Coursework 4M24: High-Dimensional MCMC

Tom Ryan

January 10, 2023

Abstract

The aim of this work is to explore the use of Markov Chain Monte-Carlo (MCMC) methods to sample from posterior distributions in a large number of dimensions. We will first explore the inference of Gaussian Process samples from noisy sub-sampled data points, using both Gaussian Random Walk Metropolis-Hastings (GRW-MH) and preconditioned Crank-Nicholson (pCN). We conclude that a subtle difference in the definition of pCN from GRW-MH, will lead to much better acceptance probabilities when sampling in high-dimensional spaces, leading to quicker computation time. Although both approaches are similar in prediction performance. We will then take the pCN approach forward, transforming the previous inference problem to a classification problem. We see promising success with this problem, especially when we consider the effects of softer classifications. Finally we will use pCN to infer bike theft data in the Lewisham borough, from sub-samples of real data. We will show that it is possible to infer the length parameter of the original Gaussian Process prior by minimising predictive error, and explore how this parameter effects the final model.

1 Part I - Simulation

1.1 Prior Sampling and Sub-Sampling

We begin by defining a 2D Gaussian Process with zero mean and squared exponential covariance function

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

Where l is a length scale hyperparameter. We then define latent variables $\{\mathbf{x}_i\}_{i=1}^N$ which make a regular $D \times D$ grid in the domain $x_1, x_2 \in [0, 1]$. Using these latent variables we can sample from the GP prior $\mathbf{u} \sim \mathcal{N}(0, K)$ where $K_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. We generate these samples by generating the kernel K and computing the Cholesky factor L where $K = LL^T$. Samples from the GP prior are then given by

$$u_i = L\mathbf{z} \tag{2}$$

Where $\mathbf{z} \sim \mathcal{N}(0, 1)$. Figure 1 shows samples from the prior for $l \in \{0.1, 0.3, 1\}$ and $D = 16$. We observe that a larger length parameter leads to more covariance between nearby latent variables and so u values for adjacent latent variables are highly dependent and the overall mesh is smoother. On the other hand a smaller length parameter leads to greater independence for nearby latent variables and so there is more variation within the u mesh. We can imagine at this stage that inferring $\mathbf{u}$ for a larger length scale will be easier and require fewer subsamples than for a smaller length scale.
We can now observe a set of sub-samples from the latent variables and inject observation noise, we generate this by introducing $G : \mathbb{R}^N \to \mathbb{R}^M$ so that

$$\mathbf{v} = G\mathbf{u} + \boldsymbol{\epsilon} \tag{3}$$

Where the observation noise is a standard Gaussian $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$. A plot of these sub-samples alongside the latent variables for $l = 0.3$ is shown in figure 2 for $M = 64$ sub-samples. We can notice an even distribution of the sub-samples above and below the u mesh as the noise has mean zero. The variance of the noise is quite large compared to the range of u, it would be difficult to infer the underlying distribution from the sub-samples by eye.

