CUED - Engineering Tripos Part IIB 2021-2022 **Module Coursework**

Module	4F13	Title of report	Coursework 1: Gaussian Processes
Date submitted:	05/11/2021	Assessment for this module is ☒ 100% / ☐ 25% coursework of which this assignment forms 33.33 %	

UNDERGRADUATE STUDENTS ONLY		POST GRADUATE STUDENTS ONLY			
Candidate number:	5588C	Name:		College:	

Feedback to the student

☐ **See also comments in the text**

		Very good	**Good**	Needs improvmt
CONTENT	**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:			
PRESENTATION	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Indicative grades are not provided for the FINAL piece of coursework in a module

Assessment (circle one or two grades)		A*	A	B	C	D
Indicative grade guideline		>75%	65-75%	55-65%	40-55%	<40%
Penalty for lateness:		*20% of maximum achievable marks per week or part week that the work is late.*				

Marker: Date:

4F13 Probabilistic Machine Learning
Coursework 1: Gaussian Processes

February 8, 2022

1 Exercise (a)

Using the isometric squared exponential kernel in Equation (1), Figure 1 was generated using the optimised hyperparameter values in Table 1, and it shows narrow error bars where data points are concentrated, and larger variance where there is less data, with uniform variance in far away regions. The small noise standard deviation, σ_n, does not allow for significant noise to be present in the training data, and the small covariance amplitude, σ_f, means the amplitude of the covariance function, with respect to the mean, is small. Thus, the change in variance across inputs corresponds to the small length-scale, l, which makes Equation (1) decay quickly for dissimilar or far-away data-points, so regions with few data points have low covariance function values.

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{(x - x')^2}{2l^2}\right) + \sigma_n^2 \delta_{xx'} \ [1] \tag{1}$$

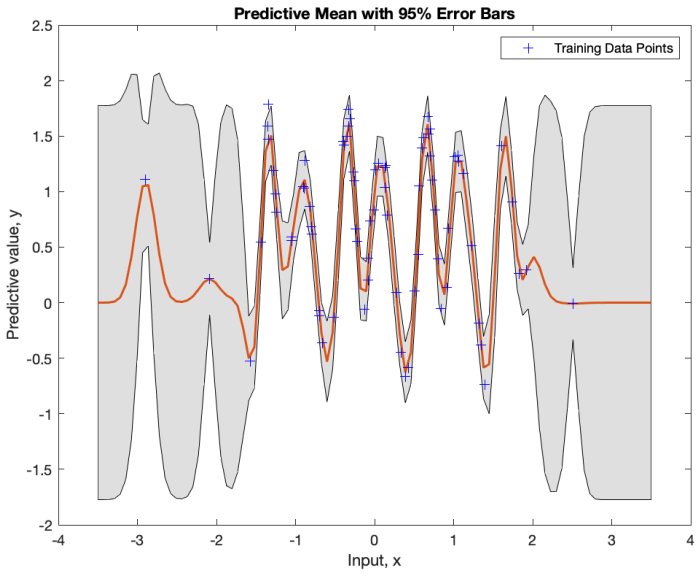

Figure 1: 95% predictive error bars of zero mean function Gaussian Process (GP) model with an isometric squared exponential covariance function.

The predictive variance in Equation (2) shows that a data-fit term is subtracted from $k(x_*, x_*) + \sigma_n^2$, which is constant $\forall\ x_*$ once the hyperparameters have been tuned. In between consecutive data points, the small

Optimised NLML	**Covariance HP** $\{\ln l,\ \ln \sigma_f\}$	**Likelihood HP:** $\{\ln \sigma_n\}$
11.899	[-2.054, -0.1087]	-2.1385

Table 1: Optimised Negative Log Marginal Likelihood (NLML) and corresponding natural logarithm hyperparameter values.

length scale causes most entries in the covariance vector $\mathbf{k}(x_*, \mathbf{x})$ to be small, so there is a larger variance than in regions where data points are more clustered and the data-fit term is larger. Similarly, for inputs very different to training, the covariance function is essentially negligible, so Equation (2) tends to the constant $k(x_*, x_*) + \sigma_n^2$.

