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

Module	4F13	Title of report	Coursework 1 - Gaussian Processes

Date submitted: Thursday 4 November 2021

Assessment for this module is ☒ 100% / ☐ 25% coursework
of which this assignment forms _____ %

UNDERGRADUATE STUDENTS ONLY		POST GRADUATE STUDENTS ONLY			
Candidate number:		Name:	Oussama Chaib	College:	Downing

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

(989 words excl. captions, headers, codes – TeXstudio)

Michaelmas 2021

Oussama Chaib, oc327@cam.ac.uk

November 2021

Question (a)

The data in `cw1a.mat` is loaded and used as a training set for a Gaussian model of squared exponential covariance function and Gaussian likelihood using the initial settings described below. The negative marginal log likelihood of the model is then minimized and the 95% predictive error bars are calculated from the obtained prediction data. The hyperparameters of the "optimal" fit (i.e: the model that minimises the negative marginal log likelihood) are described below, alongside the obtained function.

	l	ν	σ_{noise}	$likelihood$
Initial	0.3679	1	1	4.4636e-41
Predicted	0.1282	0.8970	0.1178	6.7971e-06

Table 1: Initial settings for hyperparameters

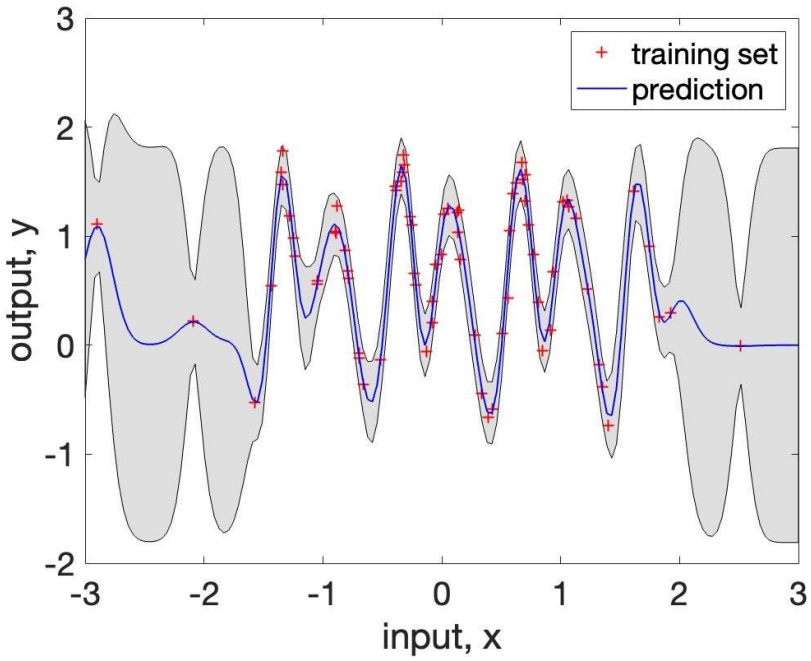

Figure 1: Training and predicted data fit using the described initial settings

We observe that the predictive errorbars are higher in regions with a low number of data points, and lower in regions containing more data points. This can be explained by the characteristic lengthscale which is in this case relatively small ($l = 0.1282$), yielding a "wiggly" fit. Therefore, when points are far away from each other (i.e: by a distance higher than l), the correlation between them drops and that is compensated by an increase in uncertainty. We also observe that the standard deviation of the function ν is 7 to 8 times higher than the standard deviation of the noise σ_{noise}. The larger error bars are therefore most likely due to the uncertainty of the underlying function, while the contribution of noise is, in this case, smaller.

```
% coursework1a.m
hyp = struct('mean', [] , 'cov', [-1,0] , 'lik', 0);
hyp_min = minimize(hyp, @gp, -100, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
[ys stds] = gp(hyp_min, @infGaussLik, meanfunc, covfunc, likfunc, x, y, xs);
```

Question (b)

The hyperparameters are initialised differently in this question. After trying different arbitrary values, we notice there are, *a priori*, two types of "optimal" models we can get by varying the lengthscale. The first one is the same as the model shown in the previous section (**Figure 1**). The second is represented in the graphs below.