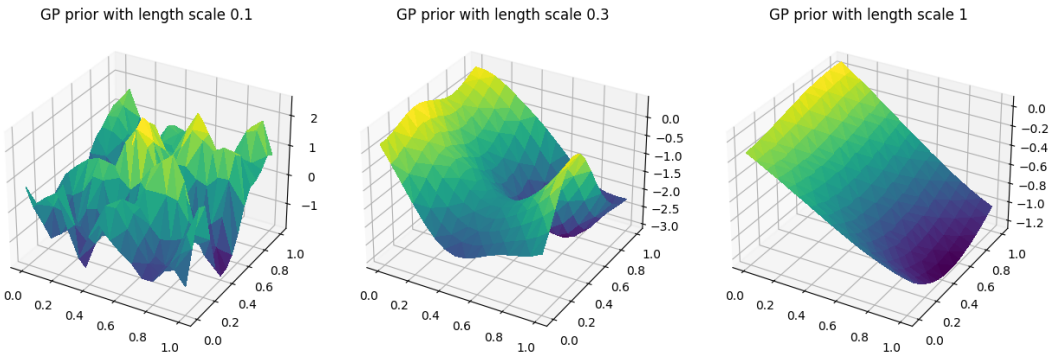

Figure 1: Gaussian Process prior samples for $l \in \{0.1, 0.3, 1\}$

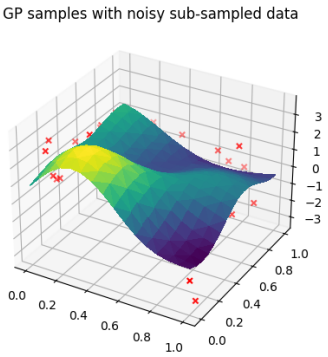

Figure 2: Noisy sub-samples (red crosses) and GP prior with $l = 0.3$

1.2 Sampling from Posterior

1.2.1 Log Probabilities

Now that we have generated the sub-samples $\mathbf{v}$, we are going to assume that we never had access to $\mathbf{u}$ and try to infer $\mathbf{u}$ from $\mathbf{v}$. We are going to do this by sampling from the posterior distribution $p(\mathbf{u}|\mathbf{v}) \propto p(\mathbf{v}|\mathbf{u})p(\mathbf{u})$ using two MCMC approaches: Gaussian Random Walk Metropolis-Hastings (GRW-MH) and preconditioned Crank-Nicolson (pCN). In both of these cases, we need to be able to compute the log-prior. We assume we know the length parameter (for now) and so can compute the prior covariance K, so the log-prior is given by

$$\ln p(\mathbf{u}) = \ln \mathcal{N}(0, K) = -\frac{N}{2} \ln 2\pi - \frac{1}{2} \ln |K| - \frac{1}{2} \mathbf{u}^T K^{-1} \mathbf{u} \tag{4}$$

For simplicity we denote the log-prior as $\ln p(.) := \pi(.)$. In order to compute GRW-MH steps, we also require the log-likelihood. From equation 3 we know that $p(\mathbf{v}|\mathbf{u}) = \mathcal{N}(G\mathbf{u}, I)$ so the log-likelihood is

$$\ln p(\mathbf{v}|\mathbf{u}) = -\frac{M}{2} \ln 2\pi - \frac{1}{2} (\mathbf{v} - G\mathbf{u})^T (\mathbf{v} - G\mathbf{u}) \tag{5}$$

Again for simplicity we denote the log-likelihood as $\ln p(\mathbf{v}|\mathbf{u} = .) = \mathcal{L}(.)$. For pCN we require the log-posterior distribution but since we already have the prior and likelihood (and do not need to normalise - see later) we can compute

$$\ln p(\mathbf{u}|\mathbf{v}) = \mathcal{L}(\mathbf{u}) + \pi(\mathbf{u})$$

We denote the posterior $\ln p(\mathbf{u} = .|\mathbf{v}) = \mathcal{P}(.)$.

1.2.2 GRW-MH

For a Metropolis-Hastings algorithm we need a proposal distribution $q(\tilde{\mathbf{u}}^{(i)}, \tilde{\mathbf{u}}^{(i+1)})$ where $\tilde{\mathbf{u}}^{(n)}$ is our sample from the posterior distribution $p(\mathbf{u}|\mathbf{v})$ at step n. For a Gaussian Random Walk we step based

on the equation

$$\tilde{\mathbf{u}}^{(i+1)} = \tilde{\mathbf{u}}^{(i)} + \beta\mathcal{N}(0, I) \tag{6}$$

Where β is the step size. As such the proposal distribution $q(a, b)$ is given by

$$q(a, b) = \mathcal{N}(b; a, I) \tag{7}$$

Note that this means that the proposal distribution is symmetric, i.e. $q(a, b) = q(b, a)$. This allows us to re-write the acceptance probability for the MH algorithm which is traditionally

$$\alpha(\tilde{\mathbf{u}}^{(i)}, \tilde{\mathbf{u}}^{(i+1)}) = \min\left(\frac{\rho(\tilde{\mathbf{u}}^{(i+1)})q((\tilde{\mathbf{u}}^{(i+1)}, (\tilde{\mathbf{u}}^{(i)})}{\rho(\tilde{\mathbf{u}}^{(i)})q((\tilde{\mathbf{u}}^{(i)}, (\tilde{\mathbf{u}}^{(i+1)})}, 1\right) \tag{8}$$

Where $\rho(.) = \exp\mathcal{P}(.)$. Since the proposal distribution is symmetric the q terms cancel and we get

$$\alpha(\tilde{\mathbf{u}}^{(i)}, \tilde{\mathbf{u}}^{(i+1)}) = \min\left(\frac{\rho(\tilde{\mathbf{u}}^{(i+1)})}{\rho(\tilde{\mathbf{u}}^{(i)})}, 1\right) \tag{9}$$

This is why we did not need to compute the constant term in the posterior. In the log domain this is

$$\ln\alpha(\tilde{\mathbf{u}}^{(i)}, \tilde{\mathbf{u}}^{(i+1)}) = \min\left(\mathcal{P}(\tilde{\mathbf{u}}^{(i+1)}) - \mathcal{P}(\tilde{\mathbf{u}}^{(i)}), 0\right) \tag{10}$$

