4F13 Coursework #1: Gaussian Processes (GPs)

CCN: **5636B** 8th Nov 2019 word count: 995

a Training 1D data with covSEiso

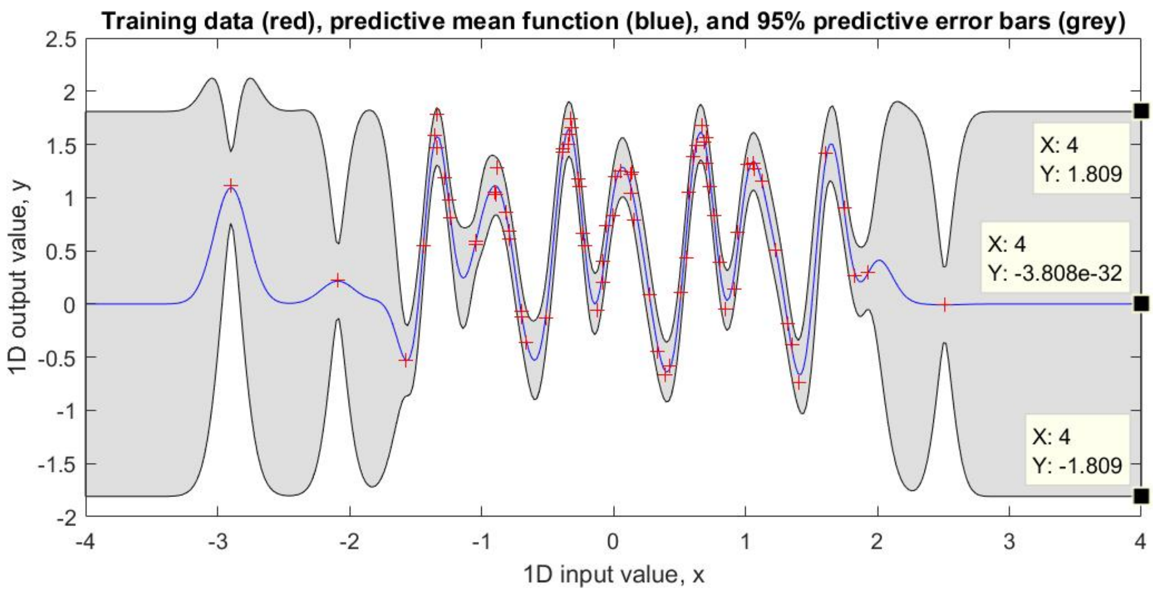


Figure 1: Data prediction from GP having trained with covSEiso.

From Figure 1, the error bars are (desirably) shorter in $x \in [-1.5, +1.5]$ where more datapoints are present, and longer outside this domain. This observation can be related to the optimized hyperparameters (hyp), which determine the shape of covariance function in the **prior** GP before training with data.

$$\text{(optimized)}\ \texttt{hyp2}: \quad ell = 0.128, \quad sf = \sigma_{prior} = 0.897, \quad sn = \sigma_{noise} = 0.118$$

Outside the mentioned domain, there are only a few datapoints. The further away a domain is from the datapoints, the more similar the **posterior** GP is to the **prior** GP (plus the effect of noise). Hence at $x = 4$, the posterior predictive variance has not changed at all from the prior variance (plus the noise variance):

$$\text{At } x = 4: \quad \sigma_{post}^2 = \sigma_{prior}^2 + \sigma_{noise}^2 = 0.897^2 + 0.118^2$$

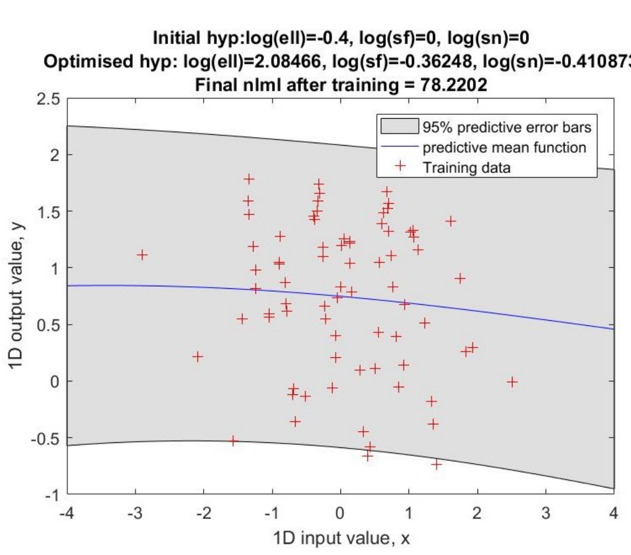
$$\Rightarrow \quad \sigma_{post} = 1.809 \quad (\textit{consistent with Figure 1})$$