	l	ν	σ_{noise}
Initial	$e^0 = 1$	1	1
Predicted	8.0421	0.6989	0.6632

	l	ν	σ_{noise}
Initial	$e^{10} = 2.2e04$	1	1
Predicted	2.2e04	0.7437	0.6671

Table 2: Hyperparameters obtained for local optima by modifying initial hyperparameters

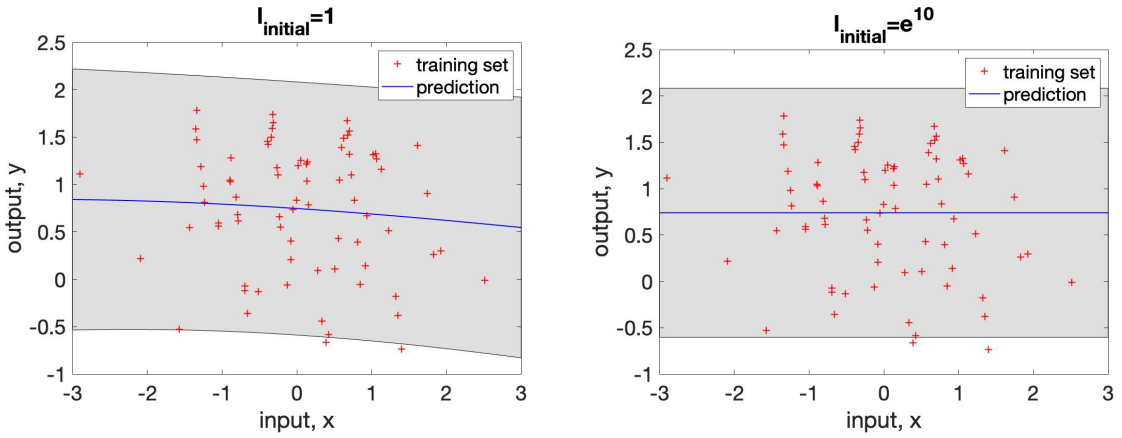

Figure 2: Functions obtained by modifying the initial hyperparameters (left to right: $log(l) = 0$ and $log(l) = 10$)

In both cases, the predicted length scale is significantly higher which results in a smooth function and a large noise standard deviation that is now comparable to the standard deviation of the model. The obtained model is too simple and can't capture any "structure" in the data which is why we have noise everywhere.
We vary both the lengthscale and the standard deviation of the noise to cover a larger range of

values (40 points each), while the standard deviation of the signal is kept constant ($\nu = 1$) as illustrated in **Figure 3**.

$$log(l) \in [-4, 10] \ ; \ log(\sigma_{noise}) \in [-5, 2]$$

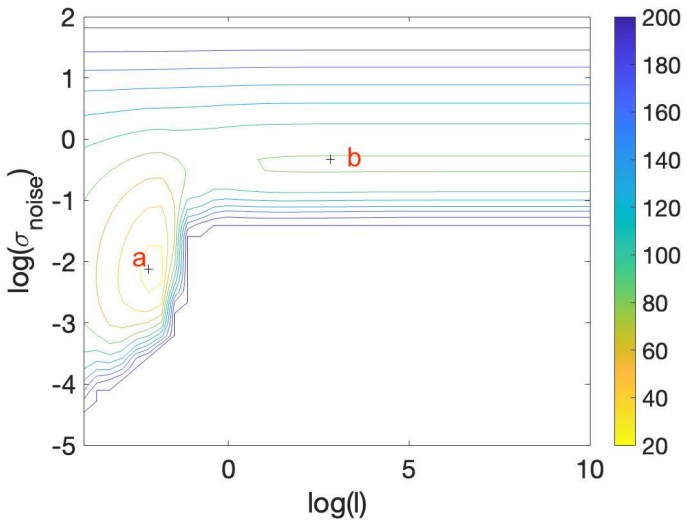

Figure 3: Negative log marginal likelihood (cf. colorbar) as a function of the lengthscale and noise standard deviation, both in log scale. Local minima are highlighted with '+' sign. (The values of $nlml \geq 200$ are set to 200 for an improved visibility.)