Which allows us to perform GRW-MH as shown in algorithm 1

Algorithm 1 Gaussian Random Walk Metropolis-Hastings

for $i \leftarrow 0$ to $T-1$ **do**
  $\tilde{\mathbf{u}}^{(i+1)} \leftarrow \tilde{\mathbf{u}}^{(i)} + \beta\mathcal{N}(0, I)$
  $z \leftarrow \ln\mathcal{U}(0, 1)$
  $\alpha \leftarrow \min(\mathcal{P}(\tilde{\mathbf{u}}^{(i+1)}) - \mathcal{P}(\tilde{\mathbf{u}}^{(i)}), 0)$
  if $z > \alpha$ **then**
    $\tilde{\mathbf{u}}^{(i+1)} \leftarrow \tilde{\mathbf{u}}^{(i)}$
  end if
end for
return $\{\tilde{\mathbf{u}}^{(0)}, \tilde{\mathbf{u}}^{(1)}, ..., \tilde{\mathbf{u}}^{(T-1)}\}$

1.2.3 pCN

Preconditioned Crank-Nicolson has a few subtle differences to GRW-MH, to ensure that everything is well defined in infinite dimensions. The most obvious change is that each step of the algorithm is based on the equation

$$\tilde{\mathbf{u}}^{(i+1)} = \sqrt{1-\beta^2}\tilde{\mathbf{u}}^{(i)} + \beta\mathcal{N}(0, I) \tag{11}$$

Simply a $\sqrt{1-\beta^2}$ term different compared to the GRW in equation 6, however this is crucial as it ensures the acceptance probability is well defined in infinite dimensions. The acceptance probability is now

$$\alpha = \min\left(\exp(\phi(\tilde{\mathbf{u}}^{(i+1)}) - \phi(\tilde{\mathbf{u}}^{(i)})), 1\right) \tag{12}$$

Where $\phi = -\mathcal{L}$. We can express this in the log domain as

$$\ln\alpha = \min\left(\phi(\tilde{\mathbf{u}}^{(i+1)}) - \phi(\tilde{\mathbf{u}}^{(i)}), 0\right) \tag{13}$$

Which allows us to perform pCN as shown in algorithm 2. An immediately apparent advantage to this approach is that we do not have to compute the prior, making it more computationally efficient.

Algorithm 2 Preconditioned Crank-Nicolson

for $i \leftarrow 0$ to $T-1$ **do**
  $\tilde{\mathbf{u}}^{(i+1)} \leftarrow \sqrt{1-\beta^2}\tilde{\mathbf{u}}^{(i)} + \beta\mathcal{N}(0, I)$
  $z \leftarrow \ln\mathcal{U}(0,1)$
  $\alpha \leftarrow \min(\phi(\tilde{\mathbf{u}}^{(i+1)}) - \phi(\tilde{\mathbf{u}}^{(i)}), 0)$
  if $z > \alpha$ **then**
    $\tilde{\mathbf{u}}^{(i+1)} \leftarrow \tilde{\mathbf{u}}^{(i)}$
  end if
end for
return $\{\tilde{\mathbf{u}}^{(0)}, \tilde{\mathbf{u}}^{(1)}, ..., \tilde{\mathbf{u}}^{(T-1)}\}$

1.2.4 Comparison of GRW-MH and pCN

We can use both GRW-MH and pCN to sample from the posterior distribution $p(\mathbf{u}|\mathbf{v})$. From the posterior samples $\{\tilde{u}^{(i)}\}$ we compute an estimate for $\mathbf{u}$ using a Monte-Carlo estimate (essentially we just average over samples).

$$\tilde{\mathbf{u}} = \frac{1}{T}\sum_{i=0}^{T-1}\tilde{\mathbf{u}}^{(i)} \tag{14}$$

Figure 3 shows the curves of $\tilde{\mathbf{u}}$ with the sub-sampled data $\mathbf{v}$, alongside the original $\mathbf{u}$. There isn't much difference between these plots visually, they both appear to model the original $\mathbf{u}$ quite well. We can quantify their accuracy by looking at the error fields, shown in figure 4.

