4M24 Coursework

5517B

January 2022

1 Simulation

1.1 Question a

A latent variable vector $\mathbf{u} \in \mathbb{R}^N, N = D^2$ is sampled from a Gaussian Process (GP) prior $\mathbf{u} \sim \mathcal{N}(\mathbf{0}, C)$, defined on input variables $\mathbf{x}^{(n)}$ such $x_1^{(n)}, x_2^{(n)}$ form a $D \times D$ regular grid in $[0, 1]$, and:

$$C_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(\frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\ell^2}\right) = C_{ji} \tag{1}$$

The data vector $\mathbf{v} \in \mathbb{R}^M$ is a noisy observation of a subset of the latent variables: $\mathbf{v} = G\mathbf{u} + \boldsymbol{\epsilon}$ $G \in \{0, 1\}^{M,N}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)$, and G is sparse such that each latent vector element is sampled once not at all. We expect that varying the length scale ℓ of the kernel changes the latent surface smoothness w.r.t. its coordinates: $\uparrow \ell \implies \uparrow \frac{-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\ell^2}$ $\forall\, i, j \implies \uparrow C_{ij}$ $\forall\, i, j$ - larger ℓ causes a smoother surface as any two pair of points have higher covariance. This is observed in Figure 1, where we visualise the latent variable surface for multiple values of ℓ, as well as the scattered data vector elements in the relevant coordinates.

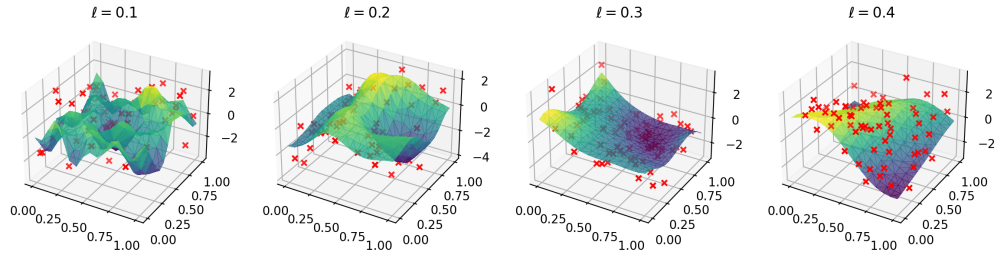

Figure 1: Latent variable surface - u_n against $x_1^{(n)}, x_2^{(n)}$ - with data vector elements - v_m. We use $D = 16, N/M = 4$.

As expected frim our formulation of $\mathbf{v}$, the points are randomly scatted in horizontal directions, and are distributed vertically with the $\mathbf{u}$ surface as their mean. Note here that the latent and observed data is randomly generated at each trial; future experiments in this part will be reproduced multiple times for reliable results.

1.2 Question b

Using the definitions in Section 1.1, we can establish the prior over latent variables:

$$p(\mathbf{u}) = \mathcal{N}(\mathbf{0}, C) \propto \exp\left(\frac{-1}{2}\langle \mathbf{u}, C^{-1}\mathbf{u}\rangle\right) \implies \begin{cases} \log p(\mathbf{u}) = -\frac{1}{2}\langle \mathbf{u}, C^{-1}\mathbf{u}\rangle + \text{Const}_{\mathbf{u}} \\ \\ \texttt{def log\_prior(u, K\_inverse):} \\ \quad \texttt{return -0.5 * (u.T @ K\_inverse @ u)} \end{cases}$$

and likelihood of the data given the latent variables:

$$p(\mathbf{v}|\mathbf{u}) = \mathcal{N}(G\mathbf{u}, I) \propto \exp\left(\frac{-1}{2}\langle(\mathbf{v}-G\mathbf{u}),(\mathbf{v}-G\mathbf{u})\rangle\right) \implies \begin{cases} \log p(\mathbf{v}|\mathbf{u}) = -\frac{1}{2}\langle(\mathbf{v}-G\mathbf{u}),(\mathbf{v}-G\mathbf{u})\rangle \ + \text{Const}_{\mathbf{u},\mathbf{v}} \\ \\ \texttt{def log\_continuous\_likelihood(u, v, G):} \\ \quad \texttt{dd = (G @ u - v)} \\ \quad \texttt{return -0.5 * (dd.T @ dd)} \end{cases}$$

Note here that our code implementations return log values up to a constant. When deriving the posterior using Bayes rule, these constants can be absorbed into the data evidence $p(\mathbf{v})$, i.e.: $p(\mathbf{u}|\mathbf{v}) \propto p(\mathbf{v}|\mathbf{u})p(\mathbf{u}) \implies \log p(\mathbf{u}|\mathbf{v}) = \log p(\mathbf{v}|\mathbf{u}) + \log p(\mathbf{u}) + \text{all constants}$.