We notice the presence of two local minima in **Figure 3**. The first (point a) corresponds to a region with low noise and small lengthscale, while the second (point b) corresponds to a region with higher noise and larger lengthscale. Indeed, a quick calculation yields the following values:

Point	l	σ_{noise}	$likelihood$
'a'	0.1102	0.1191	5.098e-7
'b'	16.7855	0.7165	6.46e-35

Table 3: Hyperparameters associated with each point. The standard deviation of the signal is fixed here ($\nu = 1$).

By plotting the models associated with the hyperparameters of each point, we obtain the functions in **Figure 4**. This highlights the presence of a second optimum which corresponds to the hyperparameters of point b. When the lengthscale is large, the marginal likelihood becomes almost independent of the lengthscale but that is compensated by a larger noise term: the model explains the data as noise everywhere (point b). Conversely, when the lengthscale decreases, the marginal likelihood becomes less dependent on noise (point a) because the model is able to explain the data.
For all these reasons, and knowing it maximises the likelihood ($lik(a)/lik(b) \sim 10^{28}$), it is reasonable to suggest that the fit associated with point 'a' in **Figure 4** (or in the previous question, **Figure 1**) is better.

```
% coursework1b.m
for xx in range(linspace(-4,10,40))
   for yy in range(linspace(-5,2,40)):
```

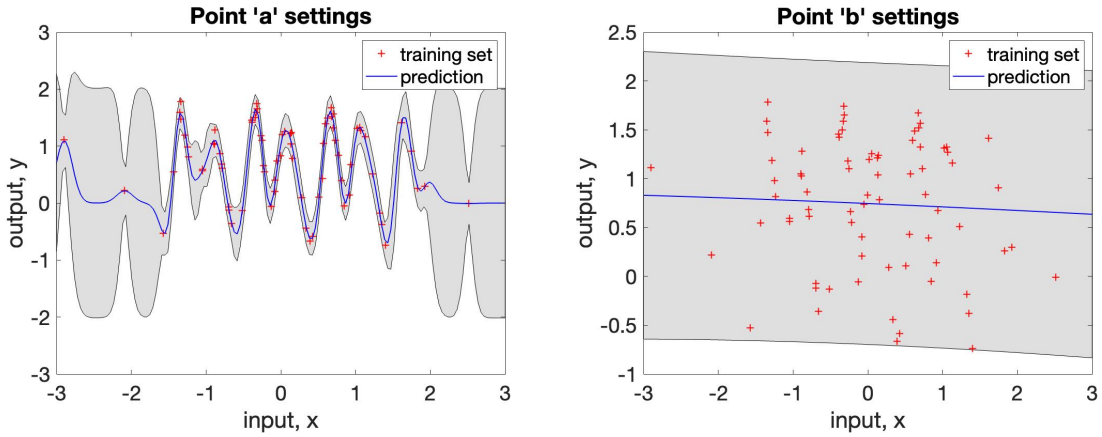

Figure 4: Functions obtained by using the hyperparameter settings for point 'a' and 'b' respectively

```
        hyp = struct('mean', [] , 'cov', [xx(i),0] , 'lik', yy(j))
        nlml=gp(hyp, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
        Z(j,i)=min(nlml,200)
    end
end
[X,Y] = meshgrid(xx,yy)
contour(X,Y,Z);
```

Question (c)

The data is `cw1a.mat` is now used to train a GP of periodic covariance function. We obtain the following results:

	l	p	σ	σ_{noise}	*likelihood*
Initial	1	1	1	1	2.8391e-35
covPeriodic predic.	1.0447	0.9988	0.9988	0.1094	2.0381e+15
covSEiso predic.	*0.1282*	-	*0.8970*	*0.1178*	6.7971e-06

Table 4: Initial and predicted hyperparameters

The obtained error bars are in this case relatively small compared to the ones obtained using a square-exponential covariance, even in low populated regions. This makes sense since using a periodic covariance function implies that:

$$k(x_1, x_2) = k(x_1, x_3) \;\; if \;\; |x_1 - x_2| \equiv |x_1 - x_3| \;\; [p]$$

where p is the period and supposing $x_1 \neq x_2 \neq x_3$.
The error bars are also almost constant everywhere because we have in this case $p \sim l$ which results in a model that is neither too simple (smooth function, higher noise) nor too complex (very wiggly, overfitting). The standard deviation of the noise is relatively smaller which suggests that the error bars are most likely due to the uncertainty of the model which is also smaller than the *covSEiso* one. Overall, the fit gives off a good impression and its likelihood is considerably higher than the one obtained using *covSEiso* which suggests that the data-generating mechanism is likely periodic.