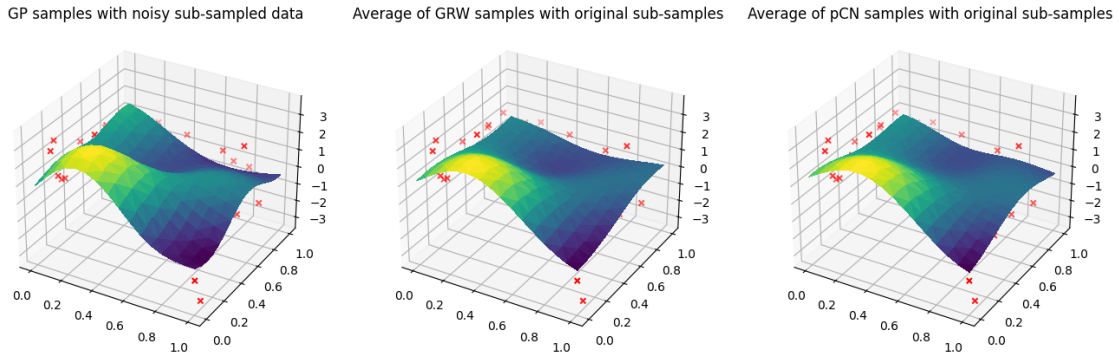

Figure 3: Original u samples (left), with inferred samples from GRW (middle) and pCN (right)

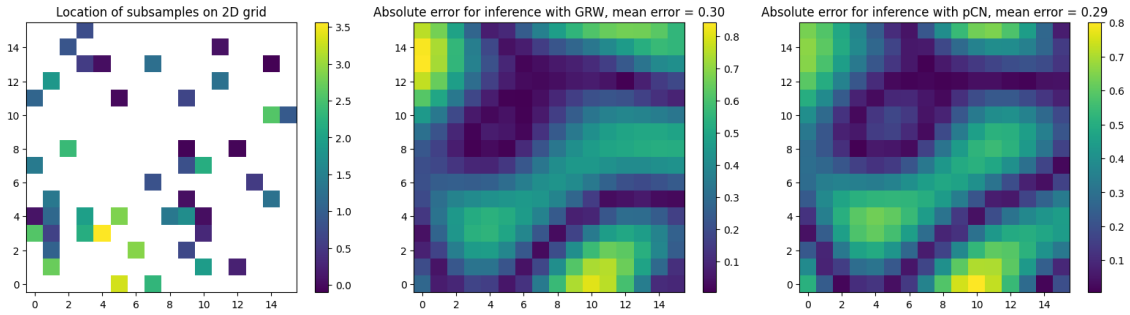

Figure 4: Location of sub-samples (left) with absolute errors for GRW (middle) and pCN (right)

The errors are generally small, with pCN sampling having a slightly better overall error than GRW. We can see that there is some connection between the locations of samples and the error field, however there is also low error in areas with fewer data points. We can attribute this to the fact we have allowed the model to use the length parameter, so the covariance between points is known and this

can be used to infer in areas with little data.

We now look at the acceptance probabilities for each approach, as this governs the speed at which the algorithm runs. There are two main factors that affect the acceptance probability, the dimension $\mathcal{D}$ of the problem, and the step size β. Increasing either of these values makes it less likely for two subsequent samples to satisfy detailed balance, which reduces the acceptance probability as this is defined as the probability required to make a transition reversible. We investigate the effects of varying both of these parameters on the acceptance probability for both GRW and pCN, for $\beta \in \{0.01, 0.05, 0.1, 0.5, 1\}$ and $\mathcal{D} \in \{4, 16\}$. The results of this are shown in table 1, we see the expected decreasing of acceptance probability for growing dimensions and step size. We also observe a much better acceptance probability for pCN than GRW, especially when $\mathcal{D} = 16$. This is to be expected, as pCN by design is more robust to an increase in dimensionality, in fact it is well defined for an infinite number of dimensions, whereas GRW-MH is not. Although a smaller step size β leads to a greater acceptance probability, we cannot make it arbitrarily small. Trivially we cannot use $\beta = 0$, because we would always draw the same sample. Furthermore the smaller the value of β the less exploratory the sampler is of the posterior distribution, we end up with a greater auto-correlation between subsequent samples and the sampler will take much longer to converge to the invariant distribution.

