Gaussian Processes

Probabilistic Machine Learning

Abhishek Shenoy

Total words = 998

[Excluding tables, figures, code, equations, abstract and references]

Abstract

This report focuses on using Gaussian Processes (GPs) for regression where GPs with various covariance functions are evaluated based on their ability to fit the datasets.

1 Part A

The prior distribution is defined by a Gaussian process with zero mean and the squared-exponential function:

$$P(f) \sim \mathcal{GP}(m=0, k_{\text{SE}})$$
$$k_{\text{SE}}(x, x') = \sigma_f^2\, e^{-\dfrac{(x-x')^2}{2l^2}} \quad (1)$$

Listing 1. GP @CovSEiso

```
meanfunc = @meanZero;     % Zero-mean func
covfunc = @covSEiso;      % Squared-Exponential covariance func
likfunc = @likGauss;      % Gaussian likelihood

% initial values for the log hyperparameters
hyp = struct('mean', [], 'cov', [-1 0], 'lik', 0);

% minimize the negative log likelihood
hyp2 = minimize(hyp, @gp, -200, @infGaussLik, meanfunc, ...
       covfunc, likfunc, x, y);

[mu s2] = gp(hyp2, @infGaussLik, meanfunc, covfunc, likfunc, ...
       x, y, xs);
```

HypA	l	σ_f	σ_n	p($\boldsymbol{y}$|$\boldsymbol{x}$)
Initial Values	0.3769	1	1	4.4636e-41
Optimised Values	0.1282	0.8970	0.1178	6.7972e-6

Table 1. Hyperparameters and marginal likelihood for $\mathcal{MA}$

For the predictive distribution (Fig 1), the range of the 95% confidence interval increases as the test point $x^\star$ moves away from the nearest training datapoint x_{nt} and decreases as it moves closer to the datapoint but is always greater than 0.

$$p(\boldsymbol{y}^\star|\boldsymbol{x}^\star, \boldsymbol{x}, \boldsymbol{y}) \sim \mathcal{N}(m(x^\star), V(x^\star))$$
$$m(x^\star) = \boldsymbol{k}(x^\star, \boldsymbol{x})^\top[\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2\boldsymbol{I}]^{-1}\boldsymbol{y}$$
$$\underbrace{V(x^\star)}_{\sigma_{y^\star}^2} = \underbrace{k(x^\star, x^\star)}_{\sigma_f^2} + \sigma_n^2 - \underbrace{\boldsymbol{k}(x^\star, \boldsymbol{x})^\top[\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{x}) + \sigma_n^2\boldsymbol{I}]^{-1}\boldsymbol{k}(x^\star, \boldsymbol{x})}_{r} \quad (2)$$

The predictive variance $V(x^\star)$ consists of 2 significant terms σ_f^2 which is the prior variance and r which is a strictly-positive term that reduces the overall variance by how much the

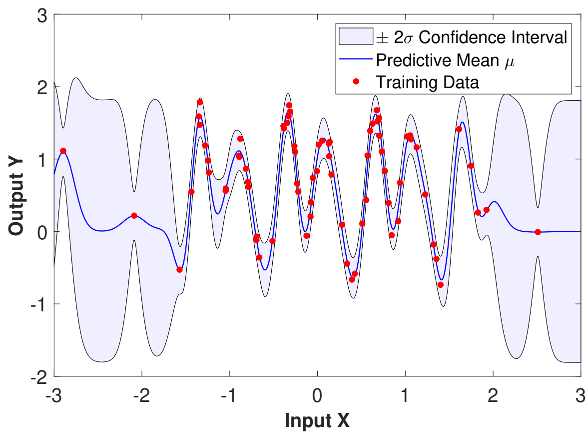


Figure 1. $\mathcal{MA}$ predictive distribution with @CovSEiso (HypA)

data $\boldsymbol{x}$ has explained. σ_n^2 is the constant noise added from the Gaussian likelihood (independent of $x^\star$) and hence $V(x^\star) > 0$.

