Gwangbin Bae gb585@cam.ac.uk

4F13: Probabilistic Machine Learning
Coursework #1: Gaussian Process

1. Question A

Command 1 *Gaussian predictive with @covSEiso*

```
meanfunc = []; covfunc = @covSEiso; likfunc = @likGauss;
hyp = struct('mean', [], 'cov', [-1 0], 'lik', 0);
hyp2 = minimize(hyp, @gp, -200, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
[mu s2] = gp(hyp2, @infGaussLik, meanfunc, covfunc, likfunc, x, y, xs);
```

Table 1 *Hyperparameters and marginal likelihood*

	l	σ_f	σ_n	$p(\mathbf{y}\|\mathbf{X})$
Initial Values	0.3679	1	1	4.4636e-41
Optimized Values	0.1282	0.8970	0.1178	6.7972e-06

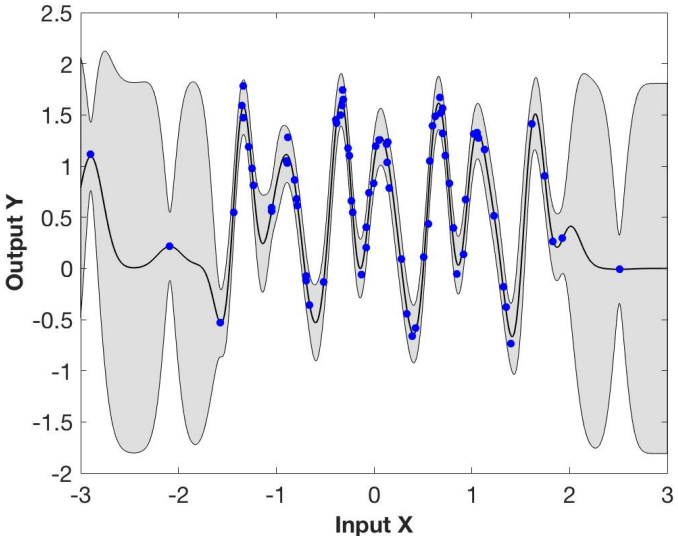

Figure 1 *Gaussian predictive with @covSEiso*

In **Figure 1,** the size of the error bar increases as the test input x_* moves away from the nearest training input x_{nt}. However, instead of increasing boundlessly, the uncertainty converges to a certain limit. Similarly, the error bar becomes smaller as x_* moves closer to x_{nt}, but is not zero even when $x_* = x_{nt}$. Such behaviour can be interpreted by looking at the mathematical form of the predictive variance:

$$\sigma_{y_*}^2 = \sigma_f^2 + \sigma_n^2 - \mathbf{k}(x_*, \mathbf{x})[\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I}]^{-1}\mathbf{k}(\mathbf{x}, x_*)$$

$\sigma_{y_*}^2$ consists of three terms. Since the first two terms are independent of x_*, the upper limit occurs when the third term becomes zero. This is when there is no correlation between x_* and $\mathbf{x}$ (i.e. $\mathbf{k}(x_*, \mathbf{x}) = \mathbf{0}$). Then, as x_* moves closer to the training input, the third term increases, thereby decreasing the predictive variance. However, the predictive variance cannot be zero because of the noise variance σ_n^2, present in both the second and third term.

The values of the optimized hyperparameters are in good agreement with the observation. Small characteristic length-scale ($l = 0.1282$) corresponds to the 'wiggly' behaviour of the function. The optimized values of σ_f and σ_n gives the maximum size of the error bar:

$$[95\%\ \text{Error Bar}]_{\max} = \left[4 \times \sigma_{y_*}\right]_{\max} = 4 \times \sqrt{\left(\sigma_f^2 + \sigma_n^2\right)} = 3.619$$

It can be confirmed from **Figure 1** that the size of the error bar never exceeds this limit. Lastly, non-zero noise variance suggests that the error bar would be present even when the training input is evaluated again.