Indeed, we use this proportional form for the Metropolis-Hastings (MH) and preconditioned Crank-Nicolson (pCN) algorithms, for sampling from the posterior $\pi(u) = p(\mathbf{u}|\mathbf{v})$. Both algorithms follow the MH framework of proposing a new sample using a proposal distribution: $\tilde{\mathbf{u}}' \sim q(\tilde{\mathbf{u}}'|\tilde{\mathbf{u}}^{(k)})$, which is accepted as the next sample in the chain with probabilities:

$$\tilde{\mathbf{u}}^{(k+1)} = \begin{cases} \tilde{\mathbf{u}}' & \text{w.p.} \quad \alpha \\ \tilde{\mathbf{u}}^{(k)} & \text{w.p.} \quad 1-\alpha \end{cases} \qquad \alpha = \min\left(\frac{\pi(\tilde{\mathbf{u}}')q(\tilde{\mathbf{u}}^{(k)}|\tilde{\mathbf{u}}')}{\pi(\tilde{\mathbf{u}}^{(k)})q(\tilde{\mathbf{u}}'|\tilde{\mathbf{u}}^{(k)})}, 1\right) \tag{2}$$

The proposal distributions used for each algorithm are:

$$\begin{aligned} q_{\text{MH}}(\tilde{\mathbf{u}}'|\tilde{\mathbf{u}}^{(k)}) = \mathcal{N}(\tilde{\mathbf{u}}'|\tilde{\mathbf{u}}^{(k)}, \beta^2 C) &\implies \tilde{\mathbf{u}}' = \tilde{\mathbf{u}}^{(k)} + \beta L\mathbf{w}^k \\ q_{\text{pCN}}(\tilde{\mathbf{u}}'|\tilde{\mathbf{u}}^{(k)}) = \mathcal{N}(\tilde{\mathbf{u}}'|\sqrt{1-\beta^2}\tilde{\mathbf{u}}^{(k)}, \beta^2 C) &\implies \tilde{\mathbf{u}}' = \sqrt{1-\beta^2}\tilde{\mathbf{u}}^{(k)} + \beta L\mathbf{w}^k \\ \mathbf{w}^k \sim \mathcal{N}(\mathbf{0}, I) \ \forall \ k & \\ C = LL^\intercal & \end{aligned} \tag{3}$$

and with derivation in Appendix A, we arrive at (4). Note here that q_{pCN} depends only on the data likelihood for all values of β, iff the proposal uses the prior matrix C. As shown in the lecture notes, this also means that pCN preserves the prior measure across samples: if $\mathbf{x}^k \sim \mathcal{N}(0, C)$, then $\mathbf{x}^{k+1} \sim \mathcal{N}(0, C)$.

$$\begin{aligned} \log \alpha_{\text{MH}} &= \log p(\mathbf{v}|\tilde{\mathbf{u}}') + \log p(\tilde{\mathbf{u}}') - \log p(\mathbf{v}|\tilde{\mathbf{u}}^{(k)}) - \log p(\tilde{\mathbf{u}}^{(k)}) \\ \log \alpha_{\text{pCN}} &= \log p(\mathbf{v}|\tilde{\mathbf{u}}') - \log p(\mathbf{v}|\tilde{\mathbf{u}}^{(k)}) \end{aligned} \tag{4}$$

The code for everything described above are compared for the two algorithms in Appendix C. The algorithms are run for $T = 10000$ iterations every time. The performance of the two algorithms can be compared in three ways: (1) Proportion of times a proposal is accepted - i.e. the computational efficiency; (2) Error between the inferred u and the true u; (3) Rate of convergence of the samples to the true distribution $p(\mathbf{u}|\mathbf{v})$

1.2.1 Acceptance rates

Comparing the acceptance rate (the number of times the chain moved) for each algorithm is a good measure of computational efficiency. Table 1.2.1 tabulates this for a selected number of parameters, while all results found are tabulated in Appendix E. Table 1.2.1 captures some of the starkest differences in acceptance rates, with that of pCN exceeding MH in all cases. More intuitive behaviour includes higher acceptance rate with lower D - the reduced latent space dimensionality means the chain is more likely to give an increased posterior proposal - and with decrease β - the chain is more likely to continue exploiting a high-posterior region in the latent space, rather than explore other regions. However decreasing β too much increases risk of continually exploiting a local optimum, with any attempts to leave rejected. Hence the chain can 'get stuck' for low β, reducing likelihood of global convergence.

Algorithm	D	ℓ	β	Acceptance rate
MH	4	0.3	0.02	0.97
pCN				0.98
MH	4	0.3	0.50	0.28
pCN				0.68
MH	4	0.3	1.00	0.04
pCN				0.34
MH	16	0.3	0.02	0.85
pCN				0.92
MH	16	0.3	0.50	0.0
pCN				0.07
MH	16	0.3	1.00	0.0
pCN				0.0

Table 1: Some acceptance rates found. In all cases, error bars (standard deviation) over 3 trials were <0.001. Full results, including those with ℓ varied, can be found in Appendix E

1.2.2 Latent error field

We first evaluate the bias of the samples from each chain as estimates of the true latent variable $\mathbf{u}$. In other words, what is the error $|\mathbf{u} - \hat{\mathbf{u}}| = |\mathbf{u} - \frac{1}{T}\sum_{k=1}^{T} \tilde{\mathbf{u}}^{(k)}|$. The issue of convergence is addressed in the next section, and the problem of sample correlation can also be analysed extensively. In reality, MCMC methods like burn-in time and thinning would be used to reduce the chain length before the estimate $\hat{\mathbf{u}}$ is made. Here however, we use all T samples, forgoing these improvements. Figure 2 plots the error field for one set of parameters - $D = 16, \ell = 0.3, N/M = 4, beta = 0.2$