In contrast, inside $x \in [-1.5, +1.5]$ the abundance of datapoints significantly reduces the predictive variance:

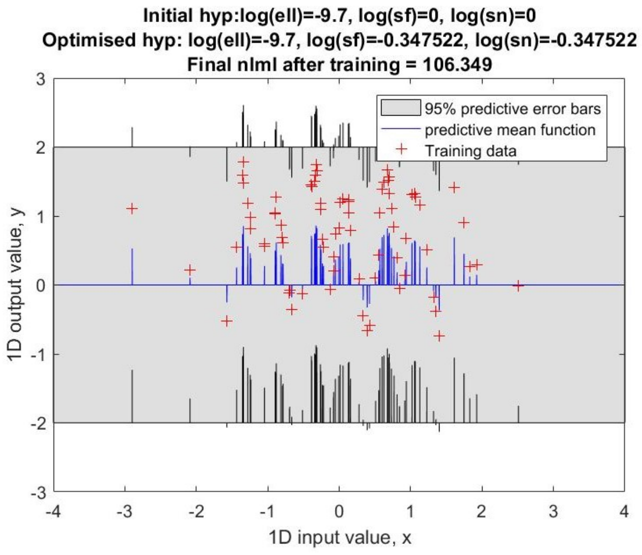
$$\sigma_{post}^2 = \sigma_{prior}^2 + \sigma_{noise}^2 - \{\text{+ve term contributed by data}\}$$

MATLAB code to optimise hyp and train GP:

```
hyp2 = minimize(hyp, @gp, -200, @infGaussLik, meanfunc, covfunc, likfunc, x, y);
xs = linspace(-4,4,801)';
[mu s2] = gp(hyp2, @infGaussLik, meanfunc, covfunc, likfunc, x, y, xs);
```



(a) large initial length-scale



(b) very small initial length-scale

Figure 2: Predictions made by varying the ell of covSEiso from the given value $log(ell) = -1$.

b Influence from hyperparameters of covSEiso

A too large or too small initial length-scale (ell) of covariance function leads to the two different minima in the negative log marginal likelihood (nlml). When it is too large (figure 2a), the optimised ell $e^{-0.4} = 0.670$ also becomes much larger than that in figure 1 (which was +0.128). As a result, the predictive mean function is enforced to be "smooth", failing to track the data. Error bars is lengthened.

In contrast, a very small ell ($e^{-9.7} = 6.1 \times 10^{-5}$) enforces the prediction to be very "wiggly", resulting in the peaks in figure 2b. Excessive requirement for oscillation also leads to failure in tracking the datapoints. Hence the predictive variance becomes large to compensate this.

Note that when instead the initial signal variance is set beyond the range $-2 < log(sf) < 0.7$ (other hyp kept unchanged), the same minima in figure 2a will be reached and same interpretation applies. Varying initial noise variance alone brings no effect, leading back to figure 1.

To analytically determine which fit is the best, nlml of each model can be consulted. A smaller nlml (larger marginal likelihood) represents a better fit to *the observed data* (details in section e). Therefore Table 1 shows that Figure 1 has the best fit. However this analysis only compares between predictions from covSEiso and hence far from exhaustive - since some periodicity can be observed in Figure 1, a periodic covariance function may offer even better result.

Model	Figure 1	Figure 2a	Figure 2b
nlml	+11.90	+78.22	+106.35

Table 1: nlml for the three predictive models using covSEiso.

