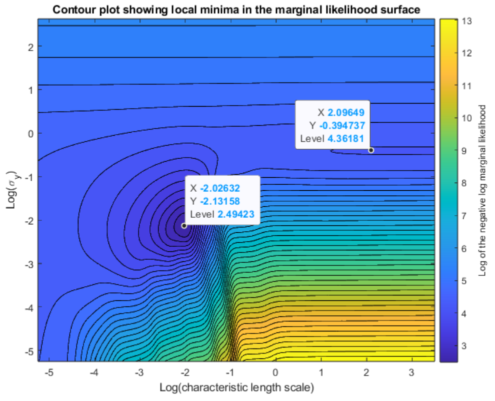

Figure 4: Illustration of two local optimuma found for varying σ_y and l with $\sigma_f = 1$. When σ_f can vary: the minima on the left is identical to minimised values seen in Table 1 and the minima on the right is identical to minimised values seen in Table 2.

For simple models, influenced largely by few hyper-parameters, we are able to try many different hyper-parameter combination in order to identify local optimums and the most viable solution; We found several local optimums using the squared exponential covariance function, as show in Figure 4. To expand this to a more complex model, one would have to sample points within suitable space, as opposed to testing a vast uniformly arranged set of discrete points as we have done.

	l	σ_f	σ_y	ML
Initialised	1.00	1.00	1.00	3.01e-40
Minimised	8.34	0.699	0.663	1.07e-34

Table 2: Shows the initialised and minimised hyper-parameters, as well as the resulting marginal likelihood, of another local optimum of the GP with a squared exponential covariance function.

As seen in Table 2, the second local optimum has a substantially larger l relative to the other optimum from Table 1 and a high amount of noise, σ_y. From Figure 4 we observe that in this region the marginal likelihood is near constant with l, hence the optimum is located where the noise itself , σ_y, can sufficiently explain the data, due to l being far too large to be capture a sizeable amount of information about the data. Comparing the marginal likelihoods of both minima (6.80e-6 from Table 1 and 1.07e-34 from Table 2), it is clear that the hyper-parameters from Table 1 are far better at describing the data; this is confirmed in Figure 5.

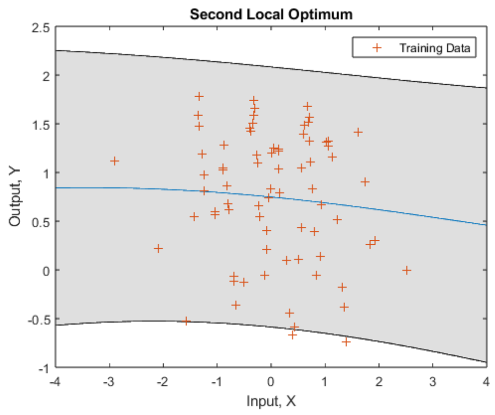

Figure 5: Predictive mean and 95% confidence bands from a Gaussian Process with a squared exponential function trained on data from cw1a.mat. The minimised hyper-parameters are shown in Table 2.

3 Exercise C

```
meanfunc = []; hyp.mean = []; covfunc = @covPeriodic; hyp.cov = [0 0 0];
likfunc = @likGauss; hyp.lik = 0;       % Gaussian likelihood
hyp2 = minimize(hyp, @gp, -100, @infGaussLik, meanfunc, covfunc, likfunc, x, y)
[mu s2] = gp(hyp2, @infGaussLik, meanfunc, covfunc, likfunc, x, y, xs);
```

Figure 6: Code for a GP with a periodic covariance function, @covPeriodic.

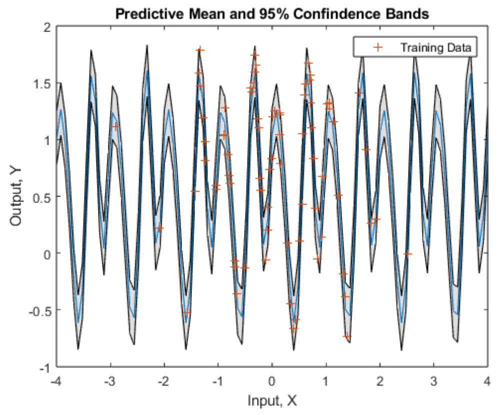

Figure 7: Predictive mean and 95% confidence bands from a GP with a periodic covariance function trained on data from cw1a.mat. The minimised hyper-parameters are shown in Table 3.