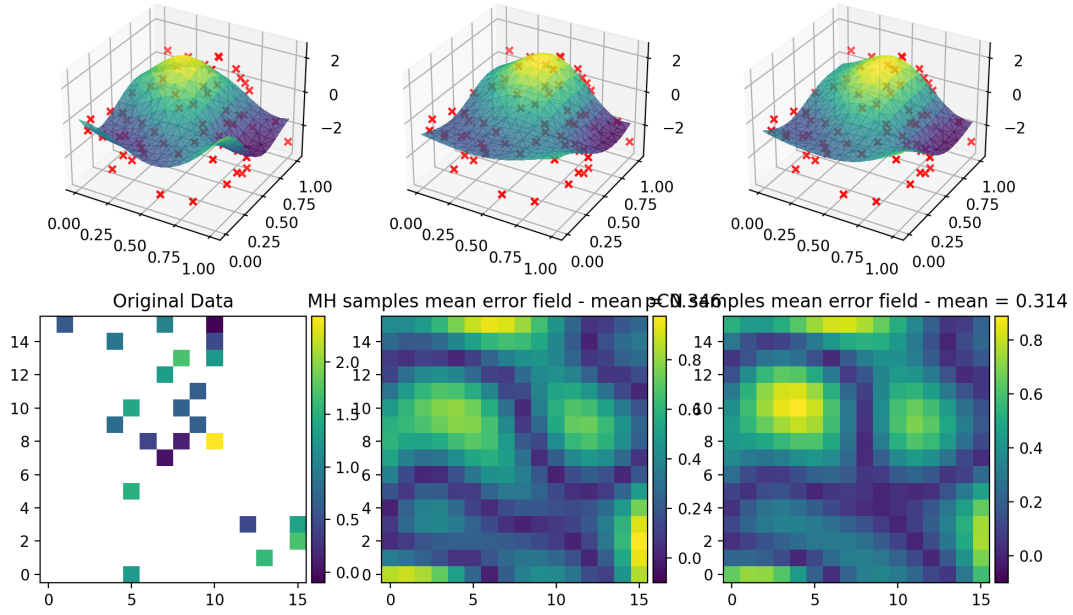

Figure 2: Left column: original latent field and noisy observations (data). Center and right columns: inferred latent fields and error fields for MH and pCN algorithms, with mean absolute error 0.346 and 0.314, respectively.

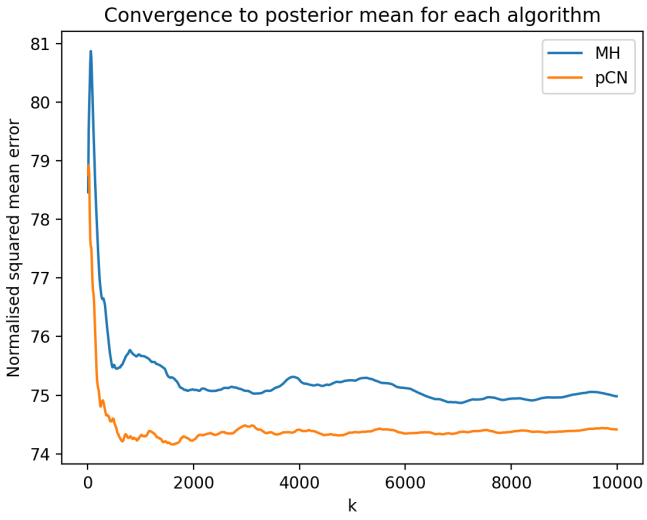

Figure 3: Convergence of chain mean to true posterior mean. (Lack of) burn-in effects can be seen at the start of the chain.

1.2.3 Convergence rates

Finally, we can directly compare the empirical distribution of our latent variables to that of the target posterior. Derived in Appendix B, the target distribution is:

$$p(\mathbf{u}|\mathbf{v}) = \mathcal{N}(\mathbf{u}|\boldsymbol{\mu}, \Sigma) \quad \text{s.t.} \quad \Sigma = (C^{-1} + G^\intercal G)^{-1}, \ \boldsymbol{\mu} = \Sigma G^\intercal \mathbf{v} \tag{5}$$

Approximating the samples as being drawn from a Gaussian, the divergence at time step k can be measured (amongst other metrics) by the KL divergence $KL\left[\mathcal{N}(\hat{\boldsymbol{\mu}}^{(k)}, \hat{\Sigma}^{(k)})||\mathcal{N}(\boldsymbol{\mu}, \Sigma)\right]$ or the Wassestein-2 difference $W_2\left[\mathcal{N}(\hat{\boldsymbol{\mu}}^{(k)}, \hat{\Sigma}^{(k)}), \mathcal{N}(\boldsymbol{\mu}, \Sigma)\right]$. In both cases, we have defined the empirical statistics as:

$$\hat{\boldsymbol{\mu}}^{(\kappa)} = \frac{1}{\kappa}\sum_{k=1}^{\kappa} \tilde{\mathbf{u}}^{(k)}; \quad \hat{\Sigma}^{(\kappa)} = \frac{1}{\kappa}\sum_{k=1}^{\kappa} (\tilde{\mathbf{u}}^{(k)} - \hat{\boldsymbol{\mu}}^{(\kappa)})(\tilde{\mathbf{u}}^{(k)} - \hat{\boldsymbol{\mu}}^{(\kappa)})^\intercal \tag{6}$$

Unfortunately, due to numerical instability when evaluating these metrics, we are forced to use the normalised square mean error:

$$d\left[\mathcal{N}(\hat{\boldsymbol{\mu}}^{(k)}, \hat{\Sigma}^{(k)}), \mathcal{N}(\boldsymbol{\mu}, \Sigma)\right] = \frac{1}{N}(\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}^{(k)})^\intercal(\boldsymbol{\mu} - \hat{\boldsymbol{\mu}}^{(k)}) \tag{7}$$

The evolution of this convergence metric is visualised in Figure 3, where we see that pCN approximates the true posterior with higher accuracy than MH. Note here that the shape of these curves was heavily volatile. Scale and even direction of the evolution was very varied between trials, making population meaning very difficult. This also revealed the sensitivity of the chain to likelihood shape and initialisation point.

