4F13 Coursework 1: Gaussian Processes

Word Count: 992

November 3, 2022

Problem a)

Here is the MATLAB code for GP training and 95% predictive error bars in a):

```
meanfunc = []; covfunc = @covSEiso; likfunc = @likGauss;
hyp.cov = [-1  0]; hyp.lik = 0; xs = linspace(-3,3,900)'; % 900 test
    points
hyp = minimize(hyp,@gp,-100,@infGaussLik,meanfunc,covfunc,likfunc,x,
    y);
[mu s2] = gp(hyp,@infGaussLik,meanfunc,covfunc,likfunc,x,y,xs);
```

The optimized length scale l, scaling factor σ_f^2 and observation noise variance σ_{noise}^2 are 0.1282, 0.8046 and 0.0139 respectively. Also, here is the predictive error bar plot:

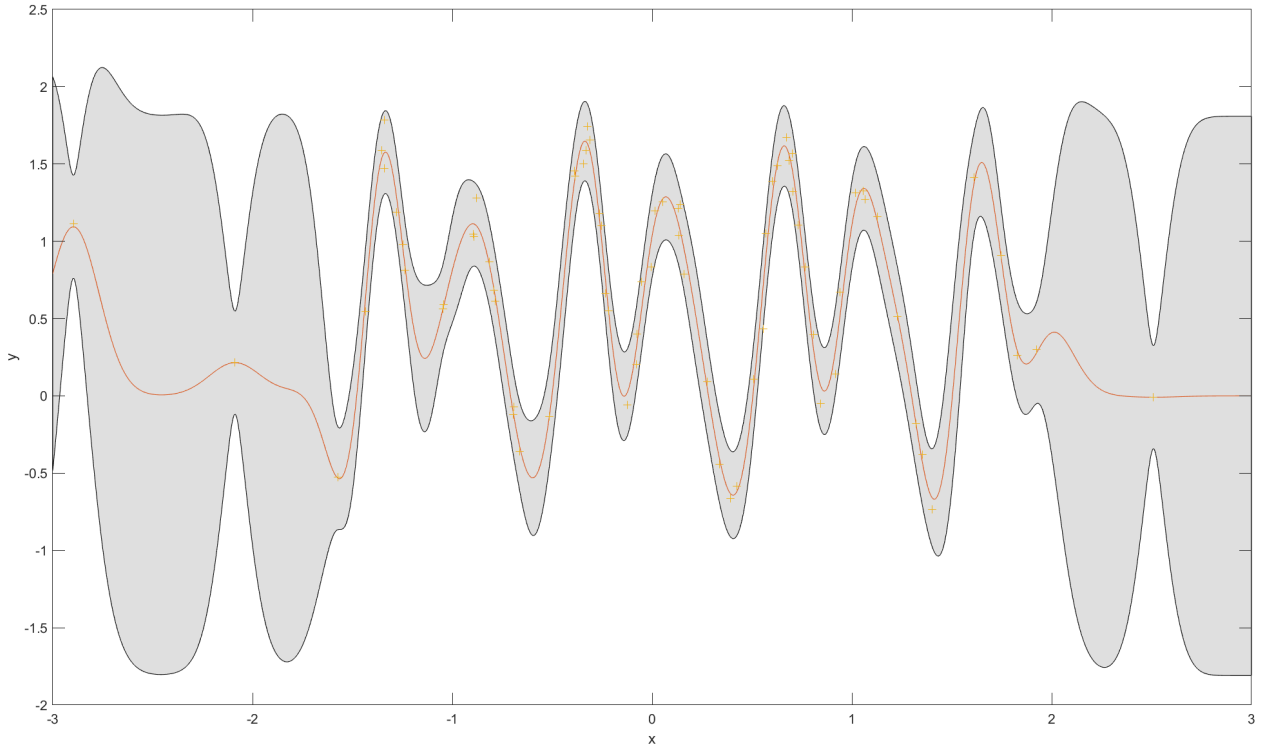


Figure 1: GP Regression Plot for cw1a data with Square Exponential Covariance Function

Given Figure 1, we can see that the error bars are narrow in the range $[-1.5, 1.5]$ and are larger in other range of x, which is reasonable as most training data lies within $[-1.5, 1.5]$ and thus we have more information about the underlying data generating function in this range and this reduces the variance of the posterior predictive distribution at $x \in [-1.5, 1.5]$.

