CUED - Engineering Tripos Part IIB 2024-2025 **Module Coursework**

Module	4M24	Title of report	High Dimensional MCMC
Date submitted: 31/1/25		Assessment for this module is ☐ 100% / ☑ 25% coursework of which this assignment forms 100 %	

UNDERGRADUATE and POST GRADUATE STUDENTS		
Candidate number:	5488A	☑ Undergraduate ☐ Post graduate

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:			

Marker: Date:

4M24 CW - High-Dimensional MCMC

Candidate Number: 5488A

December 2024

1 Simulation

a Gaussian Process Prior

Our prior is a Gaussian Process with zero mean and a squared exponential covariance kernel, $k(\boldsymbol{x}, \boldsymbol{x}')$, having length scale ℓ. The coordinates, $\{\boldsymbol{x}_n\}_{n=1}^N$, of our samples are placed on a regular $D \times D$ grid in $[0,1]^2$. It is clear that $N = D^2$.

$$k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\ell^2}\right) \tag{1}$$

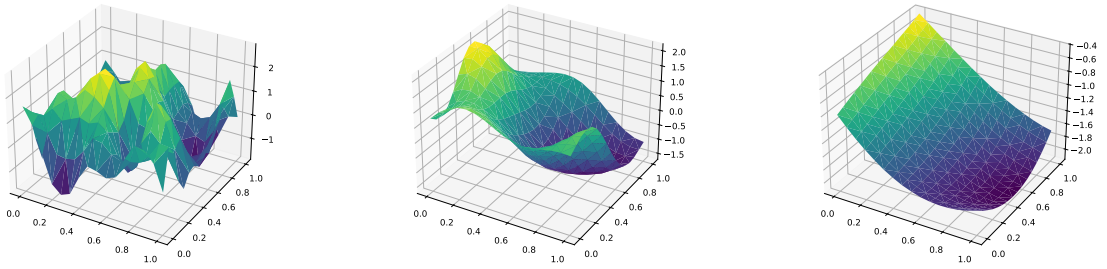
Our samples, $\boldsymbol{u}$, collected into an $N \times 1$ vector are therefore distributed $\boldsymbol{u} \sim \mathcal{N}(\boldsymbol{0}, C)$, where C is the $N \times N$ covariance matrix with entries $C_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Samples from this prior are shown in Figure 1 for 3 values of ℓ. Larger values result in a smoother surface with more correlation between nearby points.

We subsample the grid with M uniform random draws and apply independent Gaussian measurement noise, $\boldsymbol{\epsilon}$, to the observations. This subsampling can be captured by the matrix $M \times N$ matrix G with entries $G_{ij} = 1$ if the ith observation is at the jth grid point and 0 otherwise. We will denote the $M \times 1$ vector of the subsampled latent field as $\tilde{\boldsymbol{u}} = G\boldsymbol{u}$. We define the subsampling factor $f := N/M$. The observations, $\boldsymbol{v}$, are produced according to the following model.

$$\boldsymbol{v} = G\boldsymbol{u} + \boldsymbol{\epsilon} \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, I) \tag{2}$$

One sample is produced from this model with $D = 16$, $f = 4$ and $\ell = 0.3$ to be used as our dataset for the analysis within this section. Figure 2 shows the latent surface, $\boldsymbol{u}$, and $M = \frac{N}{f} = 64$ noisy observations, $\boldsymbol{v}$.



(a) $\ell = 0.1$ (b) $\ell = 0.3$ (c) $\ell = 1$

Figure 1: Samples from the Gaussian Process Prior

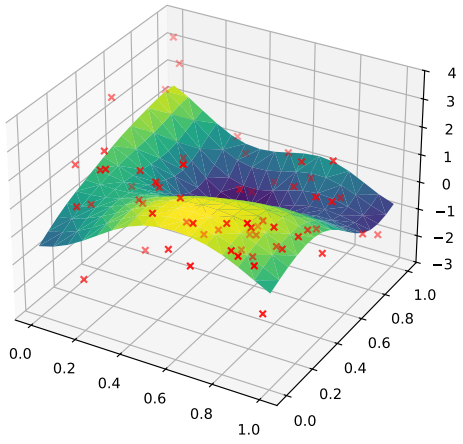


Figure 2: Simulated dataset: $\boldsymbol{v}$ - red crosses, $\boldsymbol{u}$ - surface

b Likelihoods and MCMC

We now proceed to infer the latent surface, $\boldsymbol{u}$, from the noisy observations, $\boldsymbol{v}$, using MCMC. To compute our posterior we need to evaluate the likelihood, $p(\boldsymbol{v}|\boldsymbol{u})$, and the prior, $p(\boldsymbol{u})$. The form of the prior was given previously but is repeated below and its logarithm can be computed with simple algebraic manipulation.

$$\begin{aligned} \boldsymbol{u} &\sim \mathcal{N}(\boldsymbol{0}, C) \\ \ln p(\boldsymbol{u}) &= -\frac{N}{2}\ln(2\pi) - \frac{1}{2}\ln(|K|) - \frac{1}{2}\boldsymbol{u}^T C^{-1}\boldsymbol{u} \\ &= -\frac{1}{2}\boldsymbol{u}^T C^{-1}\boldsymbol{u} + \text{const} \end{aligned} \tag{3}$$