1.3 Question c

Now, we extend the model to work for the probit classification problem, where the data is thresholded into a binary variable, giving a (log) likelihood of the form:

$$t_i = \mathbb{I}[v_i > 0], \quad i = 1, ..., M \implies p(t_i = 1|\mathbf{u}) = p(v_i > 0|[G\mathbf{u}]_i) = p([G\mathbf{u} + \boldsymbol{\epsilon}]_i > 0) = \Phi([G\mathbf{u}]_i) \tag{8}$$

$$\implies \begin{cases} \log p(\mathbf{t}|\mathbf{u}) = \log \prod_{i=1}^{M} \left(\Phi([G\mathbf{u}]_i)^{t_i} + (1-\Phi([G\mathbf{u}]_i))^{1-t_i}\right) = \sum_i^{M=1} \left(t_i \log \Phi([G\mathbf{u}]_i) + (1-t_i)\log(1-\Phi([G\mathbf{u}]_i))\right) \\ \\ \texttt{def log\_probit\_likelihood(u, t, G):} \\ \quad \texttt{p\_1 = norm.cdf(G @ u)} \\ \quad \texttt{return (t*np.log(p\_1) + (1-t)*np.log(1-p\_1)).sum()} \end{cases}$$

At inference for data not necessarily subsampled for $\mathbf{v}$, we use the predictive distribution:

$$p(t^* = 1|\mathbf{t}) = \int p(t^* = 1|\mathbf{u})p(\mathbf{u}|\mathbf{t})d\mathbf{u} \approx \sum_{t=1}^{T} p(t^* = 1|\tilde{\mathbf{u}}^{(k)}) \implies \begin{cases} p(t^* = 1|\mathbf{t}) \approx \frac{1}{T}\sum_{k=1}^{T} \Phi(\tilde{u}^{*(k)}) \\ \\ \texttt{def predict\_t(samples):} \\ \quad \texttt{p\_t = list(map(norm.cdf, samples))} \\ \quad \texttt{return np.mean(p\_t, axis = 0)} \end{cases} \tag{9}$$

where we have used the fact that $\tilde{\mathbf{u}}^{(k)} \sim p(\mathbf{u}|\mathbf{t})\ \forall\ k = 1, ..., n$ to create an unbiased estimator of the predictive distribution. For one set of parameters and a sampler using the pCN scheme, Figure 4 leftmost shows the full dataset $\mathbf{t}_{\text{true}} = [\mathbb{I}[u_1 > 0], ..., \mathbb{I}[u_N > 0]]^\intercal$, and Figure 4 left shows the noisy data $t_i = \mathbb{I}[v_i > 0]$ revealed to the sampler. After sampling, Figure 4 right shows the predictive distribution for each set of coordinates.

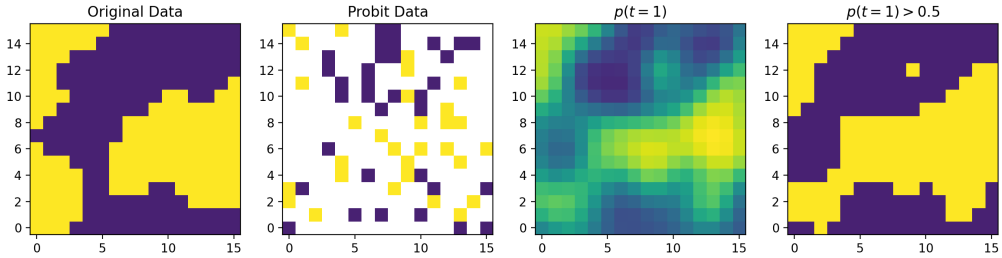

Figure 4: $D = 16, \ell = 0.15, N/M = 4, \beta = 0.2$. Predictions look a bit like Europe.

1.4 Question d

In the previous section, the pCN sampler uses exact prior knowledge of the latent variable by having access to ℓ, which is uses to propose new samples. We are interested in the performance loss of altering the length scale ℓ^* that pCN uses to generate samples. We can analyse the performance of our samples from the Bernoulli distribution $p(t^* = 1|\mathbf{t})$ by thresholding the predictive distribution at 0.5 (i.e. $\hat{t}^* = \mathbb{I}[p(t^* = 1|\mathbf{t}) > 0.5]$), and finding the accuracy rate $\frac{1}{N}\sum_{i=1}^{N} \mathbb{I}[t_i^* = \hat{t}_i^*]$ (i.e. the inverse of error rate). This thresholding has been done in Figure 4 rightmost for one set of parameters. For a fixed $\beta = 0.2$, Figure 5 shows the accuracy rate for some values of ℓ and a range of values of ℓ^*.