MATLAB code to automatically vary initial hyp:

```
for log_ell = -12:0.1:0     % similar for log_sf and log_sn.
    log_sf = 0; hyp.cov = [log_ell; log_sf];
    log_sn = 0; hyp.lik = log_sn;
    % Then central code in question (a).
```

c Training 1D data with covPeriodic

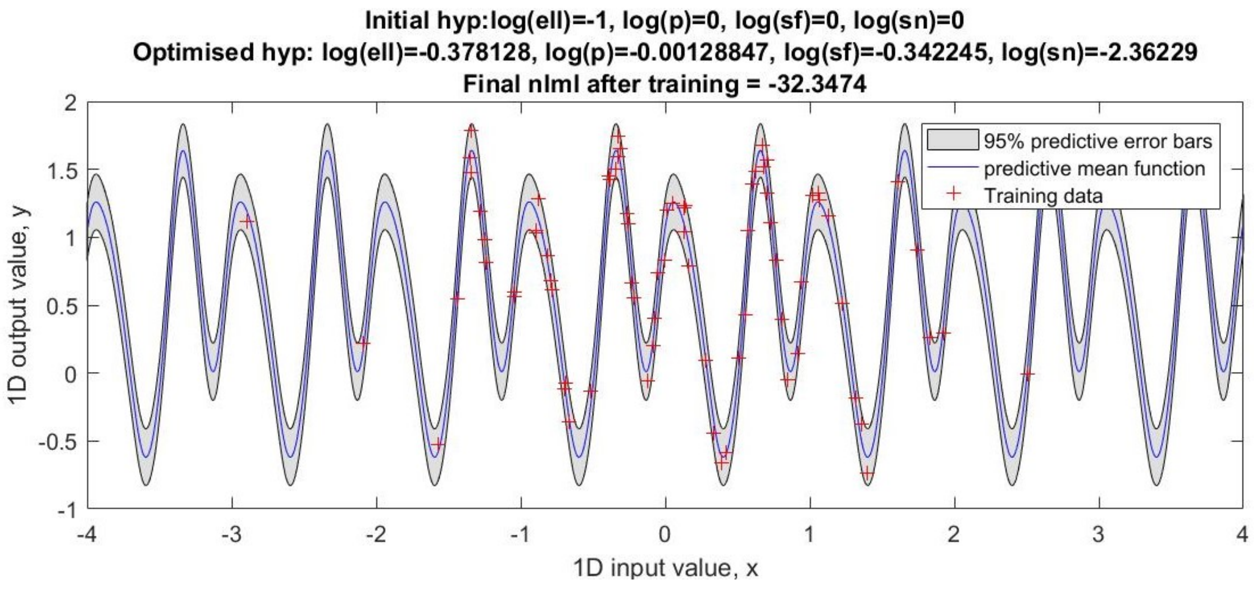


Figure 3: Data prediction from GP having trained with covPeriodic.

Setting initial period p in the range $-0.2 \leq log(p) \leq +0.2$ yields a prediction shown in Figure 3. Error bars have shortened throughout the entire x domain when compared to Figure 1. Very noticeably, even outside $x \in [-1.5, 1.5]$, this new predictive function still has the same (small) variance as inside the domain. A periodic covariance function has given rise to a periodic **posterior** signal variance.

Evaluating the data fit merely from its nlml, the periodic prediction significantly outperforms the squared exponential one (-32.35 vs. $+11.90$). However this high performance only corresponds to the data fit on the **observed datapoints**. Even though within $x \in [-1.5, 1.5]$ the observed datapoints can map out a periodicity, there are not enough datapoints outside the domain to justify the same periodicity. The data generating mechanism could likely be non-periodic outside.

Central MATLAB code:

```
% Code is the same as in question (a) except:
covfunc = @covPeriodic;     % it was @covSEiso in (a).
log_ell = -1; log_p = 0; log_sf = 0;
hyp.cov = [log_ell; log_p; log_sf];  % An extra hyperparameter log_p is present.
```

d Interpretations of covariance functions

d.1 Resolving floating point rounding error

To generate random functions of a GP, Cholesky decomposition of its covariance matrix $\mathbf{K}$ needs to be computed. Although in theory $\mathbf{K}$ is always positive definite and Cholesky decomposition can (only) be applied to any positive definite matrix, MATLAB might throw an error to the snippet below. This is caused by the rounding error in MATLAB.

Even though by definition a square matrix is positive definite iff all its eigenvalues are strictly positive, some of them can take very small values. Beyond a certain precision, MATLAB will round down those very small eigenvalues to zero. The matrix will no longer be positive definite, hence

Cholesky decomposition will not work. To address this problem, a not-too-small diagonal matrix $\Delta =$ can be manually added to $\mathbf{K}$:

$$\text{From diagonalisation of } K, \quad K + \Delta = U^T \Lambda U + U^T \Delta^* U = U^T(\Lambda + \Delta^*)U$$

where Λ contains all the eigenvalues of $\mathbf{K}$, and Δ^* is another small diagonal matrix actually adding to the eigenvalues.