Likewise the likelihood is given below.

$$\begin{aligned} \boldsymbol{v}|\boldsymbol{u} &\sim \mathcal{N}(G\boldsymbol{u}, I) \\ \ln p(\boldsymbol{v}|\boldsymbol{u}) &= -\frac{M}{2}\ln(2\pi) - \frac{1}{2}\ln(|I|) - \frac{1}{2}(\boldsymbol{v}-\tilde{\boldsymbol{u}})^T(\boldsymbol{v}-\tilde{\boldsymbol{u}}) \\ &= -\frac{1}{2}(\boldsymbol{v}-\tilde{\boldsymbol{u}})^T(\boldsymbol{v}-\tilde{\boldsymbol{u}}) + \text{const} \end{aligned} \tag{4}$$

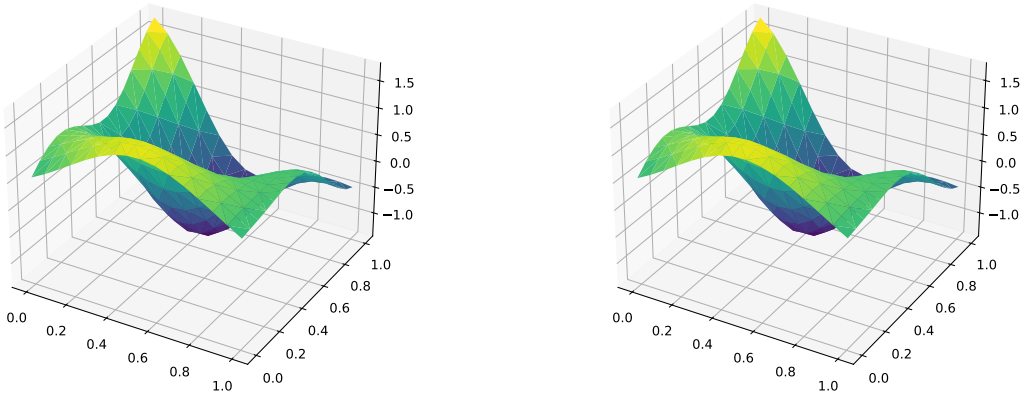
Computation of the posterior is straightforward using Bayes rule. Note that we only need to compute the log-prior and log-likelihood up to a constant which greatly saves on computation.

$$p(\boldsymbol{u}|\boldsymbol{v}) \propto p(\boldsymbol{v}|\boldsymbol{u})p(\boldsymbol{u}) \therefore \ln p(\boldsymbol{u}|\boldsymbol{v}) = \ln p(\boldsymbol{v}|\boldsymbol{u}) + \ln p(\boldsymbol{u}) + \text{const} \tag{5}$$

We now consider two MCMC algorithms for generating samples from the posterior.

b.1 Gaussian random walk Metropolis-Hastings

The Gaussian random walk Metropolis-Hastings (GRW-MH) algorithm samples from a Gaussian random walk proposal distribution, $\boldsymbol{X}'|\boldsymbol{X} \sim \mathcal{N}(\boldsymbol{X}, \beta^2 C)$. The acceptance probability for our target distribution, $\pi(\boldsymbol{u}) = p_{\boldsymbol{u}|\boldsymbol{v}}(\boldsymbol{u}|\boldsymbol{v})$, simplifies nicely as our proposal distribution is symmetric, $p(\boldsymbol{x}'|\boldsymbol{x}) = p(\boldsymbol{x}|\boldsymbol{x}')$. The reason we only need to compute our log-posterior up to a constant is now clear as we are calculating a



(a) GRW-MH (b) PCN

Figure 3: Mean of the posterior latent surface, $\mathbb{E}(\boldsymbol{u}|\boldsymbol{v})$, for GRW-MH and PCN

difference between two log-posteriors.

$$\begin{aligned}\alpha(\boldsymbol{x}, \boldsymbol{x}') &= \min\left(\frac{\pi(\boldsymbol{x}')p_{\boldsymbol{X}'|\boldsymbol{X}}(\boldsymbol{x}|\boldsymbol{x}')}{\pi(\boldsymbol{x})p_{\boldsymbol{X}'|\boldsymbol{X}}(\boldsymbol{x}'|\boldsymbol{x})}, 1\right) \\ &= \min\left(\frac{p_{\boldsymbol{u}|\boldsymbol{v}}(\boldsymbol{x}'|\boldsymbol{v})}{p_{\boldsymbol{u}|\boldsymbol{v}}(\boldsymbol{x}|\boldsymbol{v})}, 1\right) \\ \ln(\alpha(\boldsymbol{x}, \boldsymbol{x}')) &= \min(\ln p_{\boldsymbol{u}|\boldsymbol{v}}(\boldsymbol{x}'|\boldsymbol{v}) - \ln p_{\boldsymbol{u}|\boldsymbol{v}}(\boldsymbol{x}|\boldsymbol{v}), 0)\end{aligned} \tag{6}$$