The rational choice of length scale when modelling data is the value which maximises accuracy. These values are $\hat{\ell} = \text{argmax}_{\ell^*}(\text{acc}(\ell^*)) = 0.07055, 0.1417, 0.3765$ for $\ell = 0.1, 0.2, 0.3$ respectively. In lieu of a better grid search performance metric, we find the factor by which this is better than the supposed optimum, i.e. $r = \text{acc}(\ell)/\text{acc}(\hat{\ell})$, for which values close to 1 are optimal. We find that $r = 0.962, 0.977, 0, 971$, i.e. this heuristic for selecting ℓ works best for $\ell = 0.2$, but is regardless sensitive to randomness and parameters. Instead of a grid search, inferring ℓ would be optimal. Just as $\mathbf{v}$ is our observations and $\mathbf{u}$ our hidden variables, ℓ can be likened to the model form in Bayesian machine learning. As such, we can get an MLE by maximising model evidence, $\ell_{ML} = \text{argmax}\ p(\mathbf{t}|\ell)$. Going further, a prior can be introduced to l, allowing a Bayesian inference of ℓ as a model parameter itself; for example, a MAP estiamte would be $\ell_{MAP} = \text{argmax}\ p(\ell|\mathbf{t}) = \text{argmax}(p(\mathbf{t}|\ell)p(\ell))$.

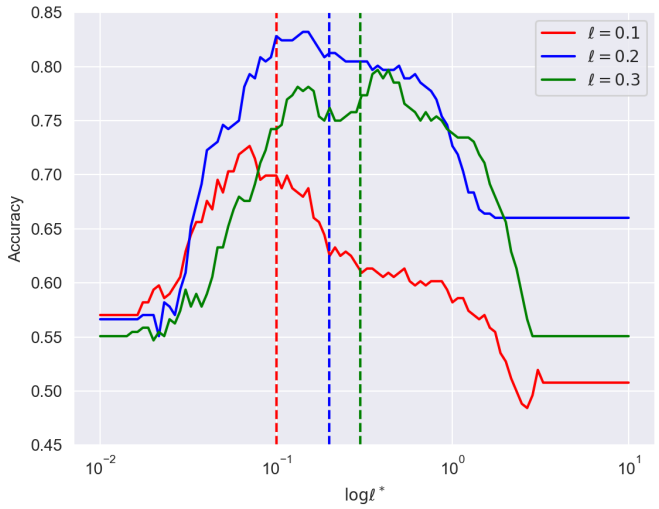

Figure 5: Accuracy for various true and test values of ℓ.

2 Spatial

2.1 Question e & f

Now, we are working with spatial data $\mathbf{c} \in \mathbb{R}^M+$. For a latent field $\mathbf{u}$, of which a subset is mapped to Poisson parameters $\boldsymbol{\theta} = [\exp([G\mathbf{u}]_1), ..., \exp([G\mathbf{u}]_M)]^\intercal$, we assume the corresponding subset of data is sampled independently from the relevant Poisson distribution. This induces a likelihood:

$$p(\mathbf{c}|\boldsymbol{\theta}) = \prod_{i=1}^{M} \frac{\theta_i^{c_i} \exp(-\theta_i)}{c_i!} \implies \begin{cases} \log p(\mathbf{c}|\mathbf{u}) = \sum_{i=1}^{M} \{c_i \log([G\mathbf{u}]_i) - [G\mathbf{u}]_i\} + \text{Const}_{\mathbf{u}} \\ \\ \texttt{def log\_poisson\_likelihood(u, c, G):} \\ \quad \texttt{subsampled\_u, subsampled\_c = G @ u, G @ c} \\ \quad \texttt{log\_term = subsampled\_c * subsampled\_u))} \\ \quad \texttt{return (log\_term - np.exp(subsampled\_u)).sum()} \end{cases} \tag{10}$$

which is derived in Appendix D, and again neglects constants with respect to the random variables. At inference, expected counts are given by

$$\mathbb{E}_{p(\mathbf{c}^*|\mathbf{c})}[\mathbf{c}^*] = \sum_{r=0}^{\infty} r p(\mathbf{c}^* = r|\mathbf{c}) = \sum_{r=0}^{\infty} r \int p(\mathbf{c}^* = r, \mathbf{u}|\mathbf{c}) d\mathbf{u} = \sum_{r=0}^{\infty} r \int p(\mathbf{c}^* = r|\mathbf{u}) p(\mathbf{u}|\mathbf{c}) d\mathbf{u} \tag{11}$$

This allows the MC estimation:

$$\hat{\mathbf{c}}^* = \mathbb{E}_{p(\mathbf{c}^*|\mathbf{c})}[\mathbf{c}^*] \approx \frac{1}{T} \sum_{r=0}^{\infty} r \sum_{k=1}^{T} p(\mathbf{c}^* = k|\tilde{\mathbf{u}}^{(k)}) = \frac{1}{T} \sum_{k=1}^{T} \mathbb{E}_{p(\mathbf{c}^*|\tilde{\mathbf{u}}^{(k)})}[\mathbf{c}^*] \overset{\text{(i)}}{=} \frac{1}{T} \sum_{k=1}^{T} \exp(\tilde{\mathbf{u}}^{(k)}) \tag{12}$$

where (i) makes use of the fact that $\mathbb{E}_{f(\mathbf{c}^*|\theta^*)}[\mathbf{c}^*] = \theta^* \implies \mathbb{E}_{f(\mathbf{c}^*|\tilde{\mathbf{u}}^{(k)})}[\mathbf{c}^*] = \tilde{\boldsymbol{\theta}}^{(k)} = \exp(\tilde{\mathbf{u}}^{(k)})$. For fixed ℓ and β, Figure 6 shows the full counts field $\mathbf{c}_{\text{true}}$, the observed counts $\mathbf{c}$, the expected counts $\hat{\mathbf{c}}$, and the absolute error $|\hat{c} - c_{\text{true}}|$ for each set of coordinates.