The maximum range of the confidence interval given by $\pm 2\sigma_{y^\star} = \pm 2\sqrt{\sigma_f^2 + \sigma_n^2}$ is satisfied in Fig 1 by our calculated range of ± 1.809 from the predictive mean.

The length-scale of $l = 0.1282$ is reasonable as the comparatively small value implies a greater fluctuation in the predictive distribution which is seen in Fig 1 by the wave-like variations.

2 Part B

Using an alternate hyperparameter initialisation (HypB), we underfit the dataset as the optimised marginal likelihood in Table 2 is lower than that of Table 1. The large length-scale $l = 8.0414$ is also clearly shown in Fig 2 as the distribution encompasses all datapoints with little variation since the points are all considered as noise.

Listing 2. Hyperparameters B

```
% initial values for the log hyperparameters
hyp = struct('mean', [], 'cov', [0 0], 'lik', 0);
```

HypB	l	σ_f	σ_n	p($\boldsymbol{y}$|$\boldsymbol{x}$)
Initial Values	1	1	1	3.0107e-40
Optimised Values	8.0414	0.6959	0.6631	1.0700e-34

Table 2. Hyperparameters and marginal likelihood for $\mathcal{MB}$

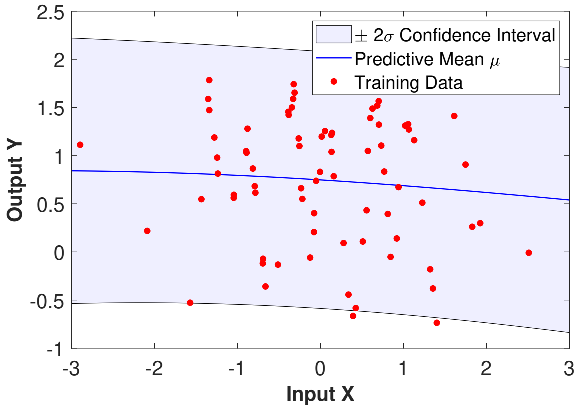


Figure 2. $\mathcal{MB}$ predictive distribution with @CovSEiso (HypB)

Listing 3. Hyperparameter variation for best fit

```
hyp = struct('mean', [], 'cov', [xx(i) 0], 'lik', ...
        yy(j));

[nlZ, dnlZ] = gp(hyp, @infGaussLik, meanfunc, ...
        covfunc, likfunc, x, y);

Z(j,i) = min(log(nlZ),5);
```

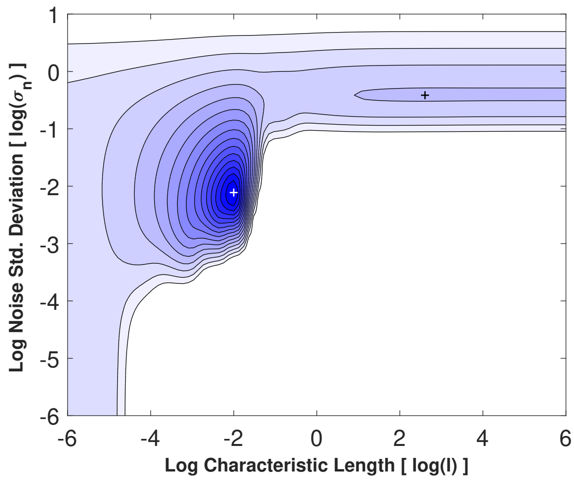


Figure 3. Hyperparameter variation to show local minima with $\sigma_f = 1$

We generate a contour plot of the negative log marginal likelihood (Fig 3) for different values of σ_n and l and observe two minima of which the global minimum corresponds to $\mathcal{MA}$ (Fig 1) with small l and σ_n. The local minima corresponding to $\mathcal{MB}$ (Fig 2) with large l and σ_n occurs when σ_n is just enough to explain the training data. If σ_n is too big, the likelihood function will be flat resulting in lower marginal likelihood. Therefore $\mathcal{MA}$ has the best fitting since the marginal likelihood is much higher for HypA (Tab 1) than for HypB (Tab 2).