$$\mathrm{Cov}(y_*|x_*, \mathbf{x}, \mathbf{y}) = k(x_*, x_*) + \sigma_n^2 - \mathbf{k}(x_*, \mathbf{x})^T \left[\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I}\right]^{-1} \mathbf{k}(x_*, \mathbf{x}) \tag{2}$$

```
meanf = [];
covf = @covSEiso; % Equation 1 in report
likf = @likGauss;
% Hyperparameter initialisation to cov_l = e^{-1}, cov_ampl = e^{0}
hyp = struct('mean', [], 'cov', [-1 0], 'lik', 0); % No mean function hyperparameters

% Hyperparameter optimisation via negative log marginal likelihood
[opt_hyp, nlml] = minimize(hyp, @gp, -100, @infGaussLik, meanf, covf, likf, data.x, data
.y); % Minimise the negative log marginal likelihood.
% Making predictions
[pred_mean, pred_std] = gp(opt_hyp, @infGaussLik, meanf, covf, likf, data.x, data.y,
test_x);
% Predictive plot
f = [pred_mean+1.96*sqrt(pred_std); flip(pred_mean-1.96*sqrt(pred_std),1)];


fill([test_x; flip(test_x,1)], f, [7 7 7]/8); % Shade in error bars
hold on;
plot(test_x, pred_mean, 'LineWidth', 1.5);
g = plot(data.x, data.y, 'b+', 'DisplayName', 'Training Data Points');
legend(g);
```

Listing 1: Code for HP Initialization and Plotting

2 Exercise (b)

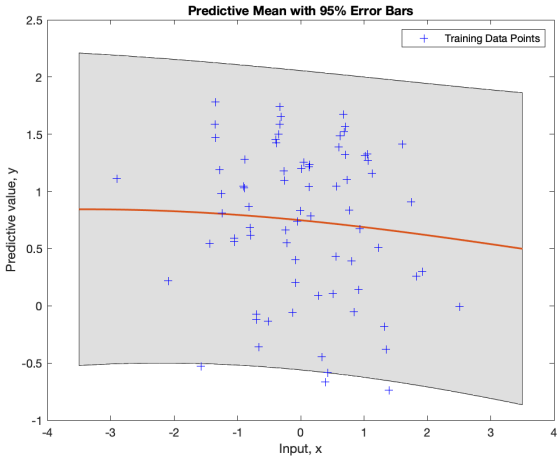

Figure 2: Predictive Distribution for GP Model with squared exponential covariance function, initialised with different hyperparameter values.

Figure 2 shows the predictive mean, explicitly stated in Equation (3), is too slow to track changes in the input, because it is dependent on the covariance function, and hence the larger length-scale makes it unable to change quickly enough to capture the data. The model's error-bars, which are proportional to the predictive variance in Equation (2), also remain large throughout the domain, and their slope is smooth, as the variation of the data fit term with respect to the input is slow due to the slow decay of the covariance function.

$$\mathbb{E}\left[y_*|x_*, \mathbf{x}, \mathbf{y}\right] = \mathbf{k}(x_*, \mathbf{x})^T \left[\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I}\right]^{-1} \mathbf{y} \tag{3}$$

Optimised NLML	**Covariance HP** $\{\ln l,\ \ln \sigma_f\}$	**Likelihood HP:** $\ln \sigma_n$
78.2202	[2.0847, -0.3625]	-0.4109

Table 2: Optimised NLML and corresponding natural logarithm hyperparameters for different initialisation values. The white crosses represent the two local minima.

The slow decay of the variance can be explained by the length-scale in Table 2, which is 62.7 times larger than the previous model, reducing the magnitude of the term inside the exponential in Equation (1) for a given pair (x, x'). Equation (4) further reinforces this idea, as a Taylor expansion on $k(x_*, x)$ shows that the rate of change of the covariance function with respect to the input x_* is inversely proportional to the square of the length scale.

$$\frac{\partial}{\partial x_*}\left[k(x_*, x)\right] = -\frac{1}{l^2}(x_* - x)k(x_*, x) \approx -\frac{\sigma_f^2}{l^2}(x_* - x) + \mathcal{O}((x_* - x)^2) \tag{4}$$

To investigate the existence of multiple local optima, the hyperparameters were initialised 500 times, and the resulting color-coded contour plot of the NLML is shown in Figure 3. Clearly, only two local optima exist for the given dataset: one corresponds to Exercise 1, a low-noise model able to track changes to data closely, and the other corresponds to Exercise 2, a higher noise model with a smoother mean and covariance

function, unable to accurately track the data. Given equal model complexities, the higher local optimum for the NLML shown in Table 2 suggests this model is a worse fit to the data than Exercise 1.