Using the same data as Section 1, we train a new GP with a periodic covariance function given in Equation 4.

$$k_{periodic}(x, x') = \sigma_f^2 \exp\left(\frac{-2sin^2(\pi|x - x'|/p)}{l^2}\right) \tag{4}$$

Using the initial values shown in Table 3, we observe from Figure 7, that following hyper-parameter minimisation and training, the periodic function is able to accurately represent all the data with a small amount of uncertainty. Unlike Figure 2 the confidence bands do not increase to a larger size as x^* moves further from the training data. This is because when using the periodic covariance function, distance between points is measured within the u space, given by $u = (sin(x), cos(x))^T$. Hence the greatest between two points is limited to $p/2$.

	l	p	σ_f	σ_y	ML
Initialised	1.00	1.00	1.00	1.00	2.84e-35
Minimised	1.04	0.999	1.23	0.109	2.03e15

Table 3: Shows the initialised and minimised hyper-parameters, as well as resulting marginal likelihood of the GP with a periodic covariance function.

Both the continuously small confidence bands and the substantial increase in the marginal likelihood , compared to Section 1 (see Table 1) indicate that it is extremely likely the data was generated from a purely periodic covariance function with added noise as shown in Equation 5.

$$cov_{periodic}[y, y'] = k_{periodic}(x, x') + \sigma_y^2 \delta_{xx'} \tag{5}$$

4 Exercise D

```
covfunc={@covProd,{@covPeriodic,@covSEiso}};
hyp.cov = [-0.5 0 0 2 0];
x = linspace(-5,5,400)';
K = feval(covfunc{:}, hyp.cov, x);
y = chol(K+1e-6*eye(400))'*randn(400,2);
```

Figure 8: Random noise free function generation from a GP.

Shown in the code in Figure 8, we are required to add a diagonal matrix with small constant, $1e{-}6$. this ensures that the matrix is positive-definite which is requirement for Cholesky decomposition to be performed.

$$k_{prod}(x, x') = k_{se}(x, x') \times k_{periodic}(x, x') \tag{6}$$

From Figure 9 we observe the result of multiplying the squared exponential covariance function (@covSEiso) with the periodic covariance function (@covPeriodic) on the functions that the GP will generate; the functions generated are periodic but are now warped vertically; before the peaks and troughs remained constant, now they are able to vary. The way this varies is determined by squared exponential covariance function, as shown by Equation 6.

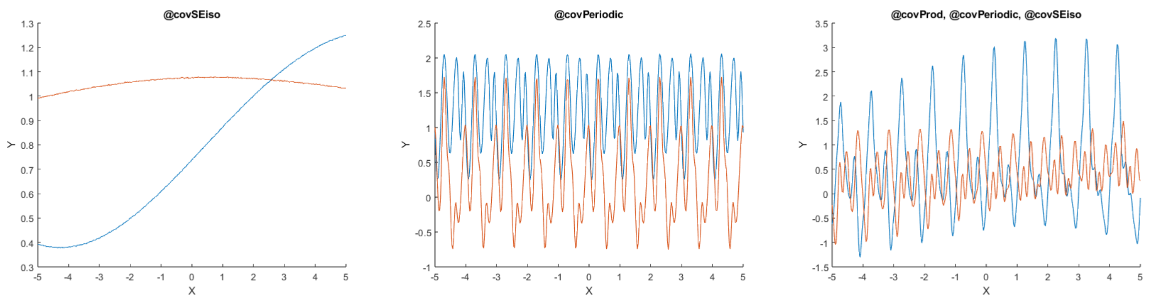

Figure 9: Plots of noise free sample functions generate from GPs with different covariance functions. (Left) @covSEiso. (Middle) @covPeriodic. (Right) Product of @covSEiso and @covPeriodic.