2. Question B

Command 2 *Same as Command 1 except:*

```
hyp = struct('mean', [], 'cov', [0 0], 'lik', 0);
```

Table 2 *Hyperparameters and marginal likelihood*

	l	σ_f	σ_n	$p(\mathbf{y}\|\mathbf{X})$
Initial Values	1	1	1	3.0107e-40
Optimized Values	8.0430	0.6959	0.6631	1.0700e-34

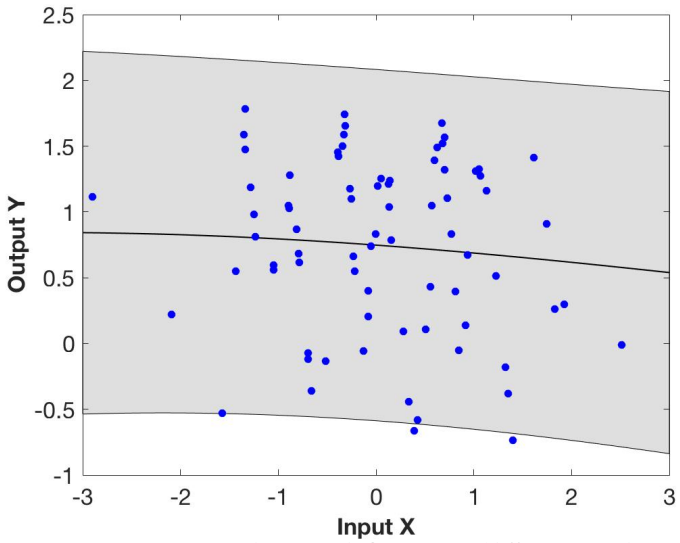

Figure 2 *Gaussian predictive with @covSEiso (different initialization)*

Figure 2 is generated by initializing the hyperparameters differently. The newly optimized length-scale ($l = 8.043$) is larger than the input domain. Hence, the predictive function shows monotonic behavior. In such regime, all the data is considered as noise.

In order to illustrate the existence of more than one local minima, a contour plot of the negative log marginal likelihood can be plotted by varying l and σ_n (for fixed $\sigma_f = 1$).

Command 3 *Contour plot of negative log marginal likelihood*

```
hyp = struct('mean', [], 'cov', [X(i) 0], 'lik', Y(j));
[nlZ, dnlZ] = gp(hyp, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
Z(j,i) = min(log(nlZ),5); % for better visualisation
Contour(X, Y, Z, 15)
```

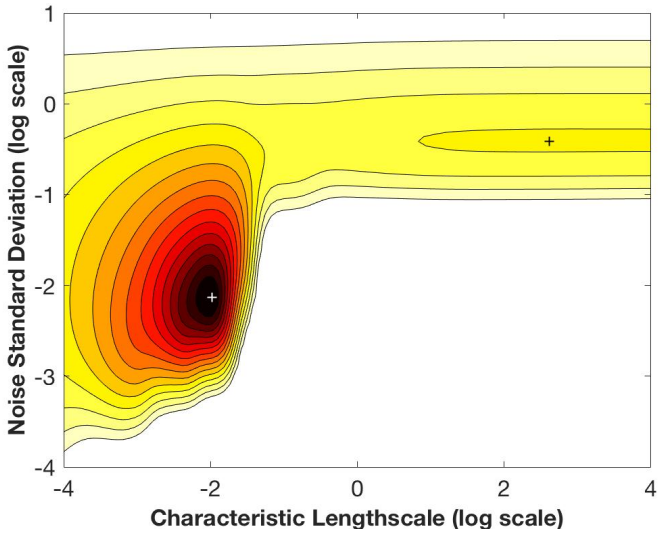

Figure 3 *Contour plot of negative log marginal likelihood (+: local minima)*

In **Figure 3**, the lower left minimum corresponds to the global optimum, where the two hyperparameters are optimized to fit the training data with high marginal likelihood. Another local minimum is observed in the region with high length-scale and high noise. In this regime, the likelihood becomes insensitive to the length-scale. Hence, the local minimum occurs when the noise is just enough to explain the training data. (If the noise is too big, the likelihood function will be flat, resulting in lower marginal likelihood.)

Since the marginal likelihood of the first fit (**Figure 1**) is 6.3525e+28 times higher than the second fit (**Figure 2**), it is reasonable to conclude that the first fit better explains the given training data.