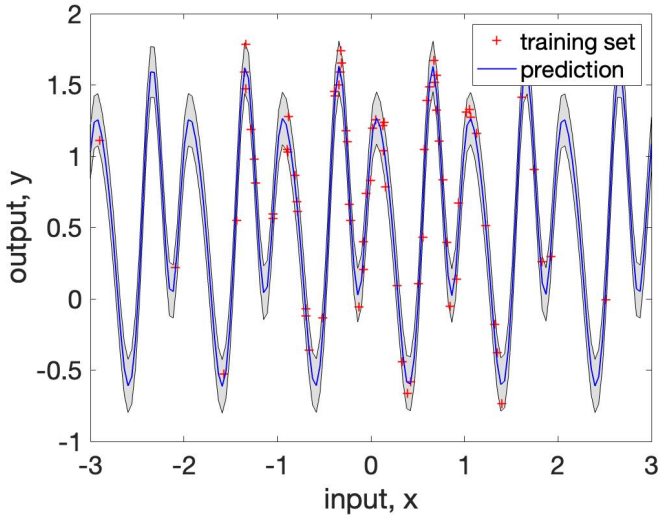

Figure 5: Training and predicted fit using a periodic covariance function (*note:*)

```
% coursework1c.m
covfunc = @covPeriodic;
hyp_min = minimize(hyp, @gp, -100, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
[ys stds] = gp(hyp_min, @infGaussLik, meanfunc, covfunc, likfunc, x, y, xs);
```

Question (d)

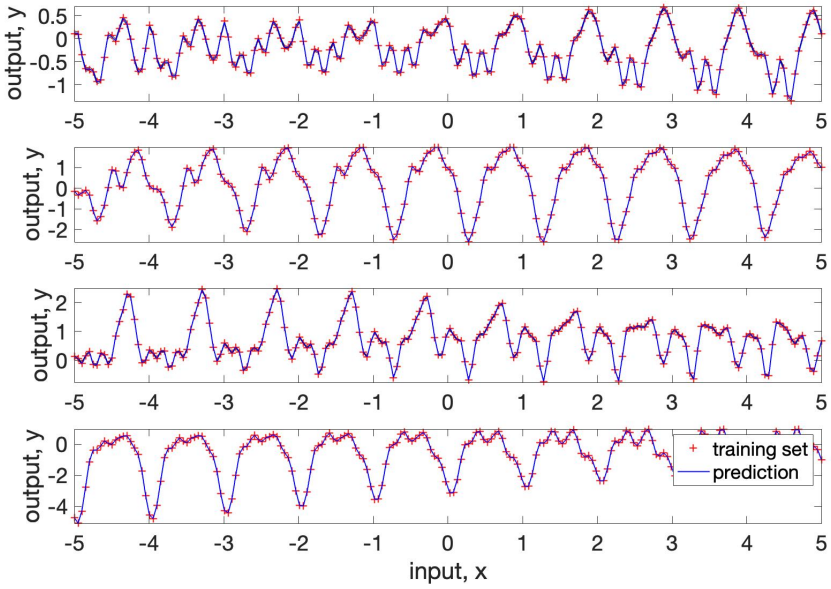

Figure 6: Functions obtained for different seeds (0.01, 0.1, 0.5, and 1 top-down) using random x data.

In order to apply Cholesky's decomposition to the covariance matrix, it should positive definite, hence the need to add the given diagonal matrix.
We observe that the functions in each case look like a combination of SEiso covariance (hump-like structure) and periodic covariance (repetitions over a period of 1). The composite kernel, therefore, combines the properties of both covariances. We notice we obtain a variety of functions when changing the seed but they all keep the properties of the two kernels. The period is approximately one which is consistent with the value calculated too. Because the lengthscale ($l \approx 0.7$) is 70% smaller than the period ($p \approx 1$), there is a small offset which "breaks" the periodic aspect of the function in certain humps.

```
% coursework1d.m
se=[.01 .1 .5 1]
for i=1:length(se)
   covfunc = {@covProd, {@covPeriodic, @covSEiso}};
   hyp.cov = [-0.5 0 0 2 0];
   x = linspace(-5,5,200)';
   K = feval(covfunc{:},hyp.cov, x);
   K_pos_def=K+1e-6*eye(200)
   y = chol(K_pos_def)'*gpml_randn(se(i), 200, 1);
   plot subplot
end
```