We already see a resemblance between the infered and true counts field for these parameters. However, the model struggles with localised high count areas, such as in the north. This suggests a larger ℓ than is accurate, prompting a wider search to test this.

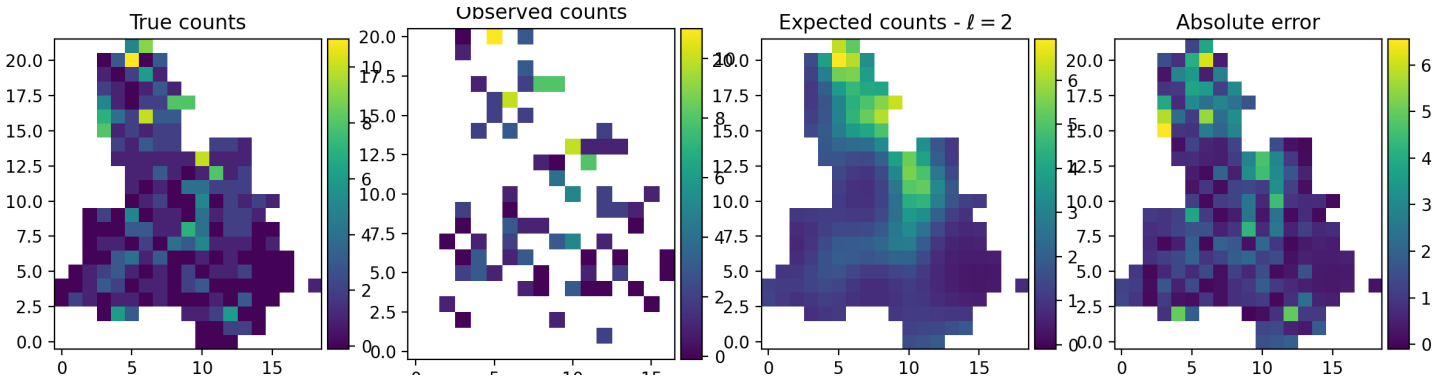

Figure 6: Data and predictions for $\ell = 2, \beta = 0.2, T = 10000, N/M = 3$.

Figure 7 plots the mean absolute error as the test length scale ℓ^* is varied, with β kept fixed. For some salient values, Figure 8 plots the count field MC estimated (instead of MC estimated latent variable, for which the relationship is provided above).

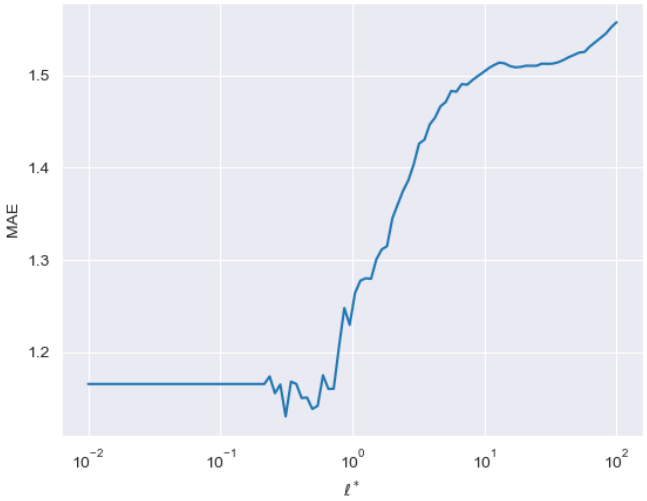

Figure 7: Mean absolute error $\frac{1}{N}\|\hat{\mathbf{c}} - \mathbf{c}_{\text{true}}\|_1$, a measure of (poor) model performance, as ℓ^*, used to model the counts, is changed

For the larger length scales, we see an extreme version of the lack of localisation, with the whole map becoming one North-South gradient. However, for $\ell = 0.01$, the inverse problem is witnessed, with the highly localised model struggling to generalise observations $\mathbf{c}$ to the full data $\mathbf{c}_{\text{true}}$.

3 Conclusion

We have used the Metropolis-Hastings and the Preconditioned Crank-Nicolson Markov chain Monte Carlo methods to sample from a distribution. This distribution is the posterior over latent variable field sampled from a Gaussian process field, where only noisy observations of the field were observed (Section 1.1). We learned how the Gaussian process form, through the length scale of its covariance matrix, and the chain dynamics, through its proposal distribution, have affected the convergence and performance of the MCMC sampler (Sections 1.2, 1.3, 1.4). This was used to optimise a model to predict the number of bike thefts in regions of Lewisham (Section 2.1).

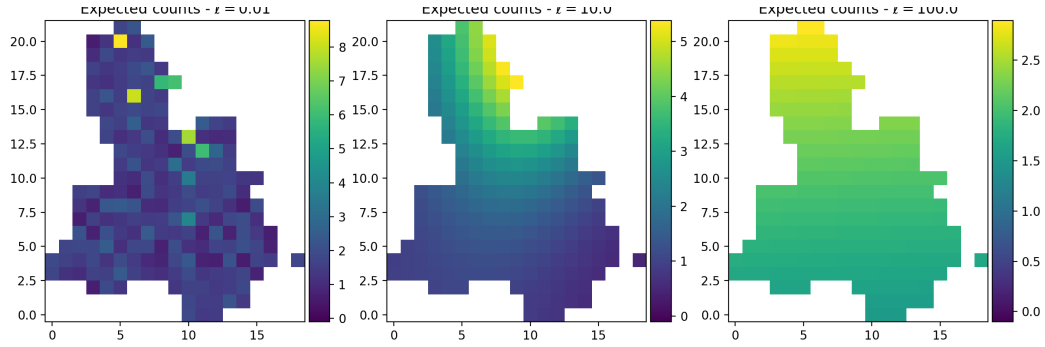

Figure 8: Inferred counts fields for various values of ℓ^*. It is clear that $\ell = 0.01$, which scored amongst the lowest in mean absolute error, also resembles the true counts field the most.