MATLAB code to generate a random function:

```
x = linspace(-5,5,n)';
z = gpml_randn(seed, n, 1);
f = chol( K + (1e-6*eye(n)) )'*z;
plot(x,f)
```

d.2 Composite covariance function: covPeriodic $\times$ covSEiso

Different covariance functions can be combined to model different features of the observed data. For example, covPeriodic generates periodic random functions with constant amplitudes (Figure 4a); covSEiso leads to functions with random oscillation (Figure 4b). By constructing a composite covariance function covPeriodic $\times$ covSEiso, and setting the length-scale of covSEiso large (the question recommends $ell = e^2 = 7.4$), periodic functions with **varying** amplitudes can be produced. Figure 4c shows that the amplitude will vary gradually - this is because the large $ell2$ enforces covSEiso to vary slowly as the difference $(x - x')$ increases:

$$\text{composite covfunc:} \quad k(x, x') = (sf1)^2 exp\{-\frac{2sin^2(\pi(x-x')/p)}{(ell1)^2}\} \times (sf2)^2 exp\{-\frac{(x-x')^2}{2 \times (ell2)^2}\}$$

e Training 2D data with single/double covSEard

Fitting data to the two specified GPs yields the predictions in Figure 5. Each sub-figure is an overlap of four surfaces as explained in the figure caption. Since each surface is semi-transparent, red region means (red) data surface is on top of (black) predictive mean surface; black region means predictive mean surface is on the top. The alternating colours in the figures indicate that the mean function is tracking the data with alternating undershoots and overshoots. The observation that the error bound surfaces (pale grey) closely track the "red-black surfaces" in both sub-figures primarily suggests that both models fit the data well.

To analytically compare the data fits, nlml can be consulted again - covSum model largely defeats covSEard model (-66.4 vs. -19.2). In fact, marginal likelihood optimises fit to the observed data in conjunction with model complexity:

$$log[p(\mathbf{y}|\mathbf{x}, M_i)] = -\frac{1}{2}\mathbf{y}^T[K + \sigma_n^2 I]^{-1}\mathbf{y} - \frac{1}{2}log|K + \sigma_n^2 I| - \frac{n}{2}log(2\pi)$$

Since covSum model is more complex than covSEard, $\frac{1}{2}log|K + \sigma_n^2 I|$ is larger, which undesirably lowers the marginal likelihood (raises the nlml). The reason why covSum still have a much lower nlml is that the much better fit to the observed data has justified the use of a more complex model. A closer look into the cross sections of the 3D prediction graphs (Figure 6) demonstrates that predictive error bar of the covSum model is shorter.

Lastly, the reason why covSum, the sum of two covSEard, offers a better data fit than a single covSEard is as follows. With single covSEard, $ell1$ and $ell2$ are optimised collectively in order to capture the data oscillation in both dimensions. In comparison, double covSEard allows each