b.2 Preconditioned Crank-Nicolson

The Preconditioned Crank-Nicolson (PCN) algorithm produces a Markov chain with invariant measure π where $\frac{d\pi}{d\mu^0}(\boldsymbol{x}) \propto \exp(-\Phi(\boldsymbol{x}))$, where $\mu^0 = \mathcal{N}(0, C_0)$ is a Gaussian measure. In our case we have $d\pi(\boldsymbol{x}) = p_{\boldsymbol{u}|\boldsymbol{v}}(\boldsymbol{x}|\boldsymbol{v}) \propto p_{\boldsymbol{v}|\boldsymbol{u}}(\boldsymbol{v}|\boldsymbol{x})p_{\boldsymbol{u}}(\boldsymbol{x})$, therefore $\Phi(\boldsymbol{x}) = -\ln p_{\boldsymbol{v}|\boldsymbol{u}}(\boldsymbol{v}|\boldsymbol{x})$ and $d\mu^0 = p_{\boldsymbol{u}}$ ($C_0 = C$). The proposal distribution is $\boldsymbol{X}'|\boldsymbol{X} \sim \mathcal{N}(\sqrt{1-\beta^2}\boldsymbol{X}, \beta^2 C)$ and the acceptance probability is

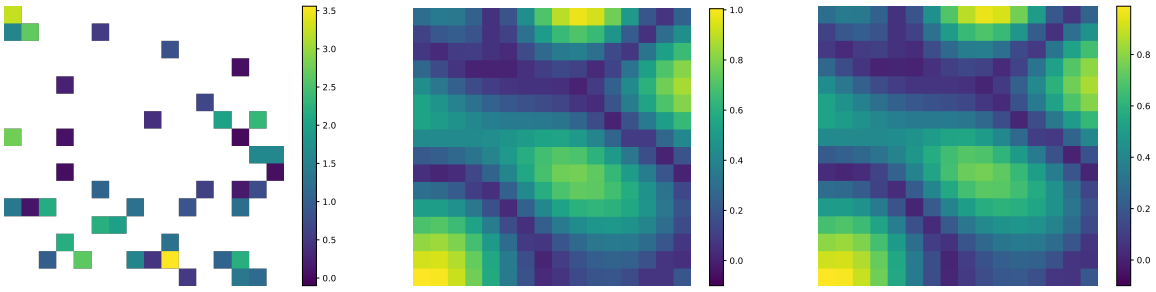
$$\begin{aligned}\alpha(\boldsymbol{x}, \boldsymbol{x}') &= \min\left(\exp(\Phi(\boldsymbol{x}) - \Phi(\boldsymbol{x}')), 1\right) \\ &= \min\left(\exp(\ln p_{\boldsymbol{v}|\boldsymbol{u}}(\boldsymbol{v}|\boldsymbol{x}') - \ln p_{\boldsymbol{v}|\boldsymbol{u}}(\boldsymbol{v}|\boldsymbol{x})), 1\right) \\ \ln(\alpha(\boldsymbol{x}, \boldsymbol{x}')) &= \min(\ln p_{\boldsymbol{v}|\boldsymbol{u}}(\boldsymbol{x}'|\boldsymbol{v}) - \ln p_{\boldsymbol{v}|\boldsymbol{u}}(\boldsymbol{x}|\boldsymbol{v}), 0)\end{aligned} \tag{7}$$

Again we only need to compute the log likelihoods up to a constant as we are calculating a difference.

b.3 Comparison

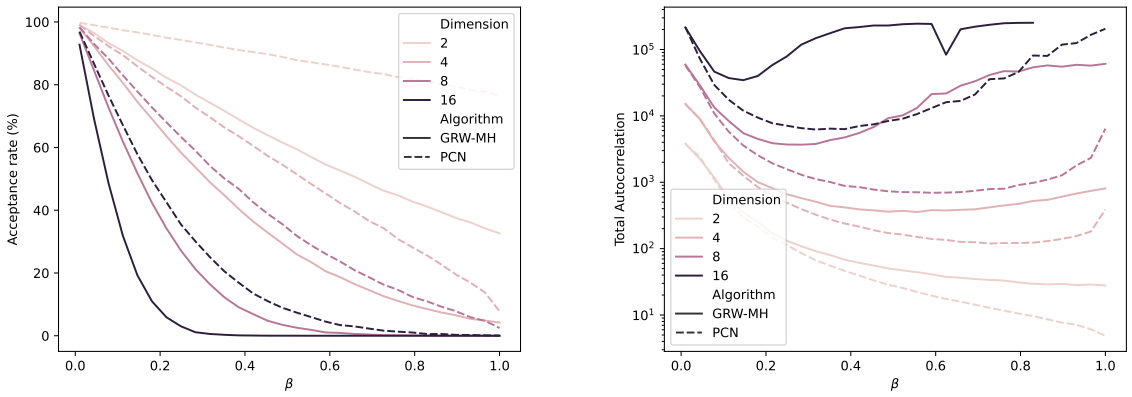
Figure 3 shows the mean of the posterior latent surface, $\boldsymbol{u}|\boldsymbol{v}$, for both the GRW-MH and PCN algorithms. Both algorithms converge to the same posterior as expected. The absolute error to the true latent surface, shown in Figure 4, is not zero but this is expected as the noise in the observations will prevent perfect recovery of the latent surface. Also note that the error is larger in regions where there are no observations because the prior is dominates the posterior in these regions.