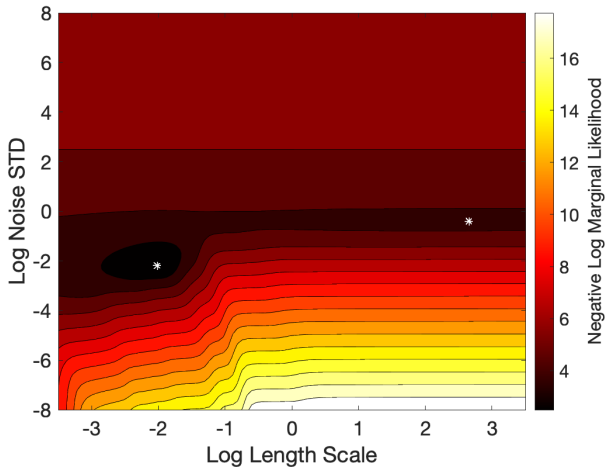

Figure 3: Contour Plot for NLML obtained when uniformly varying the length scale and likelihood noise hyperparameters.

```
N = 100;
a = 3.5;
Xs = linspace(-a,a,N);
Ys = linspace(-2*a,2*a,N);
Z = zeros(N,N);

for i = 1:N
    for j = 1:N
        hyp = struct('mean', [], 'cov', [Xs(i) 0], 'lik', Ys(j));
        [nlZ, dnlZ] = gp(hyp, @infGaussLik, meanf, covf, likf, x, y);
        Z(j,i) = log(nlZ); % Smoothen values
    end
end
contourf(Xs,Ys,Z,15)
colormap(hot)
cb = colorbar; % create and label the colorbar
cb.Label.String = 'Negative Log Marginal Likelihood';
```

Listing 2: Code for Uniform Grid HP Optimization via Contour Plot

3 Exercise (c)

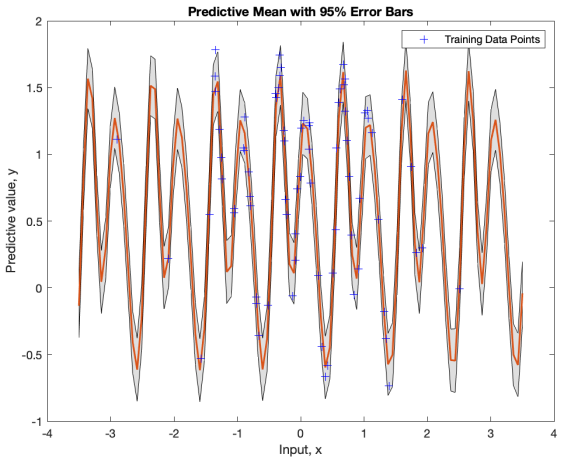

Figure 4: Predictive Distribution of Periodic Covariance Function in Equation (5) with 95% Error Bars.

Figure 4 shows a similar predictive distribution to Exercise 1 in the regions near the data, which is impressive since the length-scale in Exercise 1 is 0.128, compared to $l = 0.974$, which suggests the presence of periodicity, p, shortens the effective length-scale.

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{2}{l^2}\sin^2\pi\frac{(x - x')}{p}\right) \tag{5}$$

```
    meanf = [];
    likf = @likGauss;
    covf = @covPeriodic;

    % RANDOM initialiation for hyperparameter tuning
    N =500;
    nlmls = zeros(N, 1);
    hyps = struct('mean',[], 'cov',[], 'lik', []);
    for t = 1:N
        hyp = struct('mean',[], 'cov', 2*randn(3,1), 'lik', 2*randn(1));
        [opt_hyp, nlml] = minimize(hyp, @gp, -100, @infGaussLik, meanf, covf, likf, data.x,
    data.y);
        nlmls(t) = nlml(end);
        hyps(t) = opt_hyp;
    end

    [opt_nlml, opt_s] = min(nlmls);
    opt_hyp = hyps(opt_s);
    % Making predictions
    [pred_mean, pred_std] = gp(opt_hyp, @infGaussLik, meanf, covf, likf, data.x, data.y,
    test_x);
    % Predictive plot
    f = [pred_mean+1.96*sqrt(pred_std); flip(pred_mean-1.96*sqrt(pred_std),1)];
```