(GRW | pCN)		β				
		0.01	0.05	0.1	0.5	1
$\mathcal{D}$	4	0.978 | 0.982	0.904 | 0.928	0.821 | 0.907	0.291 | 0.523	0.035 | 0.057
	16	0.909 | 0.960	0.621 | 0.847	0.380 | 0.693	0.00 | 0.099	0.00 | 0.004

Table 1: Acceptance probabilities for a variety of $\mathcal{D}$ and β for both pCN and GRW

1.3 Sampling from Probit Classification Posterior

We augment our model to work on a probit classification problem. The binary classes are determined based on probit function

$$t_i = \begin{cases} 0 & \text{if } v_i \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{15}$$

Then, since the sub-samples $\mathbf{v}$ have standard normal observation noise the likelihood is given by

$$p(t_i = 1|\mathbf{u}) = \Phi([G\mathbf{u}]_i) \tag{16}$$

We want to compute the log-likelihood $\mathcal{L}(.) = \ln p(\mathbf{t}|\mathbf{u} = .)$. We know that

$$p(\mathbf{t}|\mathbf{u}) = \prod_{i=0}^{M-1} p(t_i|\mathbf{u}) \tag{17}$$

So then

$$\mathcal{L}(\mathbf{u}) = \sum_{i=0}^{M-1} \ln p(t_i|\mathbf{u}) \tag{18}$$

Since $p(t_i = 1|\mathbf{u}) = \Phi([G\mathbf{u}]_i)$ and $p(t_i = 0|\mathbf{u}) = \Phi(-[G\mathbf{u}]_i)$ we can rewrite this as

$$\mathcal{L}(\mathbf{u}) = \mathbf{t} \cdot \ln \Phi(G\mathbf{u}) + (\mathbf{1} - \mathbf{t}) \cdot \ln \Phi(-G\mathbf{u}) \tag{19}$$

Which is easily and efficiently computable. With this likelihood function we can generate samples from $p(\mathbf{u}|\mathbf{t})$ using pCN as before. The predictive distribution for the class at latent variable index j t^* is given by

$$p(t_j^* = 1|\mathbf{t}) = \int p(t_j^* = 1, \mathbf{u}|\mathbf{t}) d\mathbf{u} = \int p(t_j^* = 1|\mathbf{u}) p(\mathbf{u}|\mathbf{t}) d\mathbf{u} \tag{20}$$

Since we have samples from $p(\mathbf{u}|\mathbf{t})$ we can estimate the value of this integral using a Monte-Carlo approximation

$$p(t_j^* = 1|\mathbf{t}) \approx \frac{1}{T}\sum_{i=0}^{T-1} p(t_j^* = 1|\tilde{\mathbf{u}}^{(i)}) = \frac{1}{T}\sum_{i=0}^{T-1} \Phi(\tilde{u}_j^{(i)}) \tag{21}$$

Where $u_j^{(i)}$ is the pCN sample for latent variable j at iteration i. If we want the predictive distribution over all latent variable positions we can write

$$p(\mathbf{t}^* = \mathbf{1}|\mathbf{t}) = \frac{1}{T}\sum_{i=0}^{T-1} \Phi(\tilde{\mathbf{u}}^{(i)}) \tag{22}$$

The predictive distribution using this approximation and the pCN samples is shown in figure 5 alongside the true classifications from the original samples. As we may expect, it looks similar to the inferred pCN $\tilde{\mathbf{u}}$ curve projected onto a 2D plane, however here each sample from the posterior is passed through a Gaussian CDF so it is represented as a probability distribution. It is clear that the predictive probabilities show a similar shape to the true classifications, and so the classifier has worked well.

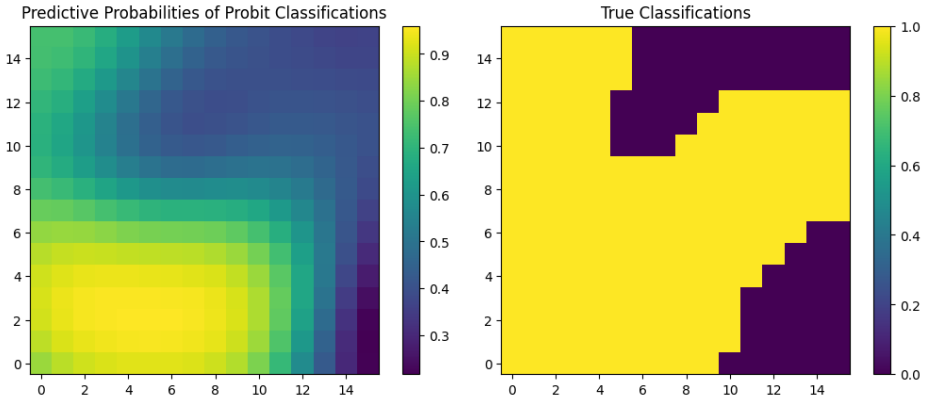

Figure 5: Predictive probabilities of inferred classes (left) and true classifications (right)

1.4 Probit Classification Hard Assignments

We can compute hard classifications at each point by thresholding the predictive distribution at $p = 0.5$. This gives the classes shown in figure 6, this figure also shows the points at which there have been classification errors.