The acceptance rate of the GRW-MH and PCN algorithms is shown in Figure 5a for varying β and dimension, D. For both algorithms the acceptance rate approaches 100% as $\beta \to 0$ and decreases with increasing β. This is expected as for $\beta = 0$ we have $\boldsymbol{X}' = \boldsymbol{X}$, the proposal is the same as the current state and therefore the acceptance probability is 1 for both algorithms. However, in this case the proposal is not exploring the space and the algorithm will not converge to the posterior. For increasing β the proposal



(a) Observed data (b) GRW-MH (c) PCN

Figure 4: Observed data (a). Error of the mean posterior latent surface, $\mathbb{E}(\boldsymbol{u}|\boldsymbol{v})$, to the true $\boldsymbol{u}$ for GRW-MH and PCN (b, c)



(a) Acceptance rate (b) Total autocorrelation

Figure 5: Effect of β and D on the acceptance rate and total autocorrelation for GRW-MH and PCN

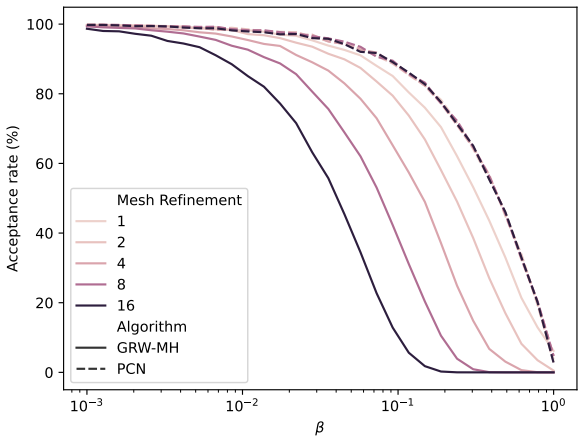


Figure 6: Effect of mesh refinement on acceptance rate for GRW-MH and PCN.

distribution can take larger steps and therefore the acceptance rate decreases. With increasing dimension the acceptance rate decreases for both algorithms. This is explained by the fact that higher dimensional spaces have more degrees of freedom and therefore the proposal distribution is less likely to step into a higher probability of the posterior.

If we wish to choose an optimal β we could choose the value that minimises the variance of Monte Carlo expectations. If we wish to compute $\mathbb{E}[f(\boldsymbol{X})]$, for some random variable $\boldsymbol{X} \sim \pi$, we can compute a Monte Carlo estimate, $\bar{f} = \frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{X}^{(t)})$, using samples, $X^{(t)}$, from a Markov chain with invariant measure π. The variance of this estimate in terms of the autocorrelation coefficient of the chain, $\rho_{f(X)f(X)}(\tau)$, is:

$$\mathrm{Var}(\bar{f}) = \mathrm{Var}(\frac{1}{T}\sum_{t=1}^{T} f(\boldsymbol{X}^{(t)})) = \frac{1}{T}\left(\mathrm{Var}(f(\boldsymbol{X})) + 2\sum_{\tau=1}^{T-1}(1-\frac{\tau}{T})\rho_{f(X)f(X)}(\tau)\right) \tag{8}$$

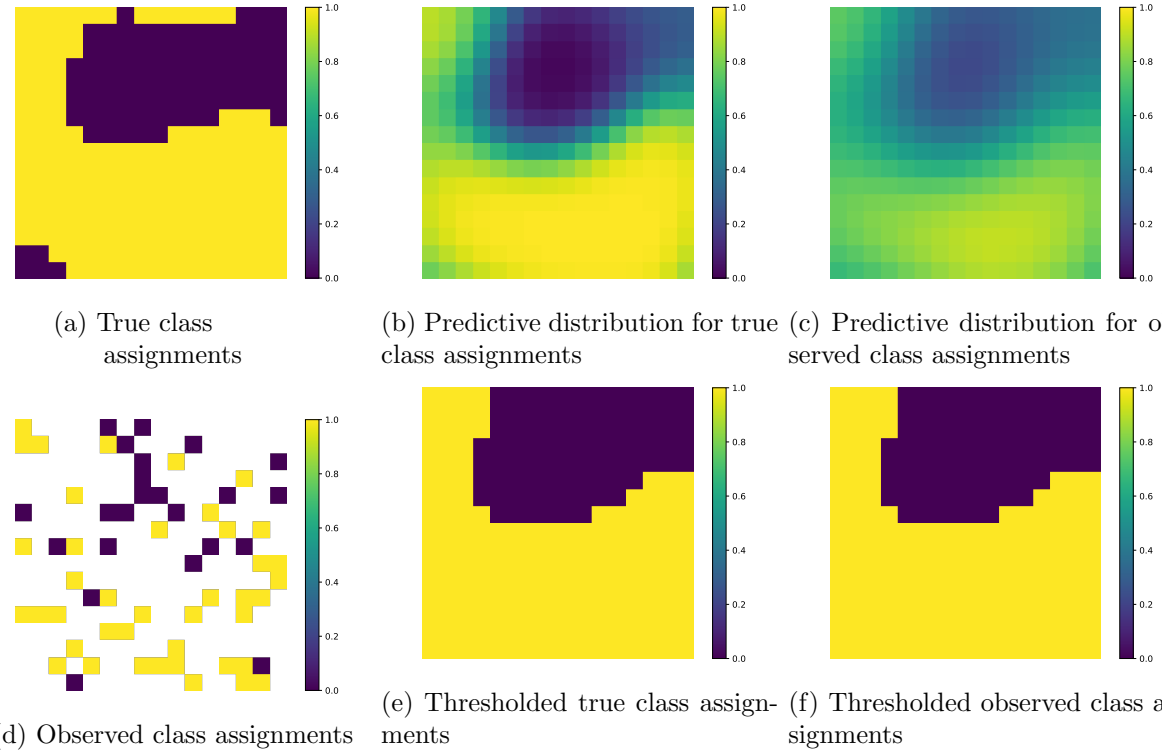
Therefore we should choose β that minimises $\sum_{\tau=1}^{T-1}(1-\frac{\tau}{T})\rho_{f(X)f(X)}(\tau)$, which we will name the total autocorrelation.