3. Question C

Command 4 *Same as Command 1 except:*

```
covfunc = @covPeriodic; hyp = struct('mean', [], 'cov', [0 0 0], 'lik', 0);
```

Table 3 *Hyperparameters and marginal likelihood*

	l	p	σ_f	σ_n	$p(\mathbf{y}\|\mathbf{X})$
Initial Values	1	1	1	1	2.8391e-35
Optimized Values	1.0727	0.9989	1.2473	0.1095	2.0445e+15

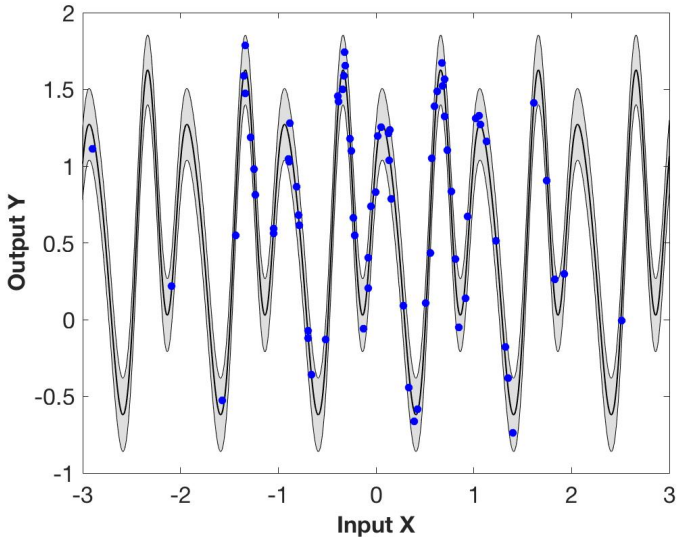

Figure 4 *Gaussian predictive with @covPeriodic*

The obtained Gaussian predictive shows great fit with the training data. Unlike the plot generated with @covSEiso, the size of the error bar is nearly constant across the input domain. This is because the effective distance between x_* and x is given as following:

$$d_{\text{effective}}(x_*, x) = \min(d, p - d) \quad \text{where} \quad d = |x_* - x| \bmod p$$

Thus, the effective distance cannot be longer than half the period($= 0.4995$). Since the characteristic length-scale($= 1.0727$) is larger than the maximum effective distance, the correlation to the training data is high for any given x_*. This explains why the error bar stays thin across the domain.

High marginal likelihood (see **Table 3**) suggests that it is likely that the data was generated from a periodic function. However, this is not enough. In order to justify the model, the normality of the residual must be tested:

$$H_0: y - f_{\text{periodic}}(x) = \epsilon(0, \sigma_\epsilon^2)$$

If the residual is Gaussian noise with no further dependence on x, it is reasonable to think that y consists only of periodic functions.

Command 5 *Normality test for residual*

```
[y_pred s2] = gp(hyp, @infGaussLik, meanfunc, covfunc, likfunc, x, y, x);
y_resid(i) = y(i) - y_pred(i);
y_resid(i) = y_resid(i)/y_resid_std; % normalization
[f,x_values] = ecdf(y_resid);
F = plot(x_values, f, 'b-'); G = plot(x_values, normcdf(x_values, 0, 1), 'r-');
[h, p] = kstest(y_resid) % Kolmogorov-Smirnov test
```