Let's look at the optimized hyperparameters. For l, as from Figure 1, the fitted posterior predictive function is wriggled, indicating a small value for l and thus a value of 0.1282 is reasonable. A value of 0.8046 for σ_f^2 indicates that on average each function in the GP Prior is 0.8046 away from its mean (0 in this case), which makes sense as most of the y values of the training data are in $[-0.7, 1.5]$. σ_f^2 should not be too large, otherwise the modelled GP Posterior might be distracted severely to extreme values in training data set and leads to overfitting. For σ_{noise}^2, a value of 0.0139 indicates little noise in the evaluation of the underlying data generating function, which is reasonable as the predictive error bars are relatively narrow for $x \in [-1.5, 1.5]$.

Problem b)

The initial values of the log hyperparameters are set at hyp.cov $\in \{[-3\ \ -1], [-1\ \ 1], [0\ \ 2], [1\ \ 4]\}$ and hyp.lik $\in \{0, 3, 5, 7\}$. Thus, there are 16 different combinations of initializations. For each combination, the GP model was first fitted:

```
hyp = struct('mean',[],'cov',Cov(j,:),'lik',Lik(k));
hyp = minimize(hyp,@gp,-100,@infGaussLik,meanfunc,covfunc,likfunc,x,
    y);
[mu s2] = gp(hyp,@infGaussLik,meanfunc,covfunc,likfunc,x,y,xs);
```

For each combination, the fit of the GP model was plotted:

```
f = [mu+2*sqrt(s2); flipdim(mu-2*sqrt(s2),1)];
subplot(length(Cov),length(Lik), length(Lik)*(j-1)+k); fill([xs;
    flipdim(xs,1)], f, [7 7 7]/8)
hold on; plot(xs, mu); plot(x, y, '+'); xlabel('x'); ylabel('y');
title(strcat('cov = ',mat2str(Cov(j,:)),'and lik = ',mat2str(Lik(k))
    ));
```

The resultant 16*16 GP fit plot is:

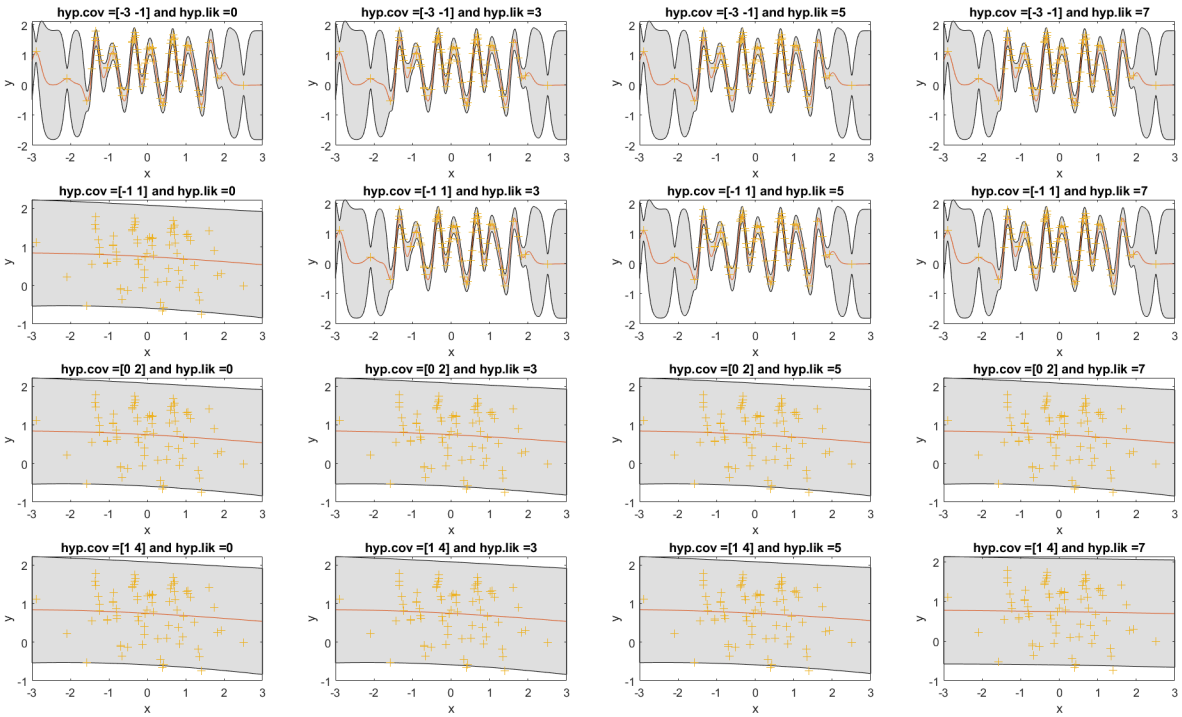


Figure 2: GP Regression Plot for cw1a data with Square Exponential Covariance Function at different initial log hyperparameters

The hyperparameters are found by minimizing the negative log marginal likelihood. For improper initializations, the minimize() optimizer might converge to a local minima or not converge within the maximum number of iterations allowed. Both the information from the GP prior and the information from the training data are used to the GP model and gives the GP posterior and predictive distributions. If the hyperparameters are bad, little information could be extracted from the training data and hence the GP posterior could not explain much of the training data.

The variance of the predictive distribution is $\mathrm{V}(x_*) = \mathrm{k}(x_*, x_*) + \sigma^2_{\text{noise}} - \mathrm{k}(x_*, \mathbf{x})[\mathrm{K} + \sigma^2_{\text{noise}}\mathrm{I}]^{-1}\mathrm{k}(\mathbf{x}, x_*)$, then bad hyperparameters will result to small $\mathrm{k}(x_*, \mathbf{x})[\mathrm{K} + \sigma^2_{\text{noise}}\mathrm{I}]^{-1}\mathrm{k}(\mathbf{x}, x_*)$ and thus $\mathrm{V}(x_*)$ would be dominated by the prior variance $\mathrm{k}(x_*, x_*)$ and the the predictive distribution would be similar to GP prior, just as the plot for hyp.cov = [1 4] and hyp.lik = 7 in Figure 2.

From Figure 2, initializing with hyp.cov = $[-3 \ -1]$ with hyp.lik = 3 works well, where its negative log marginal likelihood is also small (11.8990) compared to the negative log marginal likelihood for the underfitted GP models (78.2202). Thus, I could confidently say that an initialization of hyp.cov = $[-3 \ -1]$ with hyp.lik = 3 does lead to a good GP fit.

Problem c)

Here is the MATLAB code for GP fitting in c):

```
meanfunc = []; covfunc = @covPeriodic; likfunc = @likGauss;
hyp.cov = [-1 0 0]; hyp.lik = 0;
hyp = minimize(hyp,@gp,-600,@infGaussLik,meanfunc,covfunc,likfunc,x,
    y);
[mu s2] = gp(hyp,@infGaussLik,meanfunc,covfunc,likfunc,x,y,xs);
```

Here is the plot of GP fit:

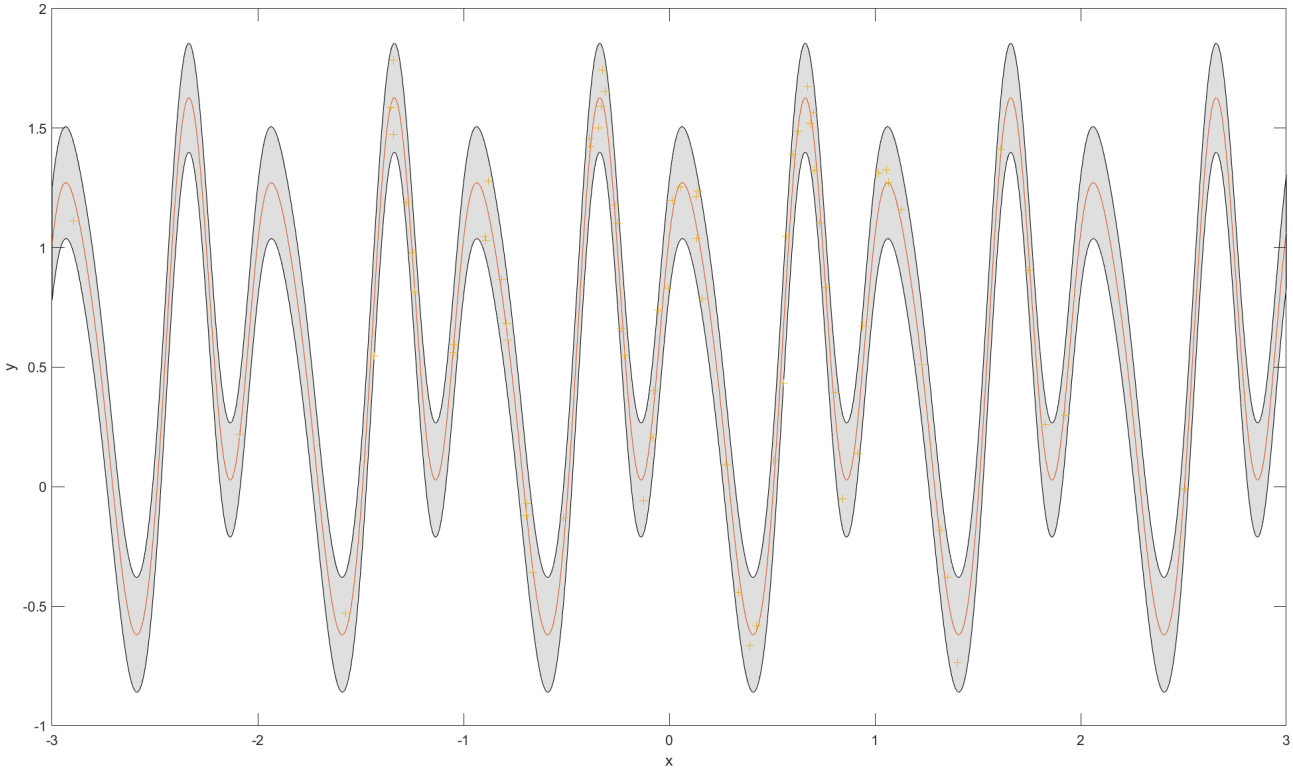


Figure 3: GP Regression Plot for cw1a data with Periodic Covariance Function

Compare to the GP fit in a), both of the models have narrow predictive error bars for $x \in [-1.5, 1.5]$ where most training data lies. However, for x outside $[-1.5, 1.5]$, the GP model with periodic kernel has much narrower predictive error bars (hence lower variance) than the square exponential GP model in a). Moreover, for $x \in [-1.5, 1.5]$, both of the predictive mean functions show similar periodic patterns.

Although both GP predictive distribution has shown periodic structures in $x \in [-1.5, 1.5]$ and a lower negative log likelihood for periodic GP model (-35.2669 compared to 11.8990 in a)) indicates that periodic kernel is preferable, there is insufficient training data outside this range (the predictive error bar of the square exponential GP model is large outside $x \in [-1.5, 1.5]$). Moreover, there exists many other non-periodic functions that could also fit the training data well. Thus, a good data fit for periodic kernel GP model does not imply a strictly periodic data generating mechanism.

Problem d)

Here is the MATLAB code for function generation:

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

The reason for adding $1e-6*\mathrm{I}$ (I is 200*200 identity matrix) to covariance matrix K is that K is not positive definite here (it should be positive definite in principle) due to numerical instability in MATLAB and Cholesky decomposition only applies to positive definite matrix. By adding a small diagonal matrix $1e-6*\mathrm{I}$ to K, we have K+$1e-6*\mathrm{I}$ which is always positive definite and whose values are almost the same as K. Thus we can apply Cholesky decomposition on K+$1e-6*\mathrm{I}$ to draw functions from GP.