Figure 5b shows how the total auto correlation for $\mathbb{E}_{\boldsymbol{u}|\boldsymbol{v}}[\boldsymbol{u}]$ varies with β and D. In general for fixed β and D GRW-MH has higher total autocorrelation than PCN, this indicates the PCN explores the posterior more efficiently leading to a producing a Monte Carlo estimate with lower variance. Also note that the optimal β decreases with increasing dimension for both algorithms. However, based on a rough observation from the Figure 5a we can see that the optimal (minimum total autocorrelation) β maintains an acceptance rate of around 20% for both cases. This is a good rule of thumb, with theoretical backing [1], for choosing β in practice. We will use this empirical rule to choose β for the remainder of the report.

Finally, we observe that the time per iteration for PCN is 10% lower than GRW-MH. This is because the PCN algorithm only requires evaluation of the likelihood while the GRW-MH algorithm requires evaluation of the likelihood and prior. Therefore the PCN algorithm is superior in convergence and computational efficiency.

b.4 Mesh Refinement

We now analyse the robustness of the algorithms to mesh refinement. While keeping the same number of observations we increase the number of mesh points, in the limit we will be drawing samples in the infinite dimensional Hilbert space. Figure 6 shows that the acceptance rate of the GRW-MH algorithm approaches 0 with increased mesh refinement, conversely the acceptance rate of the PCN algorithm is unaffected. This is because the acceptance probability of the GRW-MH algorithm is not well defined in an infinite dimensional space.



(a) True class assignments

(b) Predictive distribution for true class assignments

(c) Predictive distribution for observed class assignments

(d) Observed class assignments

(e) Thresholded true class assignments

(f) Thresholded observed class assignments

Figure 7: Probit Observations

c Probit Observations

The observation model is now augmented with a probit function that assigns 1 to positive v_n and 0 otherwise. The likelihood is therefore

$$p(\boldsymbol{t}|\boldsymbol{u}) = \prod_{m=1}^{M} p(t_m|\tilde{u}_m) = \prod_{m=1}^{M} \Phi(\tilde{u}_m)^{t_m} \Phi(-\tilde{u}_m)^{1-t_m} \tag{9}$$

The predictive distribution for the true class assignments, $\boldsymbol{t}_{true} = \text{probit}(\boldsymbol{u})$, can be computed as follows.

$$p(t^*_{n|true} = 1|\boldsymbol{t}) = \int p(t^*_{n|true} = 1|\boldsymbol{u}) p(\boldsymbol{u}|\boldsymbol{t}) d\boldsymbol{u} = \frac{1}{T} \sum_{t=1}^{T} p(t^*_{n|true} = 1|\boldsymbol{u}^{(t)}) = \frac{1}{T} \sum_{t=1}^{T} \text{probit}(u_n^{(t)}) \tag{10}$$

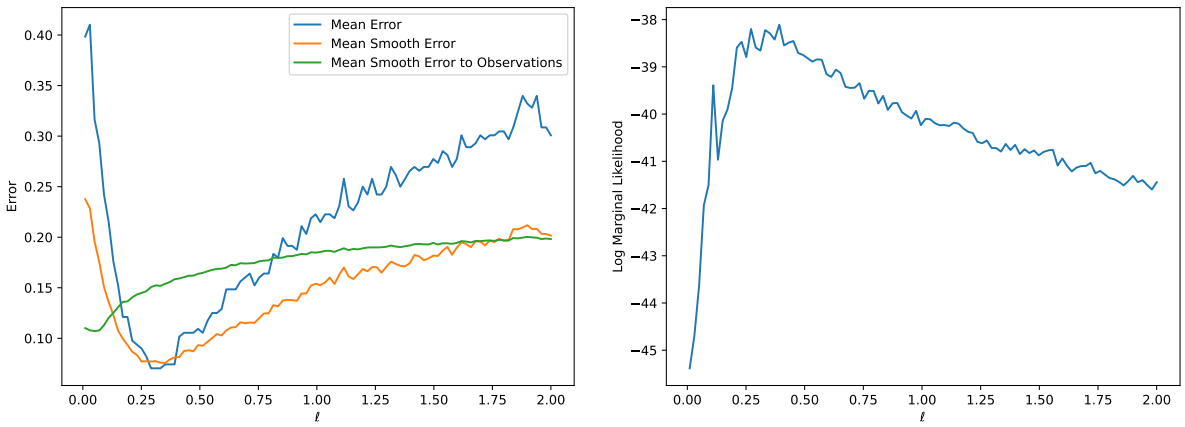
Similarly, the predictive distribution for the observed class assignments is

$$p(t^*_n = 1|\boldsymbol{t}) = \frac{1}{T} \sum_{t=1}^{T} p(t^*_n = 1|\boldsymbol{u}^{(t)}) = \frac{1}{T} \sum_{t=1}^{T} \Phi(u_n^{(t)}) \tag{11}$$