Listing 3: Code for random HP optimization of periodic covariance model

To verify the model's validity, a Kolmogorov-Smirnov test is run to check if the residuals are normally distributed [2], as the model assumes Gaussian observation noise. At a 95% confidence level, with a p-value

of $0.8568 > 0.05$, the null hypothesis in (3) cannot be rejected [4].

$$\mathcal{H}_{\mathcal{O}}: \frac{\mathbf{y}_{\text{training}} - \mathbf{y}_{\text{predictive}}}{\sigma_{\text{residuals}}} \sim \mathbb{N}(0,1)$$

Optimised NLML	**Covariance HP :** $\{\ln l, \ln p, \ln \sigma_f\}$	**Likelihood HP:** $\{\ln \sigma_n\}$
-35.2666	[0.0337, -0.0012, 0.1587]	-2.2087

Table 3: Optimised NLML and corresponding natural logarithm hyperparameters for periodic covariance function.

Yet, Figure 5 shows a poor fit of the standard Gaussian PDF to the histogram, suggesting the number of data points, 75, is too small to confidently declare the residual distribution. The model also seems overly confident in regions far away from training, as the periodicity of the mean and the narrowness of the error bars is still present, suggesting our model fails at extrapolation.

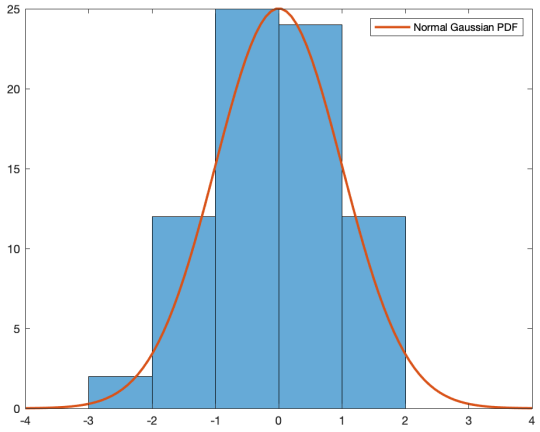

Figure 5: Residual Histogram in blue overlaid with Standard Gaussian PDF in orange.

```
[pred_mean, pred_std] = gp(opt_hyp, @infGaussLik, meanf, covf, likf, data.x, data.y,
data.x);

rs =(data.y - pred_mean);
rs_std = std(rs);
r = (data.y - pred_mean)/rs_std;
[h, p] = kstest(r)

h = histogram(r, 'Normalization', 'countdensity');
hold on;
xvals = linspace(-4,4,100);
pdf = normpdf(xvals);
pdf = pdf * (max(h.Values)/max(pdf));
g =plot(xvals, pdf, 'LineWidth', 2, 'DisplayName', 'Normal Gaussian PDF');
legend(g);
```