Note that we assume $\sigma_f = 1$ to generate the contours in Figure 3 because as $\sigma_f \to \infty$ or as $\sigma_f \to 0$, we notice that the best minimum (as in Fig 3) disappears meaning there exists a single optimum for σ_f that is independent of σ_n and the length-scale l.

3 Part C

$$k_{\text{PER}}(x, x') = \sigma_f^2 \, e^{-\frac{2}{l^2} \sin^2 \frac{\pi}{p}(x - x')} \tag{3}$$

Listing 4. GP @CovPeriodic

```
% initial values for the log hyperparameters
meanfunc = [];
covfunc = @covPeriodic; % Periodic covariance func
likfunc = @likGauss;

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

HypC	l	σ_f	σ_n	p	$p(\boldsymbol{y}\|\boldsymbol{x})$
Initial Values	1	1	1	1	2.8391e-35
Optimised Values	1.0707	1.2414	0.1093	0.9989	2.0452e+15

Table 3. Hyperparameters and marginal likelihood for $\mathcal{MC}$

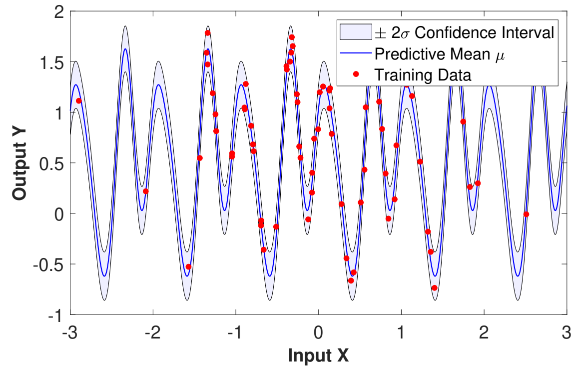


Figure 4. $\mathcal{MC}$ predictive distribution with @CovPeriodic (HypC)

By comparing the predictive distributions using k_{SE} (Fig 1) with k_{PER} (Fig 4), we see that periodicity is maintained even with sparse training points for Figure 4. The error-bars in Figure 1 widen drastically outside the training data domain while the error-bars in Figure 4 are maintained very close to the predictive mean.

The optimised marginal likelihood for $\mathcal{MC}$ in Table 3 is many magnitudes higher than for $\mathcal{MA}$ in Table 1. Therefore $\mathcal{MC}$ provides a better fit than $\mathcal{MA}$ and since $\mathcal{MC}$ uses a periodic covariance function, it implies the dataset must have some periodicity with added noise. Furthermore the predictive distribution given by $\mathcal{MA}$ (Fig 1) presents a similar periodicity to that of $\mathcal{MC}$ ($p \approx 1$ in Table 3) for which the prior GP of $\mathcal{MA}$ did not use a periodic covariance function suggesting that the dataset is periodic.

We verify the dataset is periodic by computing the residuals of the actual output y with the predicted output $\hat{y}$ from the GP and create a hypothesis test for which the null hypothesis assumes added Gaussian noise with unknown mean and variance.

$$H_0 : y - \hat{y} = \mathcal{N}(\mu_r, \sigma_r^2) \tag{4}$$

Listing 5. Residual hypothesis test

```
[y_pred, s2] = gp(hyp, @infGaussLik, meanfunc, ...
       covfunc, likfunc, x, y, x);

y_resid = zeros(size(y));
for i=1:length(y_resid)
     y_resid(i) = y(i)-y_pred(i);
end

y_resid_std = std(y_resid)
for i=1:length(y_resid)
     y_resid(i) = y_resid(i)/y_resid_std;
end