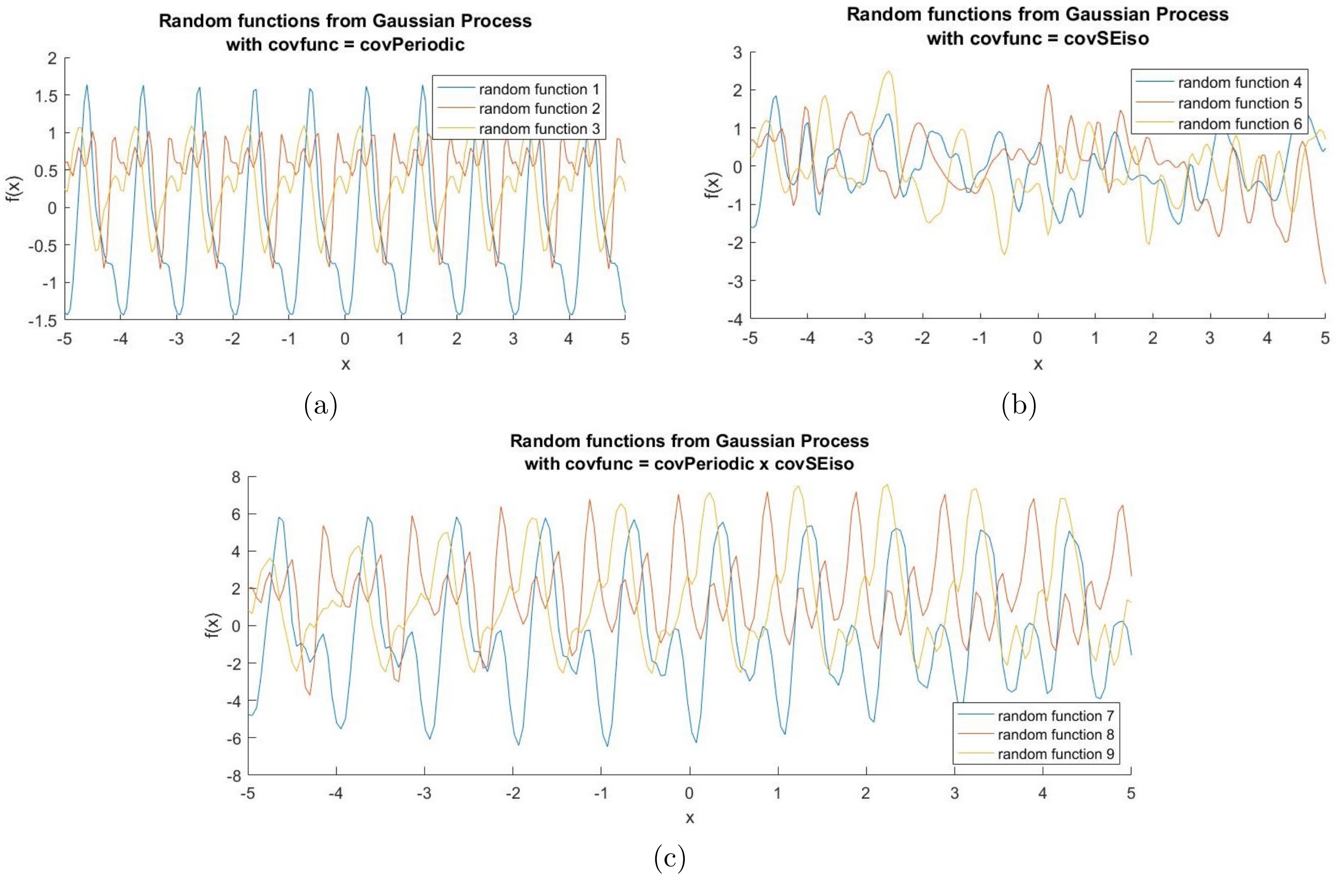


Figure 4: Random functions generated by a GP with covariance function (a) covPeriodic, (b) covSEiso, (c) covPeriodic × covSEiso

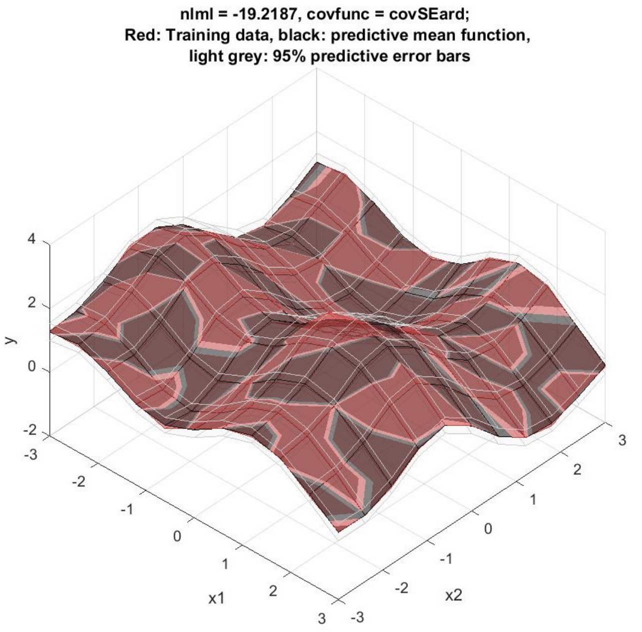
dimension to be modelled separately by each covSEard, hence greater flexibility in data fitting. Table 2 shows how in covSum the first dimension is merely fitted by *ell*1, and the second dimension merely fitted by *ell*4; *ell*2 and *ell*3 are "switched off" due to their large values.

covfunc	$ell1$	$ell2$	$sf1$	$ell3$	$ell4$	$sf2$	sn
covSEard	1.51	1.29	1.11	–	–	–	0.103
covSum	1.45	1364	1.12	1038	0.99	0.71	0.098