The true class assignments along with the two predictive distributions are shown in Figure 7. The predictive distributions broadly follow the shape of the true classifications. Notice that the predictive distribution for the observed class assignments is less confident than that of the true class assignments due to the noise in the observations.

d Hyperparameter Estimation

Hard assignments for the true probit classifications are produced by thresholding the predictive probabilities at 0.5 (or rounding). These are also visualised in Figure 7. Notice that the thresholded true



(a) Prediction error with varying ℓ.

(b) Log marginal likelihood with varying ℓ.

Figure 8: Hyperparameter estimation

class assignments are the same as the thresholded observed class assignments as the observation noise is symmetric. We can compute the mean prediction error, $e(\ell)$, by comparing our predictions to the true latent surface as follows.

$$e(\ell) = \frac{1}{N}\sum_{n=1}^{N}(\text{round}(p(t^*_{n|true} = 1|\boldsymbol{t}, \ell)) - t_{n|true})^2 \tag{12}$$

This error is not smooth as when the predictive probability crosses the threshold the class assignment changes. We can compute a smooth error, $e_{smooth}(\ell)$, as the mean squared error over the predictive probabilities.

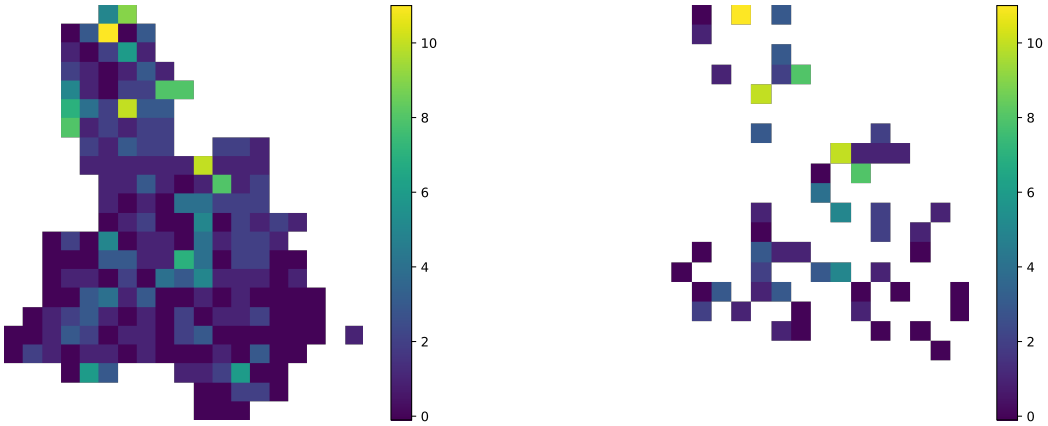
$$e_{smooth}(\ell) = \frac{1}{N}\sum_{n=1}^{N}(p(t^*_{n|true} = 1|\boldsymbol{t}, \ell) - t_{n|true})^2 \tag{13}$$

The optimal length scale, $\ell^\star$, is chosen to minimise the error. The error for varying length scale, ℓ, is shown in Figure 8a. Both formulations of the error show a minimum at $\ell^\star \approx 0.3$ which is the true length scale used to generate the data. However in a real world scenario we can only view the true latent surface through the subsampled and noisy observation model. Also shown in Figure 8a is the mean prediction error to the our observations. This has a minimum at $\ell^\star \approx 0.05$, well below the true length scale, as we are overfitting to the noise in the observations and subsampling.

An alternative approach to hyperparameter estimation is to maximise the marginal likelihood, $p(\boldsymbol{v}|\ell)$. We can compute a simple Monte Carlo estimate of the log marginal likelihood as follows.

$$p(\boldsymbol{t}|\ell) = \int p(\boldsymbol{t}|\boldsymbol{u}, \ell)p(\boldsymbol{u}|\ell)d\boldsymbol{u} = \frac{1}{T}\sum_{t=1}^{T} p(\boldsymbol{t}|\boldsymbol{u}^{(t)}) \quad \boldsymbol{u}^{(t)} \sim p(\cdot|\ell) \tag{14}$$

This estimate has high variance for small ℓ which could be reduced using various variance reduction techniques, however, we will not consider these here and will instead use a large number of samples. Notice that the computation of the marginal likelihood only requires knowledge of the observations and not the true latent variable. The log marginal likelihood for varying ℓ is shown in Figure 8b. The maximum of the log marginal likelihood is at $\ell^\star \approx 0.3$ which is the true length scale used to generate the data.