[h, p, jbstat, critval] = jbtest(y_resid, [], 0.001);

hist(y_resid, 10);
```

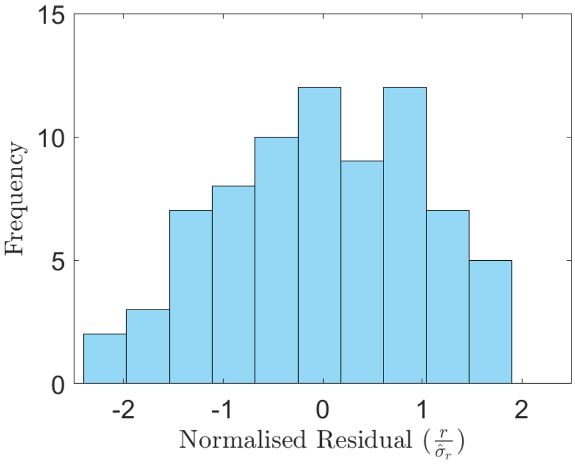


Figure 5. Histogram of residuals

By performing the Jarque-Bera test, we can check if the residuals are drawn from a Gaussian distribution. The p-value obtained from the test is 0.2150 and hence H_0 cannot be rejected. By plotting the histogram of residuals (Fig 5), we see a Gaussian-like distribution making it reasonable to assume the data was generated by adding Gaussian noise to a periodic function.

4 Part D

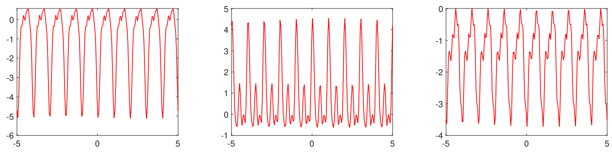
$$\begin{aligned} k_{\text{prod}}(x,x') &= k_{\text{PER}}(x,x') \cdot k_{\text{SE}}(x,x') \\ &= \sigma_{f_1}^2 \, e^{-\frac{2}{l_1^2}\sin^2\frac{\pi}{p}(x-x')} \cdot \sigma_{f_2}^2 \, e^{-\frac{(x-x')^2}{2l_2^2}} \\ &= A(x,x') \cdot e^{-\frac{2}{l_1^2}\sin^2\frac{\pi}{p}(x-x')} \end{aligned} \tag{5}$$

We can generate sample functions from a GP defined by the covariance function k_{prod} in Equation (5) by drawing samples from a standard multivariate Gaussian distribution and linearly transforming it to the desired distribution [1].

Listing 6. Generate data from GP

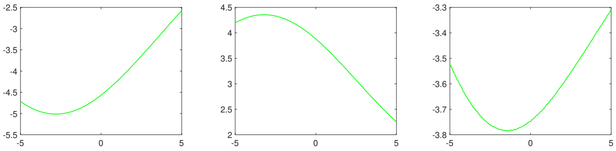
```
x = linspace(-5,5,200)';