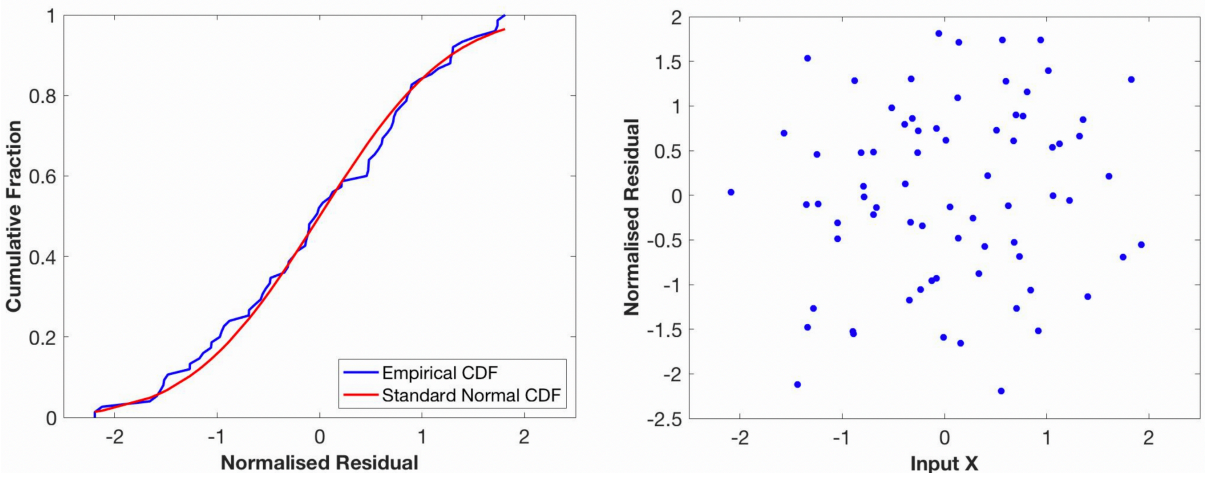

Figure 5 *Empirical CDF compared to standard normal CDF (left) & Residual vs. x (right)*

The first plot shows that the cumulative distribution of the normalized residual fits well with the standard normal CDF. The second plot shows that the residual has no visible dependence on x. Lastly, the p-value obtained from the Kolmogorov-Smirnov test is 0.5352. Hence, the null hypothesis (normality of the residual) cannot be rejected.

These evidences suggest that y, once its predictive component is removed, is Gaussian noise with no further dependence on x. Therefore, it is reasonable to think that the data was generated by adding Gaussian noise to a periodic function.

4. Question D

Command 6 *Generate data from a GP*

```
x = linspace(-5,5,200)';
covfunc = {@covProd, {@covPeriodic, @covSEiso}};
hyp.cov = [-0.5 0 0 2 0];
K =  feval(covfunc{:}, hyp.cov, x);
y = chol(K + 1e-6*eye(200))'* gpml_randn(seed, 200, 1);
```

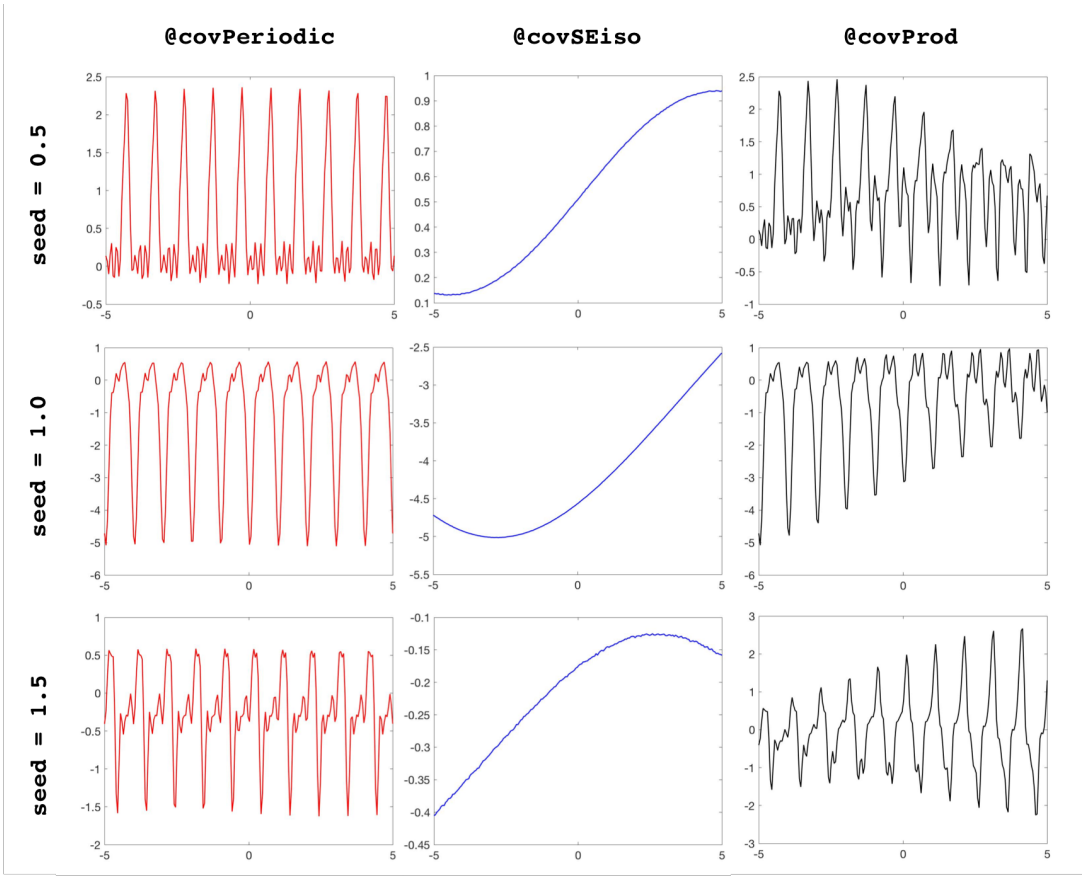

Figure 6 *Sample functions obtained by changing the random seed*

In **Command 6**, a small positive diagonal matrix is added to **K** to ensure that the covariance matrix is positive definite (a necessary condition for Cholesky decomposition).