(a) Bike thefts in Lewisham

(b) Subsampled bike theft data

Figure 9

2 Spatial Data

We now turn to the real world problem of predicting bike thefts in Lewisham borough over 2015. The area of the borough is split up into N 400m^2 cells identified by coordinates $\{(x_n, y_n)\}_{n=1}^N$. The provided coordinates were normalised between 0 and 1, therefore by measuring the width of Lewisham borough as 8km we rescaled the coordinates to indicate the true relative distance between cells. The total bike thefts reported during the year in the nth cell is c_n, we collect these into an $N \times 1$ vector, $\boldsymbol{c}$. The data is subsampled to investigate if we can still make predictions with partial observations. This is done as before with $M = \frac{N}{f}$ random draws to form a matrix G that performs the transformation. We will use a subsampling factor of $f = 4$ for the analysis in this section. The full and subsampled data are visualised in Figure 9.

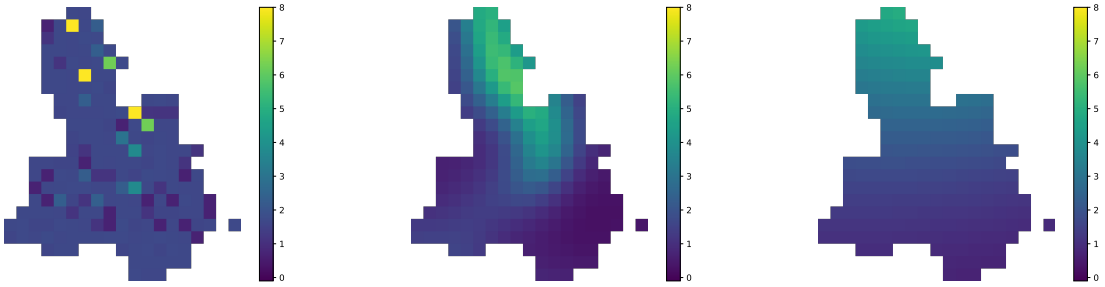
The data is modelled using a latent surface, $\boldsymbol{u}$, with a squared exponential Gaussian Process prior as before. This field is mapped to $\mathbb{R}^+$ using an exponential function to model the rate of bike thefts, $\boldsymbol{\theta} = \exp \boldsymbol{u}$, where the exponential function is applied elementwise. The observations, $\boldsymbol{c_n}$, are modelled as Poisson random variables with rate θ_n. As before we will denote the subsampled $M \times 1$ vectors with a tilde, i.e. $\tilde{\boldsymbol{u}} = G\boldsymbol{u}$, $\tilde{\boldsymbol{\theta}} = G\boldsymbol{\theta}$ and $\tilde{\boldsymbol{c}} = G\boldsymbol{c}$.

e Poisson observations

To produce samples of the posterior with the PCN sampler we need to compute the likelihood of the observations, $p(\tilde{\boldsymbol{c}}|\boldsymbol{u})$.

$$\begin{aligned} c_m|\boldsymbol{\theta} &\sim \text{Poisson}(\tilde{\theta}_m) \\ \ln p(\tilde{\boldsymbol{c}}|\boldsymbol{u}) &= \ln p(\tilde{\boldsymbol{c}}|\boldsymbol{u}, \boldsymbol{\theta}) = \ln p(\tilde{\boldsymbol{c}}|\boldsymbol{\theta} = \exp(\boldsymbol{u})) \\ &= \sum_{m=1}^{M} \left(-\exp(\tilde{u}_m) + \tilde{c}_m \tilde{u}_m - \ln(\tilde{c}_m!)\right) \\ &= -\mathbf{1}^T \exp(\tilde{\boldsymbol{u}}) + \tilde{\boldsymbol{c}}^T \tilde{\boldsymbol{u}} + f(\tilde{\boldsymbol{c}}) \end{aligned} \tag{15}$$

Where we define exp elementwise and $f(\boldsymbol{c})$ is a constant that does not depend on $\boldsymbol{u}$ and is therefore cancelled out when computing the PCN acceptance probability.



(a) $\ell = 0.1$km (b) $\ell = 1.5$km (c) $\ell = 10$km

Figure 10: Posterior mean theft field, $\mathbb{E}_{c_n^*|\tilde{\mathbf{c}}}(c_n^*)$, for varying ℓ.

f Bike theft predictions

f.1 Posterior field

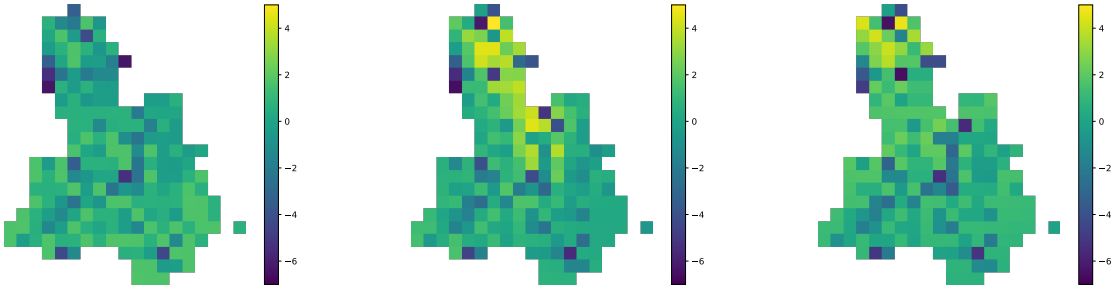
We can compute the posterior expected number of bike thefts, $\mathbb{E}_{c_n^*|\tilde{\mathbf{c}}}(c_n^*)$, at a location (x_n, y_n) as follows.