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);
```

The Cholesky decomposition can only be applied to a positive-definite matrix. Since the covariance matrix $\boldsymbol{K}$ is semi-positive-definite ($\lambda \geq 0$), adding the small positive diagonal matrix 1e-6$\boldsymbol{I}$ guarantees that the matrix is full rank and positive-definite ($\lambda > 0$).



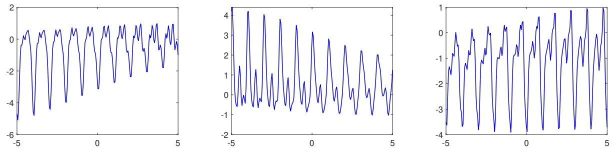
(a) Seed=1 (b) Seed=2 (c) Seed=3

Figure 6. Periodic Cov function



(a) Seed=1 (b) Seed=2 (c) Seed=3

Figure 7. Squared-Exp Cov function



(a) Seed=1 (b) Seed=2 (c) Seed=3

Figure 8. Product of Periodic and SE Cov function

By examining the plots (using multiple seeds for gpml_randn) in Fig 6, Fig 7 and Fig 8 for the different covariances, the periodic covariance k_{PER} directly provides the period of k_{prod} whereas k_{SE} only affects the amplitude. This can be interpreted as an amplitude function multiplied by a periodic function as in Equation 5 giving a covariance which has varying vertical and constant horizontal displacement between cycles. The product of two covariance functions can be interpreted as a logical AND operation [2] explaining the combined effects of both covariance functions.

5 Part E

We compare two GP models $\mathcal{M}1$ using the k_{SEard} covariance defined in Equation (6) and $\mathcal{M}2$ using the k_{SumSEard} covariance defined in Equation (7) [3, 4] trained on the dataset in Figure 9.

$$k_{\text{SEard}}(\boldsymbol{x},\boldsymbol{x}') = \sigma_f^2 \exp\Big(-\sum_{d=1}^{D=2}\frac{(x_d-x'_d)^2}{2\lambda_d^2}\Big) \tag{6}$$

$$k_{\text{SumSEard}}(\boldsymbol{x},\boldsymbol{x}') = \sigma_f^2 \exp\Big(-\sum_{d=1}^{D=2}\frac{(x_d-x'_d)^2}{2\lambda_d^2}\Big) + \sigma_f'^2 \exp\Big(-\sum_{d=1}^{D=2}\frac{(x_d-x'_d)^2}{2\lambda_d'^2}\Big) \tag{7}$$

Listing 7. Training dataset visualisation

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

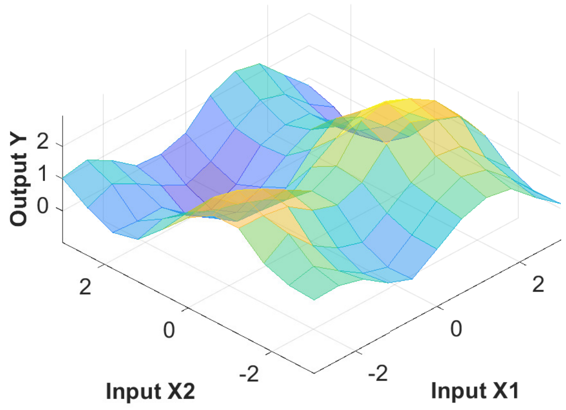


Figure 9. Training dataset 'cw1e.mat' visualisation

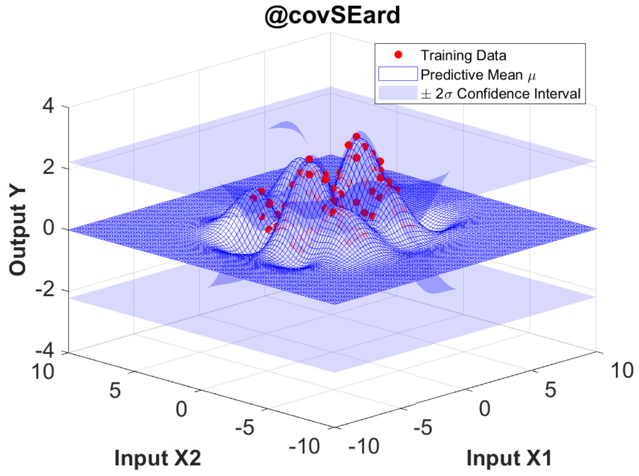


Figure 10. $\mathcal{M}1$ predictive distribution

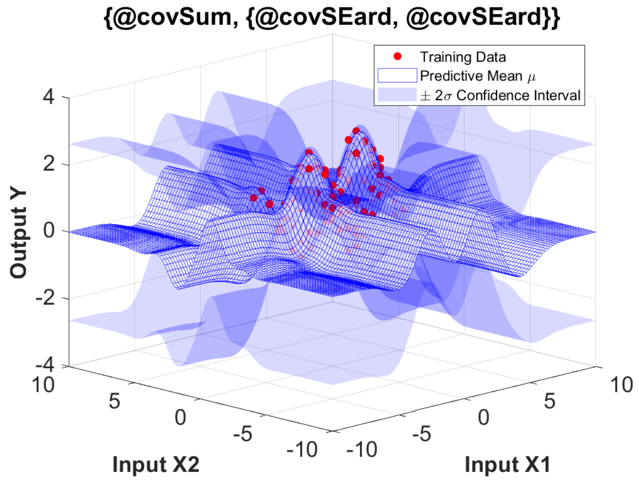


Figure 11. $\mathcal{M}2$ predictive distribution

Listing 8. GP models comparison