In **Figure 6,** each column represents different covariance function, while each row represents different random seed for the gpml_randn function. It can be observed that the periodic behaviour of @covPeriodic is partially conserved in @covProd. However, since the periodic covariance function has been multiplied by another covariance function (@covSEiso), the plot is vertically distorted. Such behaviour can be explained as following:

$$k(x,z) = \sigma_{f1}^2 \exp\left(-\frac{2}{l_1^2}\sin^2\left(\frac{\pi(x-z)}{p}\right)\right) \times \sigma_{f2}^2 \exp\left(-\frac{(x-z)^2}{2l_2^2}\right)$$
$$= A(x,z) \times \exp\left(-\frac{2}{l_1^2}\sin^2\left(\frac{\pi(x-z)}{p}\right)\right)$$

The product of @covPeriodic and @covSEiso can be interpreted as a periodic covariance function with varying amplitude. This explains why there is vertical distortion, but no horizontal distortion.

5. Question E

Command 7 *Visualisation of 'cw1e.mat'*

```
surf(reshape(x(:,1),11,11),reshape(x(:,2),11,11),reshape(y,11,11));
contourf(reshape(x(:,1),11,11),reshape(x(:,2),11,11),reshape(y,11,11));
```

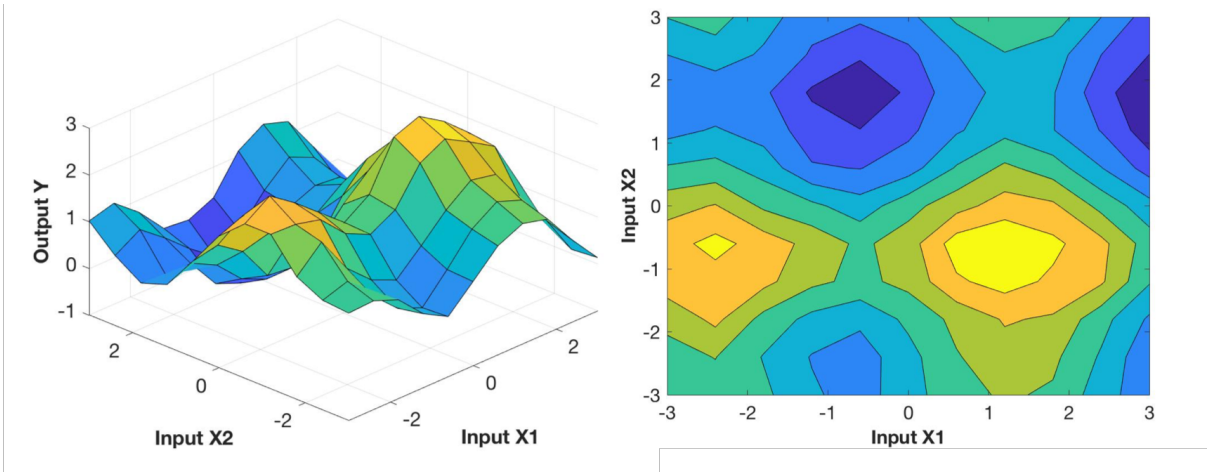

Figure 7 *Visualisation of 'cw1e.mat'*

Command 8 *Gaussian predictive with two different covariance functions*

```
covfunc = @covSEard;
hyp.cov = 0.1*randn(3,1);
```

```
covfunc = {@covSum,{@covSEard,@covSEard}};
hyp.cov = 0.1*randn(6,1);
```

```
hyp2 = minimize(hyp, @gp, -200, @infGaussLik, [], covfunc, @likGauss, x, y);
[mu s2] = gp(hyp2, @infGaussLik, [], covfunc, @likGauss, x, y, X);
surf(X1,X2,reshape(mu+2*sqrt(s2),size(X1)));
hold on; surf(X1,X2,reshape(mu-2*sqrt(s2),size(X1)));
```