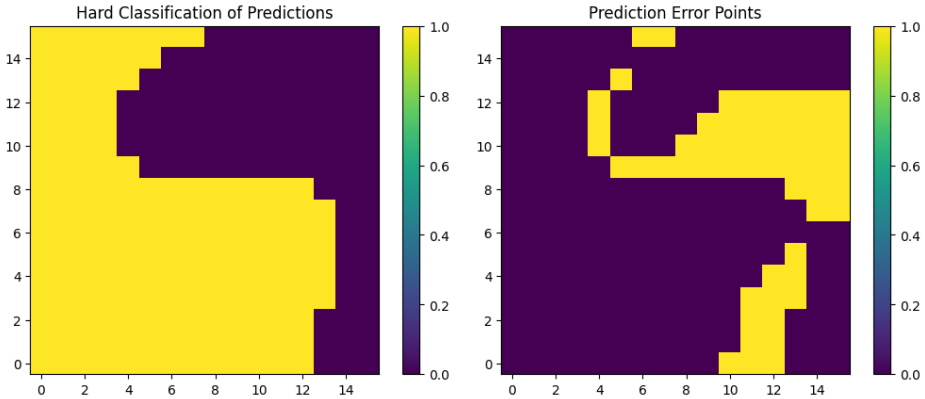

Figure 6: Hard classification for class inference (left) and points with classification errors (right) for $l = 0.3$

There are clearly a reasonable amount of errors from our classifier, the data was initially very noisy and so to have a reasonably working classifier is impressive. Additionally, many of the points that have been given hard classifications have a predictive probability $0.4 < p < 0.6$, and so the model is generally unsure about these points and should not be given a hard classification. If we instead use a soft classifier, and leave points with probability $0.4 < p < 0.6$ unclassified we get the result shown in figure 7. Most of the errors are found in these unclassified areas.

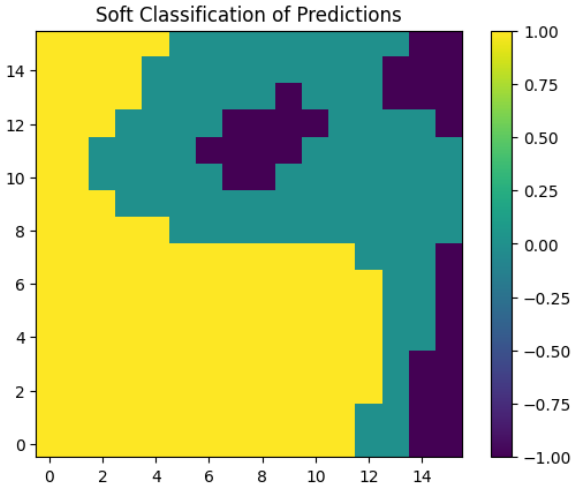

Figure 7: Soft classification with unclassified points taking value 0, and classes taking $t \in \{1, -1\}$

Up until this point we have assumed knowledge of the length parameter l, this is unrealistic, and so it is good to see if we can infer the length parameter using our model. Figure **??** shows the proportion of incorrect classifications based on varying l. We find that the lowest error comes from $l = 10^{-0.5} \approx 0.31$, so we have been able to infer the correct length parameter of the prior by considering the best classifier.

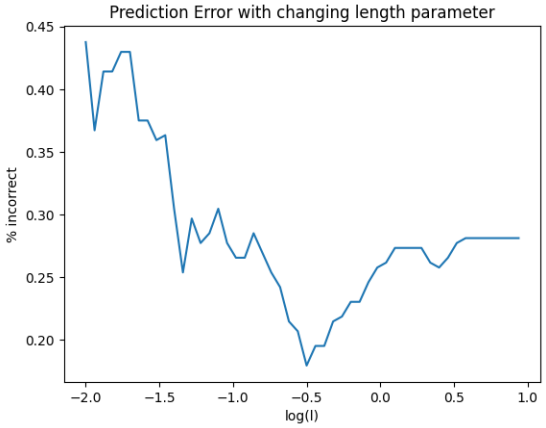

Figure 8: Evolution of prediction error with l

2 Part II - Spatial Data

2.1 Poisson Likelihood

We now consider discrete data in the form of bike thefts in Lewisham borough in 2015, split into equally sized cells throughout the borough - with each cell having a count, the data is shown in figure 9. We will sub-sample this data to give $\mathbf{c} = \{c_i\}_{i=0}^{M-1}$, and try to infer unseen counts c^* from the sub-sampled data $\mathbf{c}$.