```
meanfunc = @meanZero;
likfunc = @likGauss;

% Model 1
covfunc = @covSEard;
hyp = struct('mean', [], 'cov', ...
        0.1*randn(3,1), 'lik', 0);

% Model 2
covfunc = {@covSum, {@covSEard, @covSEard}};
hyp = struct('mean', [], 'cov', ...
        0.1*randn(6,1), 'lik', 0);

hyp2 = minimize(hyp, @gp, -200, @infGaussLik, ...
        meanfunc, covfunc, likfunc, x, y);
[mu s2] = gp(hyp2, @infGaussLik, [], covfunc, ...
        likfunc, x, y, X);

scat = scatter3(x(:,1),x(:,2),y);
hold on; predmean = ...
        mesh(X1,X2,reshape(mu,size(X1)));
hold on; predusig = ...
        mesh(X1,X2,reshape(mu+2*sqrt(s2),size(X1)));
hold on; predlsig = ...
        mesh(X1,X2,reshape(mu-2*sqrt(s2),size(X1)));
```

Model	λ_1	λ_2	σ_f	λ_1'	λ_2'	σ_f'	σ_n	$p(\boldsymbol{y}\|\boldsymbol{X})$
$\mathcal{M}1$	1.511	1.286	1.107	-	-	-	0.1026	2.221e+8
$\mathcal{M}2$	1.446	2640	1.108	1730	0.986	0.711	0.0978	6.902e+28

Table 4. Hyperparameters and marginal likelihood

The negative log marginal likelihood $\mathcal{L}(\theta)$ is the summation of 2 terms (data-fit term d_f and complexity penalty c_p) and a constant:

$$
\begin{aligned}
d_f &= -\frac{1}{2}\boldsymbol{y}^\top[\boldsymbol{K}+\sigma_n^2\boldsymbol{I}]^{-1}\boldsymbol{y} \\
c_p &= \frac{1}{2}\log|\boldsymbol{K}+\sigma_n^2\boldsymbol{I}| \\
\mathcal{L}(\theta) &= -d_f + c_p + const.
\end{aligned} \tag{8}
$$

Table 4 indicates that $\mathcal{M}2$ is a better model as it achieves a larger marginal likelihood. Both models have identical data-fit terms (identical for any stationary covariance), which means that $\mathcal{M}2$ achieves a higher marginal likelihood (lower $\mathcal{L}(\theta)$) by having a lower complexity.

The addition of two kernels can be interpreted as a logical OR operation [2] allowing $\mathcal{M}2$ to achieve a simpler fit by determining a suitable distribution from a combination of two independent kernels.

Both models fit very well in the training region as shown by the datapoints on the predictive mean surfaces in Figure 10 and 11 however for $\mathcal{M}1$ the surface remains flat while in $\mathcal{M}2$ there are variations that extend outside the training dataset. The variation in the $\pm 2\sigma$ confidence intervals in $\mathcal{M}2$ outside the training region helps to improve the likelihood since these provide reasonable uncertainty estimates unlike $\mathcal{M}1$. There is greater flexibility when summing two covariances rather than using a single function since the two covariance functions can independently fit along separate axes. Overall this suggests that $\mathcal{M}2$ provides a better fitting that $\mathcal{M}1$.

References

[1] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*, 2006.

[2] D. Duvenaud. (2013) Kernel Cookbook. [Online]. Available: https://www.cs.toronto.edu/~duvenaud/cookbook/

[3] ——. (2013) Kernel Design. [Online]. Available: https://raw.githubusercontent.com/duvenaud/phd-thesis/master/kernels.pdf

[4] C. E. Rasmussen and C. K. I. Williams. (2018) GPML Documentation. [Online]. Available: http://www.gaussianprocess.org/gpml/code/matlab/doc/