Listing 4: Code for Kolmogorov Smirnov test and histogram

4 Exercise (d)

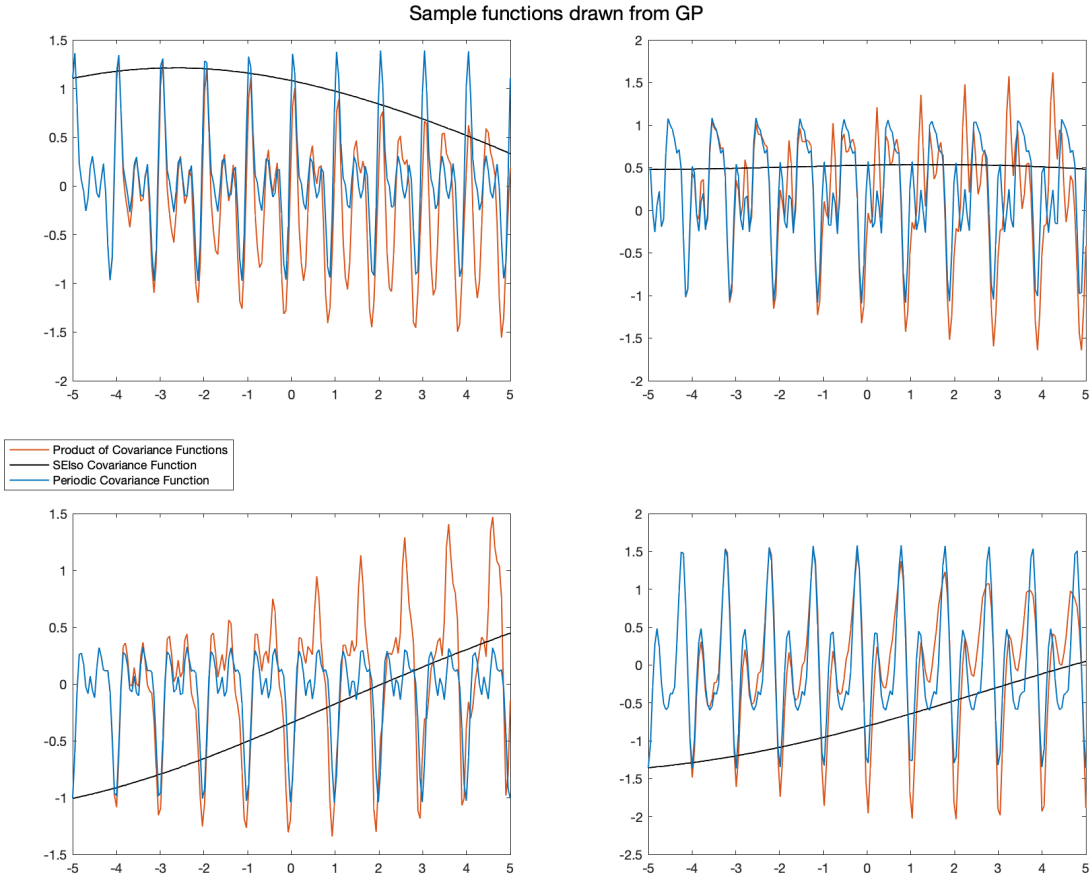

Figure 6: Random draws from four different Gaussian Process models with zero mean function, and a squared exponential, periodic and covariance function given by Equation (6), in black, blue, and orange, respectively.

In order to generate the random functions in Figure 6, a small perturbation matrix was added to the covariance matrix to ensure it remained positive definite, which is violated when finite precision arithmetic causes the squared difference between far away inputs to be very small, leading to numerical underflow when it is exponentiated in Equation (6).

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{(x-x')^2}{2l_1^2} - \frac{2}{l_2^2}\sin^2\frac{\pi\,(x-x')}{p}\right) \tag{6}$$

From Figure 6, it can be deduced that the period hyperparameter corresponds to one integer unit along the horizontal axis, as there are 10 function peaks in a range with 10 integers. The small periodic length scale, l_1 explains the high rate of change within a single period. The larger l_2 represents the decay-time of

periodicity due to the *@covSEiso* function [3], as evidenced by Figure 6, which shows the product function slowly changing shape with respect to the periodic function.

```
% Need to generate covariance matrix
K_prod = feval(covf{:}, [-0.5, 0, 0, 2, 0], test_x);
K_iso = feval(@covSEiso, [2,0], test_x);
K_periodic = feval(@covPeriodic, [-0.5, 0, 0], test_x);
for i=1:4
    z = randn(N,1);
    y_prod = chol(K_prod + 1e-6*eye(N))'*z;
    y_iso = chol(K_iso + 1e-6*eye(N))'*z;
    y_periodic = chol(K_periodic + 1e-6*eye(N))'*z;
    subplot(2,2,i)
    plot(test_x, y_prod, 'Color',[0.8500 0.3250 0.0980], 'DisplayName',"Product of
Covariance Functions", 'LineWidth', 1);
    hold on;
    plot(test_x, y_iso, 'Color', 'k', 'DisplayName',"SEIso Covariance Function", '
LineWidth', 1 );
    plot(test_x, y_periodic, 'Color',[0 0.4470 0.7410] , 'DisplayName',"Periodic
Covariance Function", 'LineWidth', 1) ;
end
```