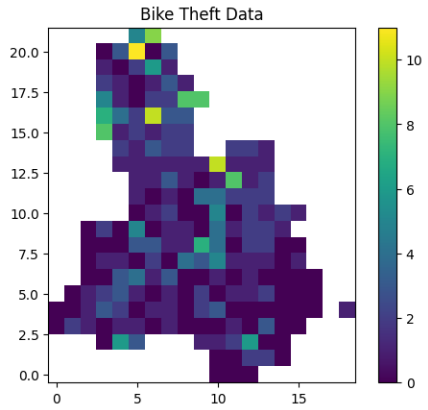

Figure 9: Bike Theft data from Lewisham in 2015

We can once again use a GP prior to model the correlation of bike thefts between adjacent regions, however a GP can take values in the range $(-\infty, \infty)$ and so we must apply an exponential function to ensure the values are positive. Thus our prior parameters are given by

$$\theta_i = \exp([G\mathbf{u}]_i) \text{ where } \mathbf{u}_i \sim \text{GP}(0, K) \tag{23}$$

We now need a discrete model for the data, a Poisson distribution is fitting as we are considering a count over a fixed time scale. Since we are using a Poisson distribution we can interpret our prior samples θ_i as rate parameters and our likelihood given the rate parameters is

$$p(\mathbf{c}|\boldsymbol{\theta}) = \prod_{i=0}^{M-1} \frac{e^{-\theta_i}\theta_i^{c_i}}{c_i!} \tag{24}$$

In order to perform pCN sampling we want to acquire the log-likelihood $\mathcal{L}(.) = \ln p(\mathbf{c}|\mathbf{u} = .)$, this is given by

$$\mathcal{L}(\mathbf{u}) = -\sum_{i=0}^{M-1} \exp(u_i) + \sum_{i=1}^{M} c_i u_i - \ln c_i! \tag{25}$$

However we can ignore the $\ln c_i!$ term as we have seen previously, because it has no dependence on $\mathbf{u}$. With the log-likelihood with respect to prior samples, we can sample from the posterior using $p(\mathbf{u}|\mathbf{c})$ using the pCN approach as before.

2.2 Expected Bike Thefts

We wish to infer the expected count for an unseen region c^* given our sub-sampled counts $\mathbf{c}$. This is given by

$$\mathbb{E}_{p(c^*|\mathbf{c})}[c^*] = \sum_{k=0}^{\infty} k p(c^* = k|\mathbf{c}) \tag{26}$$

We sum to infinity since the count in theory can take any value up to infinity with finite (but converging) probability. We know the predictive distribution $p(c^* = k|\mathbf{c})$, it is given by

$$p(c^* = k|\mathbf{c}) = \int p(c^* = k, \mathbf{u}|\mathbf{c}) d\mathbf{u} = \int p(c^* = k|\mathbf{u}) p(\mathbf{u}|\mathbf{c}) d\mathbf{u} \tag{27}$$

Since we have samples from the posterior $p(\mathbf{u}|\mathbf{c})$ we can compute predictive probabilities using a Monte-Carlo approximation of the form

$$p(c^* = k|\mathbf{c}) \approx \frac{1}{T} \sum_{i=0}^{T-1} p(c^* = k|\mathbf{u}^{(i)}) \tag{28}$$

Where $\mathbf{u}^{(i)}$ is the i-th sample from pCN. Substituting this back into equation 27 and some re-arranging gives

$$\begin{aligned}
\mathbb{E}_{p(c^*|\mathbf{c})}[c^*] &= \frac{1}{T}\sum_{i=0}^{T-1}\sum_{k=0}^{\infty} kp(c^* = k|\mathbf{u}^{(i)}) \\
&= \frac{1}{T}\sum_{i=0}^{T-1} \mathbb{E}_{p(c^*|\mathbf{u}^{(i)})}[c^*] \\
&= \frac{1}{T}\sum_{i=0}^{T-1} \theta^{*(i)} \\
&= \frac{1}{T}\sum_{i=0}^{T-1} \exp\left(u^{*(i)}\right)
\end{aligned} \tag{29}$$

Where we use the fact that the expectation of a Poisson random variable is its mean. Note that although we sub-sampled the data to produce $\mathbf{c}$, we can generate θ and u from anywhere, because this comes from the prior. The expectation allows us to infer counts from regions that are not part of our sub-sample $\mathbf{c}$, the only caveat here is that we do not know what length-scale to give to our prior, and so must investigate multiple values. We do this by computing the expected counts for a particular length scale and compare it to the true value, the results are shown in figure 10.