$$\begin{aligned}
\mathbb{E}_{c_n^*|\tilde{\mathbf{c}}}(c_n^*) &= \mathbb{E}_{\boldsymbol{\theta}|\tilde{\mathbf{c}}}(\mathbb{E}_{c_n^*|\boldsymbol{\theta},\tilde{\mathbf{c}}}(c_n^*)) \\
&= \mathbb{E}_{\boldsymbol{\theta}|\tilde{\mathbf{c}}}(\mathbb{E}_{c_n^*|\boldsymbol{\theta}}(c_n^*)) \\
&= \mathbb{E}_{\boldsymbol{\theta}|\tilde{\mathbf{c}}}(\theta_n) \\
&\approx \frac{1}{T}\sum_{t=1}^{T} \theta_n^{(t)} \\
&= \frac{1}{T}\sum_{t=1}^{T} \exp(u_n^{(t)})
\end{aligned} \tag{16}$$

Here we use the law of conditional expectation to condition the expectation on the rate field, $\boldsymbol{\theta}$, and notice that the expectation of a Poisson random variable is equal to its rate. We can then compute a Monte Carlo estimate for the posterior mean theft field using the samples from the PCN sampler. We have computed the this field and its error to the observations for various ℓ in Figures 10 and 11 respectively. For small values $\ell = 100m$ (Figure 10a) the model only fits at the coordinates of the subsampled data while returning to the prior theft rate of 1 elsewhere because neighbouring cells are effectively independent. This is not useful for making predictions outside of the subsampled data. For large values $\ell = 10km$ (Figure 10a) the model only captures the large scale trend of the data, in this case only capturing the North-South variation in the theft rate. A more modest value $\ell = 1.5km$ (Figure 10b) manages to capture the local variations in theft rate while still being able to make predictions outside of the observations.

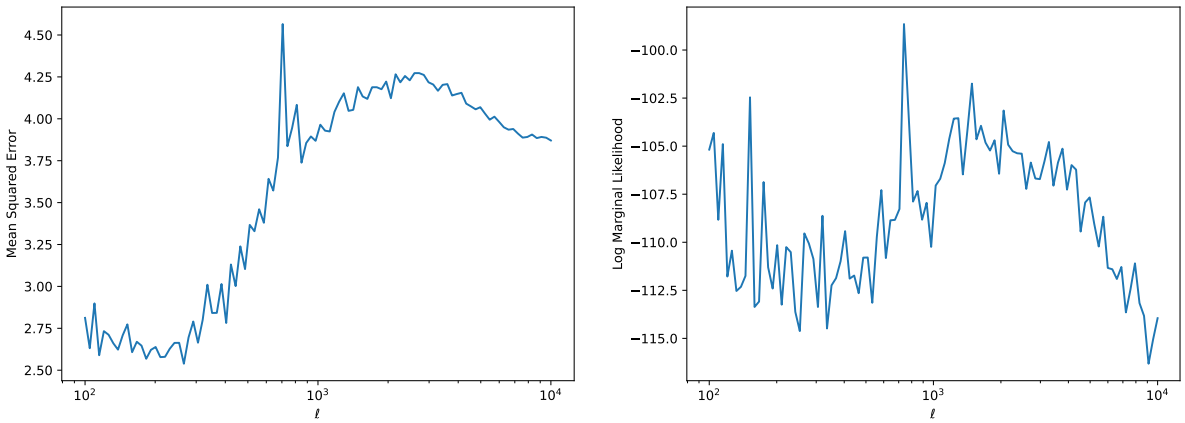
f.2 Hyperparameter estimation

Using the same methods as in Section d we can estimate the optimal length scale, $\ell^\star$, for the bike theft data. The error and log marginal likelihood for varying ℓ are shown in Figure 12a and Figure 12b respectively. For the mean squared error we see a minimum at $\ell^\star \approx 0.4$km while with the log marginal likelihood we see a maximum at $\ell^\star \approx 1.5$km (if you read through the noise). This difference is likely from a combination of two issues, we do not have access to the true latent field and mean squared error is not an accurate measure of the likelihood of a Poisson observation model. Minimising the mean squared error when observations are Gaussian is accurate as this is equivalent to maximising the likelihood. However, this is not the case for Poisson observations.



(a) $\ell = 0.1$km (b) $\ell = 1.5$km (c) $\ell = 10$km

Figure 11: Error field for varying ℓ.



(a) Mean squared error. (b) Log marginal likelihood.

Figure 12: Hyperparameter, ℓ, estimation for the spatial data.

One could interpret ℓ as the radius of the zone of operation of a typical bike thief. $\ell^\star \approx 1.5$km is sensible in this context. Figure 10b visualises the inferred field for this ℓ, the model captures the local variations in theft rate without overfitting to individual measurements. The estimation of ℓ could be further improved by taking a fully Bayesian approach with a prior on ℓ to compute its posterior.

References

[1] C. Sherlock and G. Roberts. Optimal scaling of the random walk metropolis on elliptically symmetric unimodal targets. *Bernoulli*, 15(3):774–798, 2009.