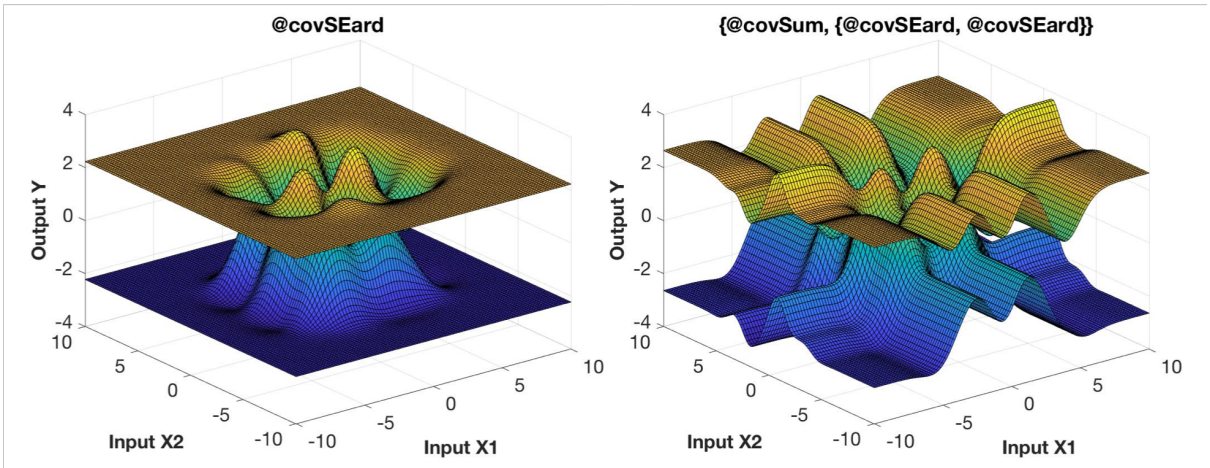

Figure 8 *Gaussian predictive with different covariance functions* (*Each surface represents* $\mu_{y_*} \pm 2\sigma_{y_*}$)

Table 4 *Hyperparameters and marginal likelihood optimized for two different covariance functions*

	λ_1	λ_2	σ_f	λ_1'	λ_2'	σ_f'	σ_n	$p(\mathbf{y}\|\mathbf{X})$
@covSEard	1.5116	1.2859	1.1073				0.1026	2.2212e+08
@covSum	1.4464	1.6e+03	1.1119	1.3e+03	0.9868	0.7135	0.0979	6.8668e+28

Figure 8 shows the two plots generated with the given covariance functions. As the test input moves away from the training data, the uncertainty grows in both plots. However, unlike the first plot which gives no information about the predictive mean, the second plot returns non-trivial predictive mean. Such behaviour can be explained by comparing the two covariance functions:

$$k_1(x,z) = \sigma_f^2 \exp\left(-\sum_{d=1}^{2}\frac{(x_d - z_d)^2}{2\lambda_d^2}\right)$$

Consider the case when x and z are close in one dimension, but far in the other dimension (i.e. $(x_1 - z_1)^2 \approx 0$ and $(x_2 - z_2)^2 \gg 0$). Since the exponent of the ARD function is given as the "sum" of the effective squared distances, large distance in only one dimension will still make $k_1(x,z)$ close to zero. This suggests that a single ARD function is not flexible enough to detect the correlation along each axis separately.

$$k_2(x,z) = \sigma_f^2 \exp\left(-\sum_{d=1}^{2}\frac{(x_d - z_d)^2}{2\lambda_d^2}\right) + \sigma_f'^2 \exp\left(-\sum_{d=1}^{2}\frac{(x_d - z_d)^2}{2\lambda_d'^2}\right)$$

Such problem can be solved by adding two ARD functions. If the symmetry in the initial hyperparameters is broken, each ARD function will be trained to model the covariance along each axis. **Table 4** shows that the first term, by having a large λ_2, became insensitive to the second dimension. Similarly, the second term became insensitive to the first dimension. Therefore, the second model can adapt itself to the training data with higher flexibility. This explains why the marginal likelihood is much higher for @covSum. In conclusion, the second model is more preferable than the first model.