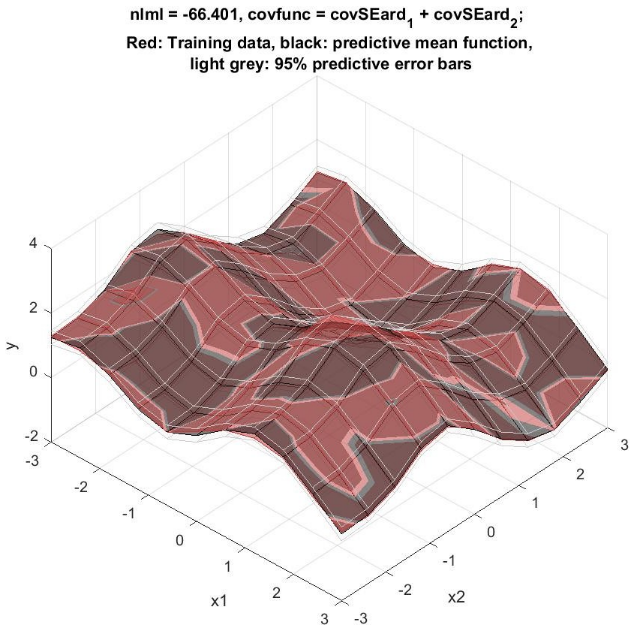
Table 2: Optimised hyperparameters of the two covariance functions.

Central MATLAB code to plot each surface in Figure 5:

```
Colour(:,:,1) = ones(11)*0.8; % red
Colour(:,:,2) = ones(11)*0.2; % green
Colour(:,:,3) = ones(11)*0.2; % blue
mesh(reshape(x(:,1),11,11), reshape(x(:,2),11,11),reshape(y,11,11), Colour, ...
    'FaceAlpha','0.5', 'FaceColor', 'flat')
```

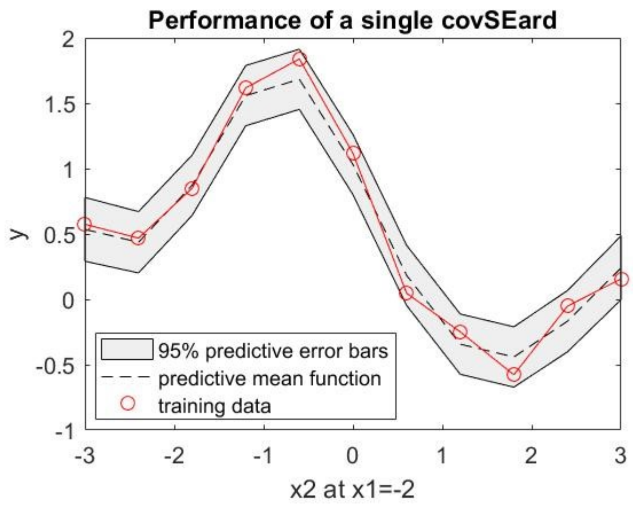


(a)

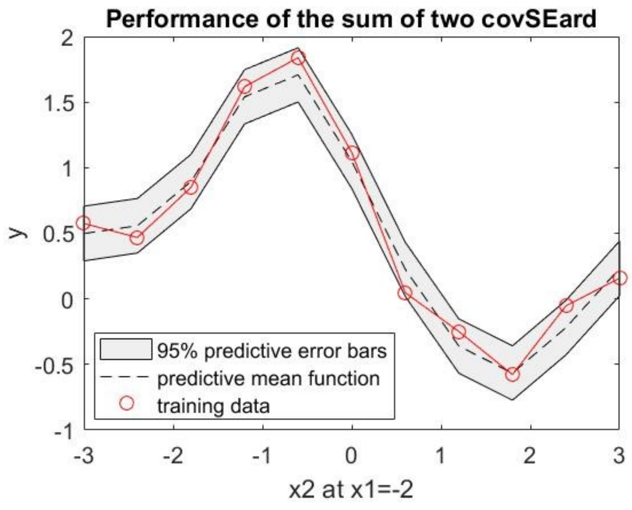


(b)

Figure 5: Prediction from GP with covariance function (a) covSEard, (b) covSum = covSEard$_1$ + covSEard$_2$. **Red surface** - observed data; **black surface** - predictive mean; **pale grey** - 95% predictive error bounds (upper and lower).



(a)



(b)

Figure 6: Cross sections of (a) Figure 5a and (b) Figure 5b at $x_1 = 2$.