5 Exercise E

```
% Model 1 --------------------------------------------------------------------
covfunc_1 = @covSEard; hyp_1.cov = [0 0 0]; likfunc_1 = @likGauss; hyp_1.lik = 0;
% Model 2 --------------------------------------------------------------------
covfunc_2 =  {@covSum, {@covSEard, @covSEard}}; hyp_2.cov = 0.1*randn(6,1);
likfunc_2 = @likGauss; hyp_2.lik = 0;
```

Figure 10: Code for GPs utilising the @covSEard covariance functions with 2-D inputs.

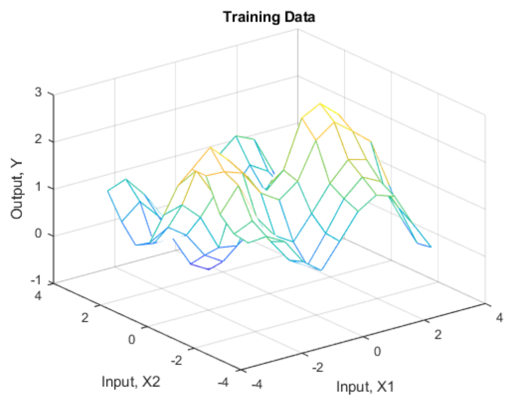

Figure 11: Visualisation of the training data from cw1e.mat.

Table 4 Model 1 shows the minimised hyper-parameters of a GP trained on data seen in Figure 11, with a covariance function given by Equation 7.

$$k_{ard}(x,x') = \sigma_f^2 \exp\left(-\sum_{d=1}^{D=2} \frac{(x_d - x'_d)^2}{2\nu_d^2}\right) \quad (7)$$

Notice in Equation 7, the distance between two 2-D points is determined by the sum of the distances in each of the dimensions, scaled by two different characteristic length scales. From Table 4 Model 1 we see that ν_1 and ν_2 have a ratio close to one and hence will contribute roughly the same amount; this is clearly a set back, as two separate points despite being close in one-dimension, will have a large characteristic distance if the other dimension has a large distance., which will reduce the value of $k(x,x')$ to zero. This can be overcome by summing two ARD covariance functions.

	ν_1	ν_2	σ_f	ν'_1	ν'_2	σ'_f	σ_y	ML
Model 1	1.51	1.29	1.11	NA	NA	NA	0.103	2.22e8
Model 2	395.	0.987	0.704	1.45	736	1.12	0.0978	6.54e28

Table 4: Shows the minimised hyper-parameters and marginal likelihoods. Model 1 has a covariance function @covSEard. Model 2 has a covariance function {@covSum, {@covSEard, @covSEard}}.

$$k_{sum-ard}(x,x') = k_{ard}(x,x') + k_{ard}(x,x')' \quad (8)$$

Using Equation 8 it is now possible to have a $k(x,x')$ that doesn't rapidly descend to zero if just one-dimension has a large characteristic length, as the covariance function is now the summation of two different ARD covariance functions. As seen in Figure 10, we achieve this by initialising parameters of both ARD functions to different values, so they don't both optimise to the same point. From Table 4, we see this added model complexity achieves greater performance than the single ARD function as shown by the significantly higher marginal likelihood (comparing ML in 4). This is due to each of the two ARD functions focusing on a distinct dimensions, therefore being able to capture the variation in both dimensions simultaneously as opposed to being dominated by a single dimension. $\nu_1 \gg \nu_2$ indicates the first ARD function focuses on the first dimensions and $\nu_2' \gg \nu_1'$ indicates the second ARD function focuses on the second dimension.

Referring to Figure 12 we can visualise this performance increase. Comparing the top left plot to the bottom left plot, we see that the more complex model is better able to produce more complex surface shapes in 2-D and hence better represent the data. Furthermore the plots on the right show that the confidence bounds follow this same behaviour and are therefore smaller for a greater area.