Question (e)

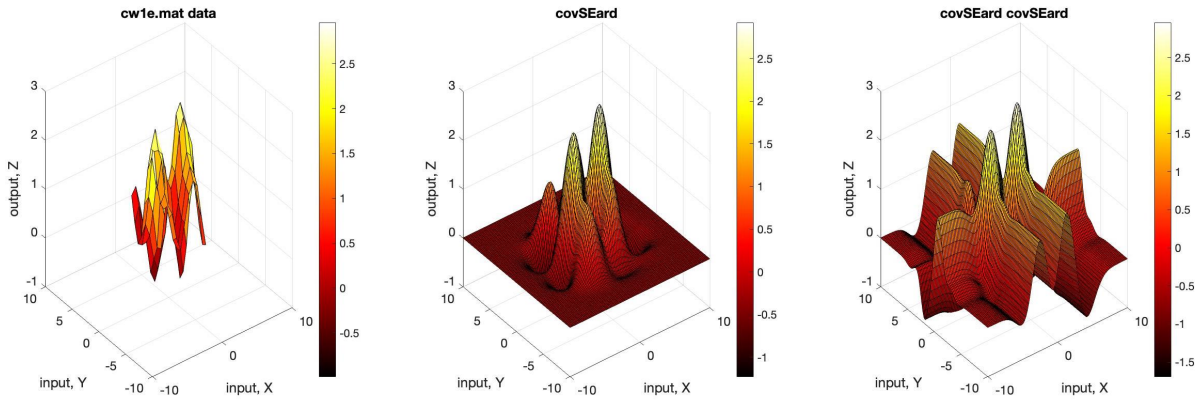

Figure 7: 3D plots of the two GP models using `covSE` and `covSum covSE covSE`

We train the two GP models to fit a 3D data set in the domain $[-10, 10]$ with 100 points. We first notice the graphs obtained are smoother than the original fit which can be explained by the higher point density of the models used ($100 * 100$ points instead of $11 * 11$). Overall, both models do a good job at fitting the data in the original region $[-3, 3]$. Outside this region, the second model (sum of `covSEard`) does a better job at extrapolating the data, while the first model "converges" to zero as x and y leave the original region. Looking at the marginal likelihoods, the second model has the higher marginal likelihood, which makes sense by looking at the graphs since it is able to fit data in spite of point scarsity. The first model is simple while the second is relatively complex.

	l_1	l_2	σ	σ_{noise}	*marginal likelihood*
`covSEard`	1.5116	1.2859	1.1073	0.1026	2.22e08
`covSum covSEard covSEard`	1.4496	2.29e03	1.1115	0.0978	6.9e28
	1.86e03	0.9845	0.7110	,,	,,

Table 5: Predicted hyperparameters

A quick calculation of hyperparameters shows that the first model has relatively small lengthscales which are approximately of the same order, while the second one is a sum of two `covSEard` covariances which both have one very large lengthscale in a direction. We recall that when the lengthscale is very large, the covariance becomes almost independent of that direction which essentially yields in our case one covariance along x and another over y. This is particularly useful as isolating both dimensions allows us to be more flexible. If `covSum` is thought of as an "or" operator, the covariance between two points would be high if they are either close in 'x' direction or in the 'y' direction.

```
% coursework1d.m
X=reshape(x(:,1),11,11);
Y=reshape(x(:,2),11,11);
Z=reshape(y,11,11);
surf subplot
covfunc = @covSEard;
hyp = struct('mean', [] , 'cov', [0.1,0.1,0.1] , 'lik', 0)
hyp_min = minimize(hyp, @gp, -200, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
surf subplot
covfunc = {@covSum, {@covSEard, @covSEard}};
hyp = struct('mean', [] , 'cov', 0.1*randn(6,1), 'lik', 0)
hyp_min2 = minimize(hyp, @gp, -200, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
surf subplot
```