4M24: Computational Statistics & Machine Learning High-Dimensional MCMC

Tommy Rochussen

December 2022

Abstract

For many models of practical interest, Bayesian inference is intractable and so we must we resort to approximation techniques. In this report we resort to Metropolis-Hastings (MH) Markov chain Monte Carlo (MCMC) to generate posterior samples for various applications. We begin with a toy example of a dataset generated from a Gaussian Process (GP) prior sample and, under a Gaussian likelihood, compare the Gaussian-random-walk (GRW) and preconditioned Crank-Nicolson (pCN) MH algorithms. After concluding that pCN is superior, we discard GRW, augment the toy example into a binary classification problem, and investigate two techniques for model selection. We then approach a real-world example and discuss model selection in an open-ended setting where hyperparameters are not known *a priori*.

1 Simulation

Task A

Given the covariance matrix of a Gaussian process $\mathbf{K}$, we can generate function samples as follows

$$f^{(i)} \sim \mathcal{N}(\mathbf{0}, \mathbf{K})$$

$$f^{(i)}(\mathbf{x}) = \mathbf{K}_c \boldsymbol{\epsilon}$$

where $\mathbf{K}_c$ is the Cholesky decomposition of the covariance matrix, $\mathbf{x}$ is an N-dimensional vector of function query locations corresponding to the $N \times N$ covariance matrix, and $\boldsymbol{\epsilon}$ is an N-dimensional vector of standard Gaussian random samples.

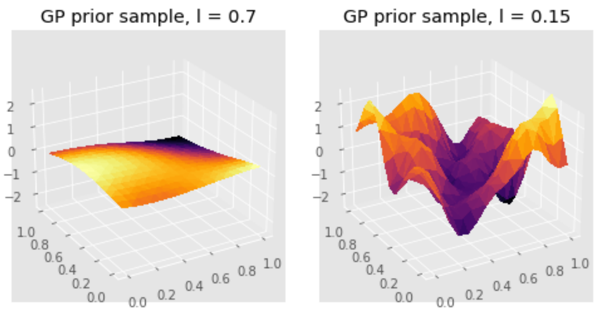

Figure 1: Prior samples from a GP with squared-exponential covariance function for high (left) and low (right) length-scale parameter values

Figure 1 Shows that the larger the length-scale parameter l, the larger the scale over which it takes the GP prior function samples to vary in value. The form of the covariance function

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

indicates that a larger l drives the covariance function value up towards $\kappa(\mathbf{x}, \mathbf{x}') = 1$ for any two inputs $\mathbf{x}$ and $\mathbf{x}'$. This means that, for a larger l, the GP function values for any two points have a higher covariance, and as such tend to be closer in value than for smaller l.

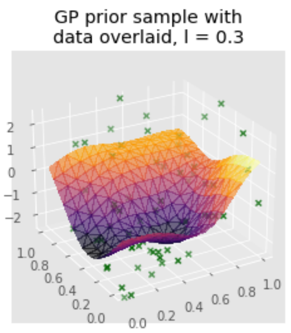

Figure 2: Gaussian Process prior sample function with subsampled and noisy function values overlaid

Figure 2 shows a GP prior function sample with noisy subsampled values overlaid. The simulated task is to infer the values of all N points on the $D \times D$ grid (i.e. $N = D^2$) from the noisy subsampled values—the dataset.

Task B

Under a GP prior and a Gaussian likelihood, the posterior distribution is tractable. However, we resort to MCMC methods since we intend to use likelihoods that result in intractable posteriors later on. Note that this report assumes knowledge of the pCN and GRW algorithms and so omits details of either algorithm. This is because they are detailed in the course lecture notes. As with any MCMC method, we must ensure that the obtained samples are both

1. Independent (a necessary assumption for any Monte Carlo simulation)

2. Identically distributed (from the posterior distribution)

Ensuring Sample Independence

Since our samples come from a Markov chain, and as such are subject to the Markov property, they are not independent. However, we can still obtain independent samples by applying *thinning* whereby we discard all but every t-th sample. Thinning is effective since Markov chains have *limited memory*, meaning that sample correlations decrease to zero as sample lag is increased.

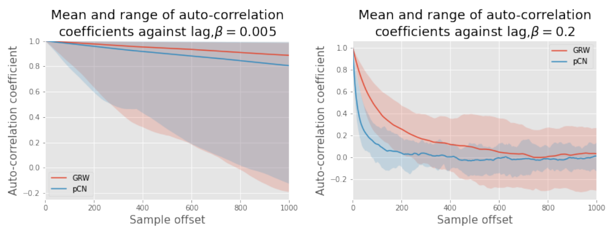

Figure 3: Mean and range of sample auto-correlation coefficients for each dimension of the parameter $\mathbf{u}$ against lag for low (left) and high (right) step-size parameter β

Figure 3 suggests that for $\beta = 0.2$ we take $t = 400$ for pCN MCMC and $t = 800$ for GRW MCMC, and this is what is done for the rest of the simulation section. For a very small step size the sample correlations remain high after many successive samples—this is because with such a small step size it takes *very* many samples to move to a different region of the posterior distribution. For a very large step size ($\beta \approx 1$) however, we might expect sample correlations to decay similarly to the small step-size case—this is because if the proposed samples are far away from the current proposal, they are likely to be in the tails of the posterior and so unlikely to be accepted (see Figure 6), and so the sequence of accepted samples could consist of long subsequences of repeated samples. Thinning leaves us with fewer usable samples and so the variance of any Monte Carlo estimate we make is increased, but we can run the MCMC simulations for longer to ensure that we still have sufficient samples.