Listing 5: Code for Cholesky Decomposition

5 Exercise (e)

```
% Plot 3D Plot
a =8;
T = ((2*a)/0.1 + 1);
[t1 t2] = meshgrid(-a:0.1:a, -a:0.1:a);
test_x = [t1(:), t2(:)];
% Prediction code in between %
h1=scatter3(data.x(:,1),data.x(:,2), data.y, 'r', 'filled');
hold on;
h2=mesh(t1,t2,reshape(pred_mean,T,T));
legend([h1,h2], ["Training data points", "Predictive Plot"]);
xlabel('Input First Dim')
ylabel('Input Second Dim', 'Rotation',-0)
zlabel('Predictive Output')
% Data-Fit error
rs =(ys - pred_mean);
rs_mean = mean(rs);
rs_std = std(rs);


% Plot 2D Slice
test_x = [linspace(-8, 8, 121)', linspace(-8, 8, 121)'];
% Prediction code in between %
fill([test_x; flip(test_x,1)], f, [7 7 7]/8);
hold on;
plot(test_x(:,1), pred_mean, 'r');
ylabel("Predictive value, y");
xlabel("First Dimension of Test Points");
subplot(2,1,2);
fill([test_x; flip(test_x,1)], f, [7 7 7]/8);
hold on;
plot(test_x(:,2), pred_mean,'b');
sgtitle("Predictive Mean with 95% Error Bars");
xlabel("Second Dimension of Test Points");
ylabel("Predictive value, y");
```

Listing 6: Code for Exercise (e) Plots and Data-Fit error

Figure 7 shows a non-linear, multi-modal surface plot of data from *cw1e.mat*, and a quick inspection of the data array shows the input values are 11 blocks of 11 uniformly distributed between $[-3, 3]$.

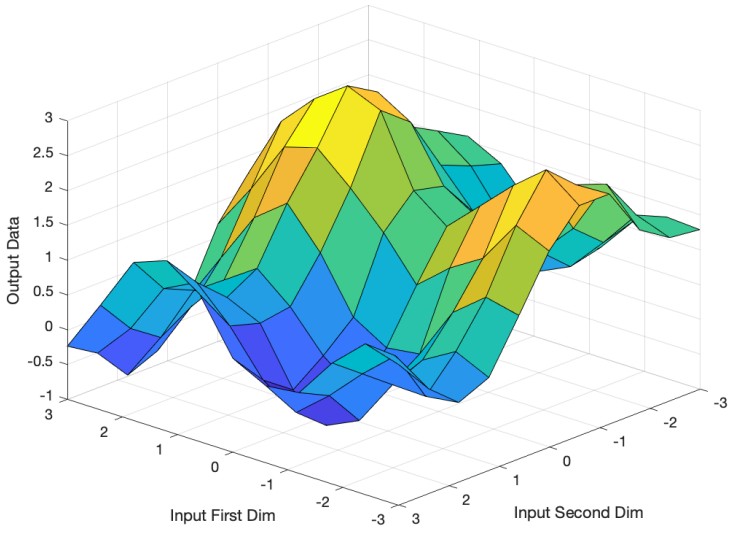

Figure 7: Data Surface Plot from *cw1e.mat*. Lighter colors represent higher output values.

Using the Automatic Relevance Function in Equation (7), Table 4 suggests the variation of the output with respect to the input in each dimension is similar, since $\frac{l_1}{l_2} = 1.18$, and the large length scales $\{l_1, l_2\} = \{1.54,\ 1.29\}$ explain why the left-hand plot in Figure 8 shows smooth predictive plots in both dimensions.

$$k_{ARD}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left\{-\frac{1}{2}\sum_{i=1}^{D}\frac{(x_i - x_i')^2}{l_i^2}\right\} \tag{7}$$

Optimised NLML	**Covariance HP :** $\{\ln l_1, \ln l_2, \ln \sigma_f\}$	**Likelihood HP:** $\{\ln \sigma_n\}$	**Residual STD**
-19.2187	[0.4131, 0.2515, 0.1019]	-2.2765	0.4827

Table 4: Optimised NLML and corresponding natural logarithm hyperparameters for ARD Covariance Model.