Future work that could lead directly from this coursework was outlined at the end of (Section 1.4). So far, we have used Bayes's rule to infer some hidden variables from an observed dataset: $p(\theta|x) \propto p(x|\theta)p(\theta)$. However, this formulation skips over the dependence on the model form $\mathcal{M}$, which would extend the Bayesian inference: $p(\theta|x, \mathcal{M}) \propto p(x|\theta, \mathcal{M})p(\theta|\mathcal{M})$. Our tuning of ℓ corresponds to tuning the final term here, $p(\theta|\mathcal{M})$. However for a full picture, we should infer a posterior over $\mathcal{M}$ itself. □

Appendix A Deriving acceptance probabilities

Ignoring the min operator, (2) for MH can be written as:

$$\begin{aligned}
\log \alpha_{\text{MH}} &\overset{\text{a}}{=} \log p(\tilde{\mathbf{u}}'|\mathbf{v}) + \log q_{\text{MH}}(\tilde{\mathbf{u}}^{(k)}|\tilde{\mathbf{u}}') - \log p(\tilde{\mathbf{u}}^{(k)}|\mathbf{v}) - \log q_{\text{MH}}(\tilde{\mathbf{u}}'|\tilde{\mathbf{u}}^{(k)}) \\
&\overset{\text{b}}{=} \log p(\tilde{\mathbf{u}}', \mathbf{v}) - \log p(\tilde{\mathbf{u}}^{(k)}, \mathbf{v}) - \log q_{\text{MH}}(\tilde{\mathbf{u}}'|\tilde{\mathbf{u}}^{(k)}) + \log q_{\text{MH}}(\tilde{\mathbf{u}}^{(k)}|\tilde{\mathbf{u}}') \\
&\overset{\text{c}}{=} \log p(\tilde{\mathbf{u}}', \mathbf{v}) - \log p(\tilde{\mathbf{u}}^{(k)}, \mathbf{v}) \\
&\overset{\text{d}}{=} \log p(\mathbf{v}|\tilde{\mathbf{u}}') + \log p(\tilde{\mathbf{u}}') - \log p(\mathbf{v}|\tilde{\mathbf{u}}^{(k)}) - \log p(\tilde{\mathbf{u}}^{(k)})
\end{aligned} \tag{13}$$

and for pCN:

$$\begin{aligned}
\log \alpha_{\text{pCN}} &\overset{\text{e}}{=} \log \frac{p(\mathbf{v}|\tilde{\mathbf{u}}')}{p(\mathbf{v}|\tilde{\mathbf{u}}^{(k)})} + \log p(\tilde{\mathbf{u}}') - \log p(\tilde{\mathbf{u}}^{(k)}) + \log q_{\text{pCN}}(\tilde{\mathbf{u}}^{(k)}|\tilde{\mathbf{u}}') - \log q_{\text{pCN}}(\tilde{\mathbf{u}}'|\tilde{\mathbf{u}}^{(k)}) \\
&\overset{\text{f}}{=} \log \frac{p(\mathbf{v}|\tilde{\mathbf{u}}')}{p(\mathbf{v}|\tilde{\mathbf{u}}^{(k)})} - \frac{1}{2}(\langle \tilde{\mathbf{u}}', C^{-1}\tilde{\mathbf{u}}' \rangle - \langle \tilde{\mathbf{u}}^{(k)}, C^{-1}\tilde{\mathbf{u}}^{(k)} \rangle) \\
&\qquad - \frac{1}{2\beta^2}(\langle \tilde{\mathbf{u}}^{(k)} - \sqrt{1-\beta^2}\tilde{\mathbf{u}}', C^{-1}(\tilde{\mathbf{u}}^{(k)} - \sqrt{1-\beta^2}\tilde{\mathbf{u}}') \rangle \\
&\qquad\qquad - \langle \tilde{\mathbf{u}}' - \sqrt{1-\beta^2}\tilde{\mathbf{u}}^{(k)}, C^{-1}(\tilde{\mathbf{u}}' - \sqrt{1-\beta^2}\tilde{\mathbf{u}}^{(k)}) \rangle) \\
&\overset{\text{g}}{=} \log p(\mathbf{v}|\tilde{\mathbf{u}}') - \log p(\mathbf{v}|\tilde{\mathbf{u}}^{(k)})
\end{aligned} \tag{14}$$

Justifications for each equality are:

a Using log law $\log \prod(\cdot) = \sum \log(\cdot)$

b Removing $p(\mathbf{v})$ fromt he conditional, as it appears in numerator and denominator

c The proposal for MH is symmetric, i.e. $q_{\text{MH}}(a|b) = q_{\text{MH}}(b|a)$, so we can remove it from numerator and denominator

d This uses the probability chain rule, giving us some familiar objects

e This uses (a) and (d)

f Expanding the probability terms into their log-Gaussian terms. NB: log normalisation constants are removed in all cases, as they appear in numerator and denominator.

g Expanding and contracting the log-Gaussian terms shows that most of the terms cancel out iff the propsal uses the prior matrix C.