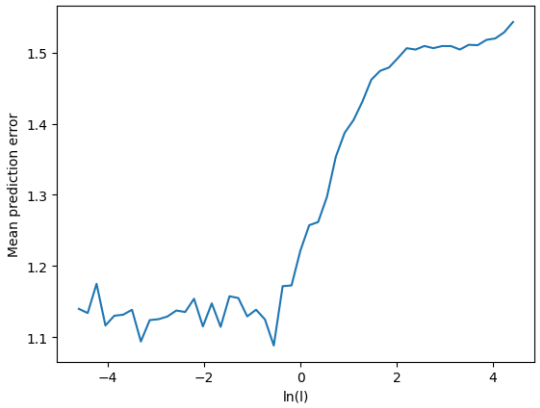

Figure 10: Predictive errors for different length scales

It turns out that $l \approx 0.575$ is the value that leads to the smallest prediction error, the predictions using this length scale are shown in figure 11, alongside the original data, the sub-samples, and the error field. Although this length scale has the smallest error, we see that there is only a correlation between areas with sub-samples and low error for the points themselves, and not with adjacent points. This is because the length scale is too small to allow adjacent points to be influenced by nearby ones, the whole point of using a GP prior was to model correlation between nearby regions and so it is possible that this assumption could be reconsidered.

Finally, it is of interest to run the model with extreme high and low length scale values, this is shown in figure 12, with an $l = 0.2$ plot for reference.
The $l = 0.01$ data looks somewhat similar to the $l = 0.575$ plot, with little correlation between nearby points. However the $l = 0.01$ data is not as good at inferring between sub-samples, because the length scale is so small the prior distribution is such that nearby points are near-independent. Seemingly this leads to a better prediction however, according to figure 10, this is further evidence that the correlation assumption was misguided. The large length scale $l = 100$ clearly does not fit the data very well, there is too much correlation between points and actually only has variation in one direction, this is because the prior GP has very few degrees of freedom.

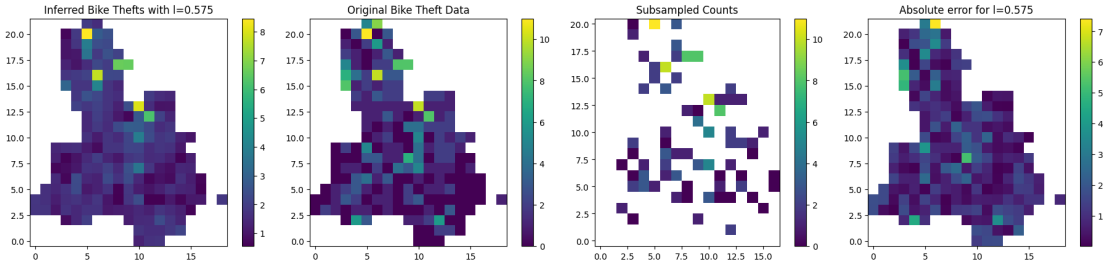

Figure 11: Predictive outputs, original data, sub-sampled data, and error field for $l = 0.575$

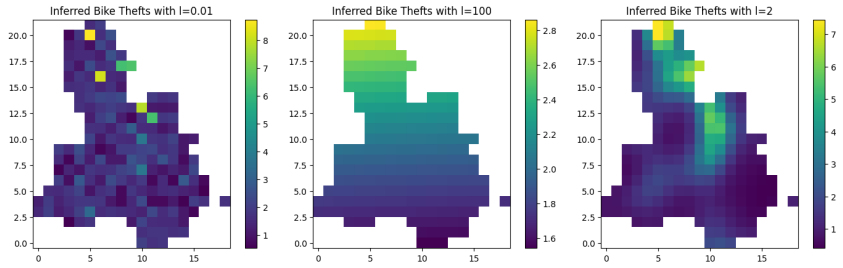

Figure 12: Predictive outputs for $l = 0.01$, $l = 100$, and $l = 2$

3 Conclusion

In this work we have concluded that although GRW-MH and pCN exhibit similar predictive performance when sampling from a posterior distribution, pCN exhibits much better acceptance rates: especially as the number of dimensions increases. We have concluded that we can infer well Gaussian Process values from noisy sub-samples using MCMC methods when we know the length parameter in advance, even when we limit the data to simply the probit classification of the noisy sub-samples. Even when we do not know the length parameter, we have shown that this can be inferred by minimising predictive error. We have also demonstrated the applications of this approach on real data, in inferring bike theft counts in Lewisham. We concluded that our initial assumption of correlation between nearby regions may not be as accurate as we first assumed, and that greater independence actually leads to better predictions.