Figure 8 shows similar predictive plots for the single ARD and the sum of ARD function models in regions near the training data, but by splitting the input data into 80% training, 20% test, an estimate for the data-fit error, shown in Tables 4 and 5, suggests that the second model fits the data better. Table 5 also shows that the first covariance function in the covariance sum model captures the variation in the output with regard to the first dimension, as the length-scale ratio is $\frac{l_2}{l_1} = 89063$, and the second function captures the variation in the second dimension. Since the effective covariance is the sum of these two functions, the more complex model captures the same trend as the first model, but fitting the data better as the NLML ratio is $\frac{NLML_2}{NLML_1} = 3.3 \times 10^{20}$. The increased marginal likelihood and improved data-fit, as quantified by the residual standard deviation, suggest the summation model is better, but at the expense of a more complex model. Care is also needed to avoid overfitting as the NLML does not marginalise out the model hyperparameters,

and the right plot in Figure 8 shows non-zero predictions in far away regions from the data, indicating possible overfitting.

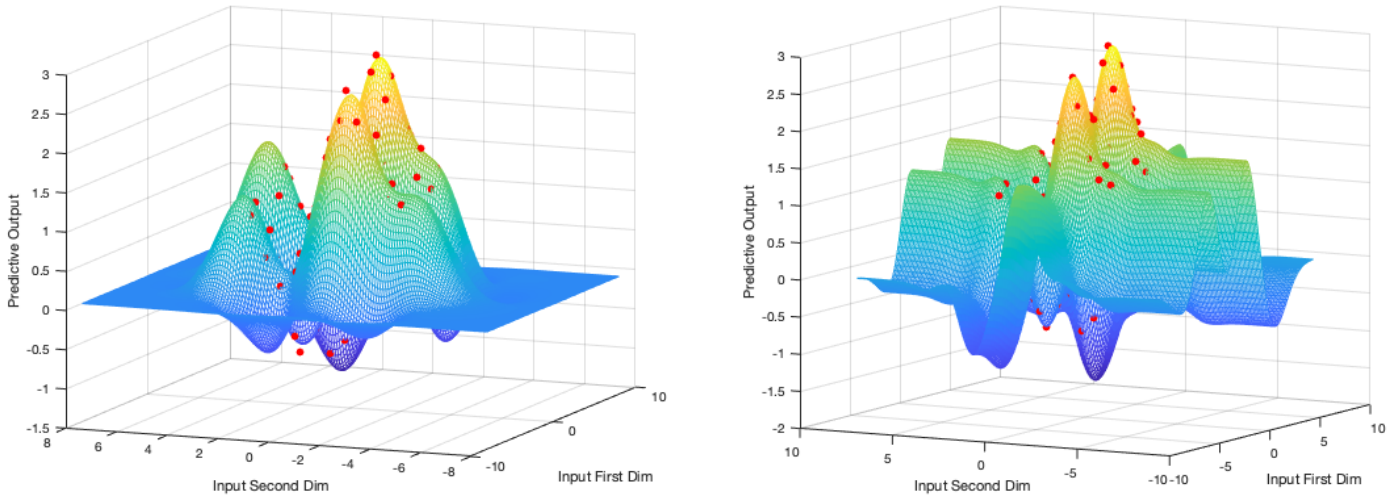

Figure 8: 3D Visualisation of Predictive Output. On the left, the single ARD covariance model; on the right, the sum of two ARD covariance functions. Training points in red.

Optimised NLML	**Covariance HP :** $\{\ln l_1, \ln l_2, \ln \sigma_{f1}, \ln l_3, \ln l_4, \ln \sigma_{f2}\}$	**Likelihood HP:** $\{\ln \sigma_n\}$	**Residual STD**
-66.3792	[0.3694, 11.7665, 0.1041, 9.8185, -0.0146, -0.3421]	-2.3246	0.3043

Table 5: Optimised NLML and corresponding natural logarithm hyperparameters for Sum ARD covariance model.

Words: 999

References

[1] Evelina Gabasova. *Ariadne Covariance Functions*. URL: `http://evelinag.com/Ariadne/covarianceFunctions.html`.

[2] NIST/SEMATECH. *Chapter 4.4 How can I tell if a model fits my data?* URL: `https://www.itl.nist.gov/div898/handbook/pmd/section4/pmd44.html`.

[3] C. E. Rasmussen and C. K. I. Williams. *Chapter 5, Model Selection and Adaptation of Hyperparameters*. URL: `http://www.gaussianprocess.org/gpml/chapters/RW5.pdf`.

[4] Wikipedia. *Kolmogorov Smirnov Test*. URL: `https://en.wikipedia.org/wiki/Kolmogorov-Smirnov_test`.