Lastly, this increase of model complexity does not increase the likelihood of the model overfitting. From [2] we have the automatic Occam's razor principle, the marginal likelihood is optimised to favour a simple model that fits the data well. This is Shown in Equation 9 [2], where the second term penalises complexity. Note $\mathbf{K}_y = \mathbf{K}_f + \sigma_y \mathbf{I}$. In practice we see this happen with our more complex model, which is reduced to a more simple form for most characteristics lengths, due to the large amount of scaling from ν_1 and ν_2'.

$$log(\mathbf{p}|\mathbf{X}, \theta) == -\frac{1}{2}\mathbf{y}^T\mathbf{K}_y^{-1}\mathbf{y} - \frac{1}{2}log(\mathbf{K}_y) - \frac{n}{2}log(2\pi) \tag{9}$$

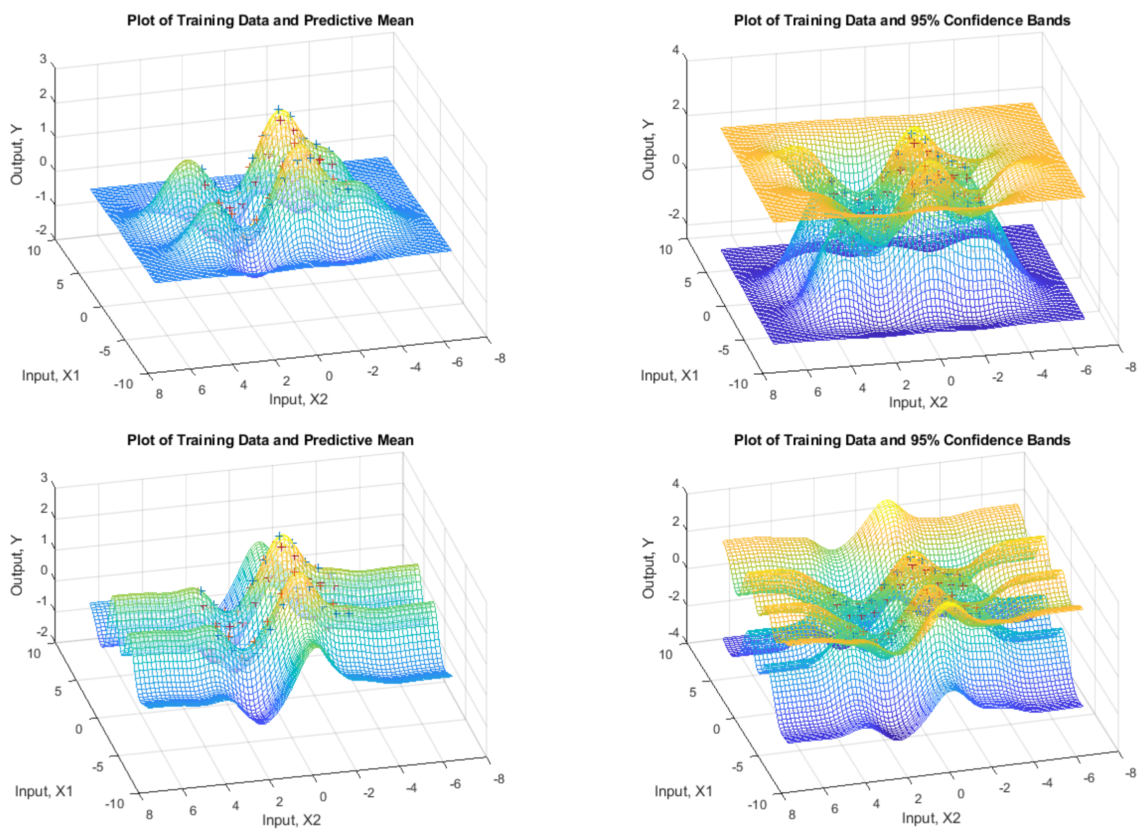

Figure 12: (Top) Plots of the GP predictions with an ARD covariance functions. (Bottom) Plots of the GP predictions with a covariance function given by Equation 8.

6 References

[1] Murphy, K.P., 2012. Machine learning: a probabilistic perspective. MIT press.

[2] C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006