Here is the four sampled functions:

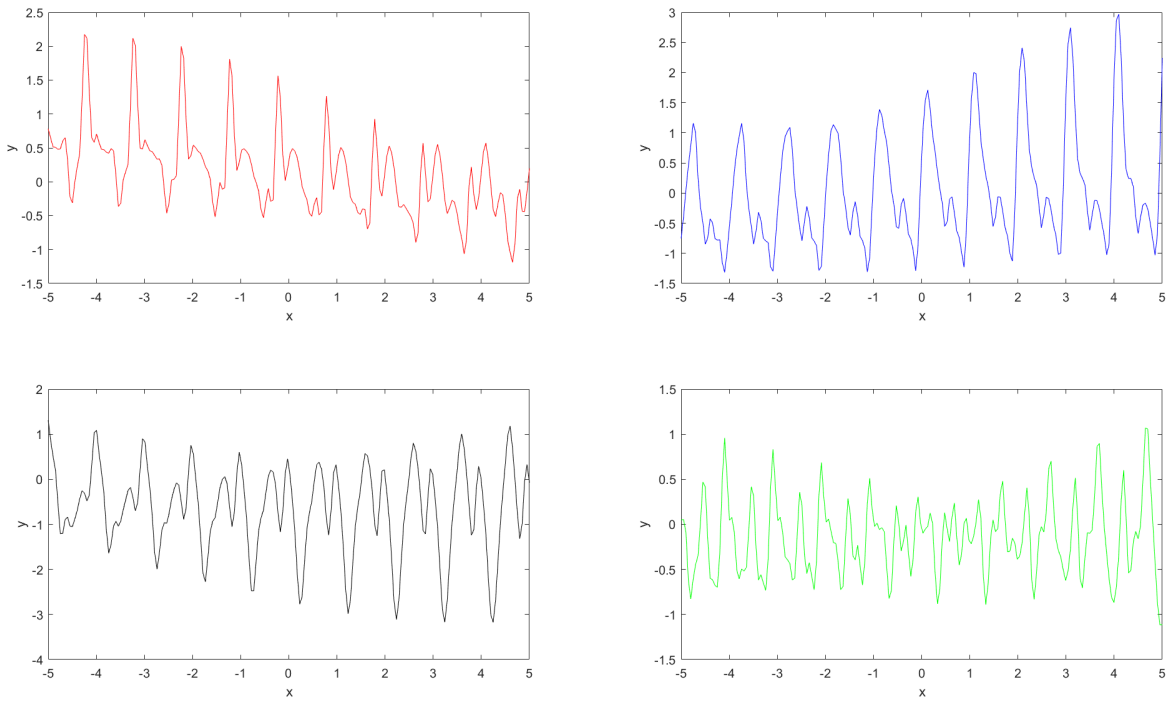


Figure 4: Four sampled functions from GP with the Product of Square Exponential and Periodic Kernel

The covariance function for the GP is the product of a square exponential and a periodic kernel, thus it entails the properties of both kernels. From Figure 4, all functions has periodic patterns which corresponds to the property of periodic kernel (imposing periodicity on sampled functions). Also, the shape of the repeating periodic part as well as the amplitude of the sampled functions changes with x. This is the result of multiplying the periodic kernel by a square exponential kernel which allows variations of periodic patterns of sampled functions.