Appendix B True Posterior

We are interested in deriving the true form of the posterior $p(\mathbf{u}|\mathbf{v})$. Given that prior $p(\mathbf{u})$ and likelihood $p(\mathbf{v}|\mathbf{u})$ are both Gaussians, we expect the posterior to also be of Gaussian form. Namely:

$$\begin{aligned} p(\mathbf{u}|\mathbf{v}) \propto p(\mathbf{u},\mathbf{v}) = p(\mathbf{v}|\mathbf{u})p(\mathbf{u}) &\propto \exp\left\{\frac{-1}{2}\left[\mathbf{u}^\intercal C^{-1}\mathbf{u} + \mathbf{v}^\intercal\mathbf{v} - \mathbf{u}^\intercal G^\intercal\mathbf{v} + \mathbf{u}^\intercal G^\intercal G\mathbf{u}\right]\right\} \\ &\propto \exp\left\{\frac{-1}{2}\left[\mathbf{u}^\intercal(C^{-1} + G^\intercal G)\mathbf{u} - \mathbf{u}^\intercal G^\intercal\mathbf{v}\right]\right\} \end{aligned} \tag{15}$$

completing the square for a general distribution:

$$\begin{aligned} \mathcal{N}(\mathbf{u}|\mathbf{m},S) &\propto \exp\left\{\frac{-1}{2}\left[\mathbf{u}^\intercal S^{-1}\mathbf{u} - 2\mathbf{u}^\intercal S^{-1}\mathbf{m} + \mathbf{m}^\intercal S^{-1}\mathbf{m}\right]\right\} \\ &\propto \exp\left\{\frac{-1}{2}\left[\mathbf{u}^\intercal S^{-1}\mathbf{u} - 2\mathbf{u}^\intercal S^{-1}\mathbf{m}\right]\right\} \end{aligned} \tag{16}$$

we arrive at the result used in Section 1.2.

Appendix C Code snippets for MH and pCN implementations

Step	**MH code**	**pCN code**

Inverse cov. matrix	`pdK = K + 1e-6 * np.eye(N)`
	`Kc = np.linalg.cholesky(pdK)`
	`Kc_i = np.linalg.inv(Kc)`
	`K_inverse = Kc_i.T @ Kc_i`
		No need

Propose next sample	`z = np.random.randn(N)`
	`u_new = u_prev + beta * (Kc @ z)`
		`z = np.random.randn(N)`
		`u_new = (`
		`  u_prev*(1-beta**2)**0.5`
		`  + beta * (Kc @ z)`
		`)`

Metropolis step	`lt_new = log_target(`
	`  u_new, data, K_inverse, G`
	`)`
	`log_alpha = lt_new - lt_prev`
	`log_u = np.log(np.random.random())`
	`accept = log_u < log_alpha`
		`ll_new = log_likelihood(`
		`  u_new, data, G`
		`)`
		`log_alpha = ll_new - ll_prev`
		`log_u = np.log(np.random.random())`
		`accept = log_u < log_alpha`

Strict implementation of the algorithm would require `log_alpha = min(lt_new - lt_prev, 0)`, however because `log_u` is upper bounded by 0, this will not change script behaviour.

Appendix D Deriving Poisson Log-Likelihood

$$p(\mathbf{c}|\boldsymbol{\theta}) = \prod_{i=1}^{M} \frac{\theta_i^{c_i} \exp(-\theta_i)}{c_i!} \quad \therefore \ \log p(\mathbf{c}|\boldsymbol{\theta}) = \sum_{i=1}^{M} \left\{ c_i \log(\theta_i) - \theta_i - \sum_{j=1}^{c_i} \log(j) \right\} = \sum_{i=1}^{M} \{ c_i [G\mathbf{u}]_i - \exp([G\mathbf{u}]_i) \} + \text{Const}_{\mathbf{u}}$$

Appendix E Acceptance rates comparison

ℓ	β	Acceptance rate $\frac{\text{MH}}{\text{pCN}}$
0.1	0.01	0.98 0.99
0.1	0.02	0.97 0.98
0.1	0.10	0.82 0.94
0.1	0.20	0.66 0.87
0.1	0.50	0.28 0.7
0.1	1.00	0.04 0.36
0.3	0.01	0.98 0.99
0.3	0.02	0.97 0.98
0.3	0.10	0.83 0.93
0.3	0.20	0.66 0.87
0.3	0.50	0.28 0.68
0.3	1.00	0.04 0.34

ℓ	β	Acceptance rate $\frac{\text{MH}}{\text{pCN}}$
0.1	0.01	0.92 0.96
0.1	0.02	0.86 0.94
0.1	0.10	0.38 0.71
0.1	0.20	0.08 0.46
0.1	0.50	0.0 0.06
0.1	1.00	0.0 0.0
0.3	0.01	0.92 0.96
0.3	0.02	0.85 0.92
0.3	0.10	0.38 0.69
0.3	0.20	0.08 0.44
0.3	0.50	0.0 0.07
0.3	1.00	0.0 0.0

Table 2: All acceptance rates found for D = 4 (left) and D = 16 (right). In all cases, error bars (standard deviation) over 3 trials were <0.001