Problem e)

Here is the MATLAB code for model fitting:

```
meanfunc = []; covfunc = @covSEard; likfunc = @likGauss; % Fitting
    with covSEard
hyp1.cov = [-1 0 0]; hyp1.lik = 0;
hyp1 = minimize(hyp1,@gp,-100,@infGaussLik,meanfunc,covfunc,likfunc,
    x,y);
[mu_ardse s2_ardse] = gp(hyp1,@infGaussLik,meanfunc,covfunc,likfunc,
    x,y,xs);
meanfunc = []; covfunc = {@covSum, {@covSEard, @covSEard}}; likfunc
    = @likGauss; % Fitting with {@covSum, {@covSEard, @covSEard}}
hyp2.cov = 0.1*randn(6,1); hyp2.lik = 0;
hyp2 = minimize(hyp2,@gp,-100,@infGaussLik,meanfunc,covfunc,likfunc,
    x,y);
[mu_ardse_sum s2_ardse_sum] = gp(hyp2,@infGaussLik,meanfunc,covfunc,
    likfunc,x,y,xs);
```

And the plot of training data:

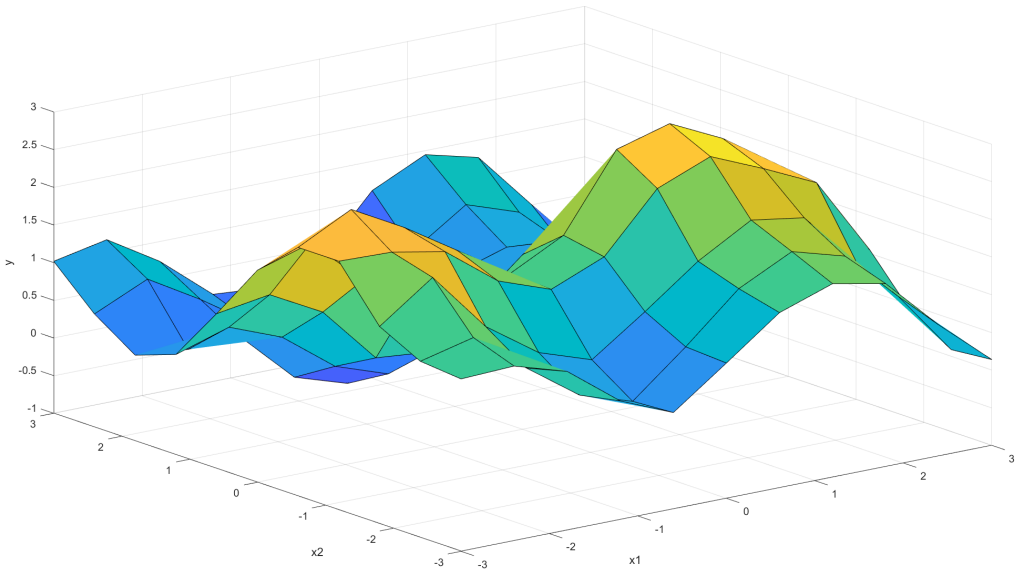
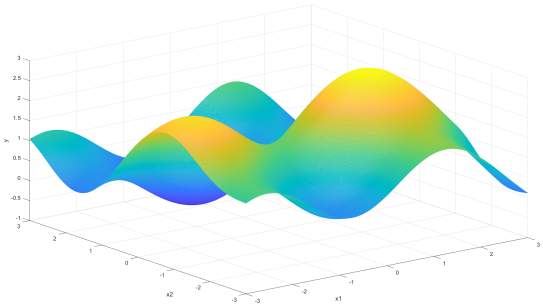
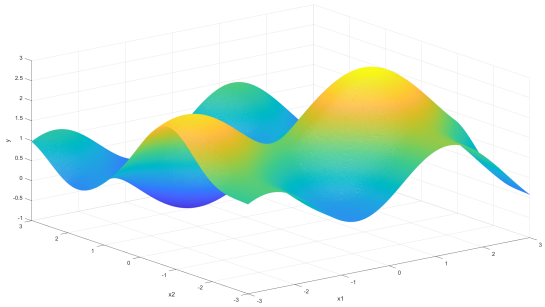


Figure 5: 3d visualization of cw1e.mat data

The 3d visualizations of the two GP model fits are:

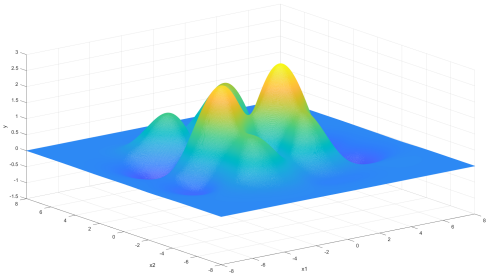


(a) Predictive mean plot with @covSEard

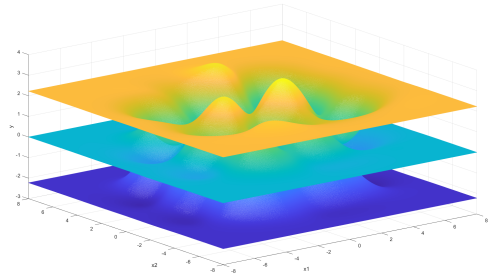


(b) Predictive mean plot with {@covProd, {@covPeriodic, @covSEiso}}

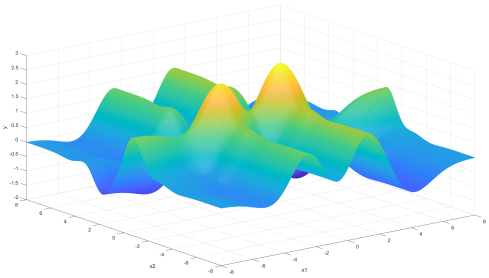
Figure 6: 3d visualizations of the predictive mean values of the two GP models with @covSEard and {@covProd, {@covPeriodic, @covSEiso}} kernel with test points $x_1, x_2 \in [-3, 3]$



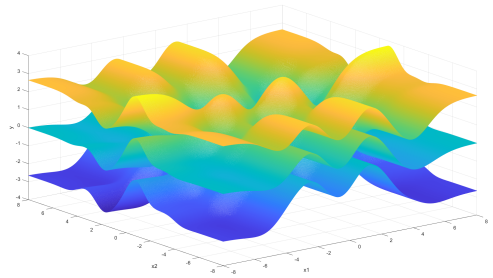
(a) Predictive mean plot with @covSEard



(b) Error bars with @covSEard



(c) Predictive mean plot with {@covProd, {@covPeriodic, @covSEiso}}



(d) Error bars with {@covProd, {@covPeriodic, @covSEiso}}

Figure 7: 3d visualizations of the two GP models with @covSEard and {@covProd, {@covPeriodic, @covSEiso}} kernel with test points $x_1, x_2 \in [-8, 8]$ (The middle surfaces in (b) and (d) are predictive mean μ_* while the upper and lower surfaces in (b) and (d) are $\mu_* \pm 2\sigma_*$, where σ_* stands for predictive standard deviation.)

From Figure 6 and 7, both GP models fit the training data well within $x_1, x_2 \in [-3, 3]$. When the test data gets further away from the region where most training data lies, the @covSEard model returns a flat predictive mean around 0 (provides no useful information) while the {@covProd, {@covPeriodic, @covSEiso}} model still remains wriggled (provides useful information).

The log marginal likelihood is $-\frac{1}{2}\mathbf{y}^\intercal[\mathrm{K}+\sigma^2_{\text{noise}}\mathrm{I}]^{-1}\mathbf{y}-\frac{1}{2}\log|\mathrm{K}+\sigma^2_{\text{noise}}\mathrm{I}|-\frac{n}{2}\log 2\pi$, which comprises a data fit term ($-\frac{1}{2}\mathbf{y}^\intercal[\mathrm{K}+\sigma^2_{\text{noise}}\mathrm{I}]^{-1}\mathbf{y}$) and a complexity penalty term ($\frac{1}{2}\log|K+\sigma^2_{\text{noise}}\mathrm{I}|$). The {@covProd, {@covPeriodic, @covSEiso}} model here has more flexibility and hence leads to higher values of the data fit term (as it could explain the training data better), which outweighs the increases in its complexity penalty term (due to increased model complexity) and this explains the larger log marginal likelihood for the {@covProd, {@covPeriodic, @covSEiso}} model (66.3785 compared to 19.2187).