Ensuring Chain Invariance

MCMC methods work by sampling from a Markov chain whose invariant distribution is the target distribution. In our case the target distribution is the posterior distribution over the model parameter $\mathbf{u}$, so we use the terms *invariant*, *target*, and *posterior* interchangeably. It takes time for the Markov chain conditional distribution to settle on the invariant distribution—a process known as convergence, and in general it is difficult to judge convergence since successive samples will continue to traverse the support of the invariant distribution after convergence.

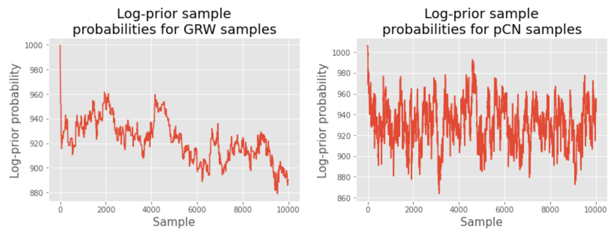

Figure 4: Sample log-prior probabilities for GRW (left) and pCN (right)

Since our initial sample is generated by sampling from the prior, one way we can judge convergence is by looking at the evolution of sample prior probabilities. Looking at Figure 4, we see significantly higher prior log probabilities for the first few samples than for the rest of the samples for both pCN and GRW, and so we will assume that convergence is satisfied after 500 samples as a safety margin for both pCN and GRW. For the rest of the section, we discard the first 500 samples—a procedure known as *burn-in*.

Performance of GRW & pCN

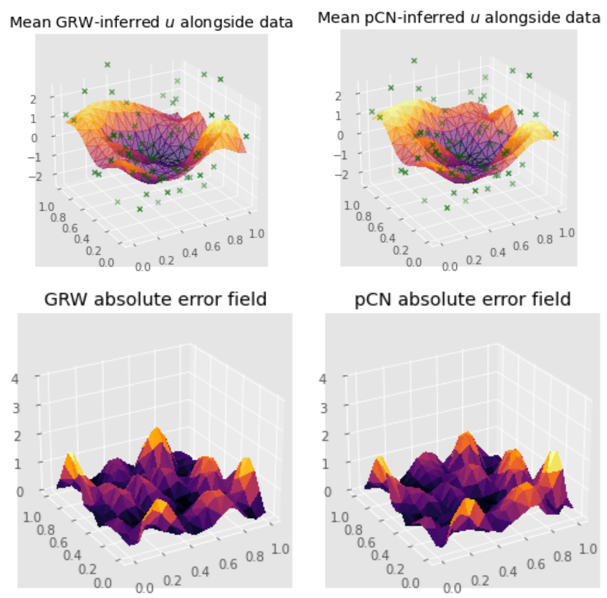

Figure 5: Mean of posterior samples for GRW (top left) and pCN (top right), as well as corresponding error fields (bottom left and right respectively)

The plots in Figure 5 show that the two MCMC methods produce very similar posterior means. The error fields are too similar to be able to say which method is the better performing of the two, however pCN is dramatically quicker than GRW—this opens to the door to performing useful but computationally costly procedures such as hyperparameter optimisation. The pCN algorithm is quicker because only the ratio of proposed sample likelihoods is required to calculate the acceptance probability—not the ratio of proposed sample non-normalised posteriors as is the case for GRW, and as such the computation of sample prior probability is avoided.

The best way to compare the performance of each flavour of MCMC would be to compare the full posterior distributions with the true posterior distribution (in this case an exact GP posterior), however this is difficult to do since there is no easy way to visualise and compare distributions in three dimensions, not least distributions for which we only have samples and thus no easy way to visualise a probability density hypersurface.

	GRW	pCN
$\bar{\alpha}$	0.0776	0.4518

Table 1: Average sample acceptance rate for 10,000 iteration MCMC runs on 25 random datasets with $l = 0.3$ and $\beta = 0.2$

Another helpful metric for comparing the two techniques is the average sample acceptance rate. This gives us an insight into how diverse the sequence of posterior samples are—a higher acceptance rate means that there are fewer repeated samples and thus there are more unique samples. This is desirable as it means the samples are likely to more evenly cover the support of the posterior distribution instead of existing in clusters of repeated samples across the distribution. A higher average acceptance rate also means that the support of the distribution is traversed more quickly with further samples. Table 1 shows us that, for $\beta = 0.2$, the acceptance rate under pCN is significantly higher on average than under GRW, and so is again preferable since it provides us with a more desirable sequence of samples.

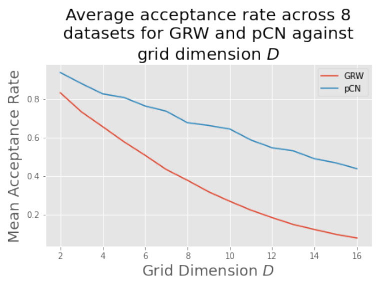

Figure 6: Average sample acceptance rates for 10,000 iteration MCMC runs on 8 random datasets with $l = 0.3$ against grid dimension D for $\beta = 0.2$

Looking at Figure 6, the acceptance rates for both algorithms decrease as the number of inferred dimensions increases, however the pCN acceptance rate is higher for all values of D and decreases at a slower rate. The acceptance rate decreases in this way for both algorithms since a larger grid dimension results in a greater number of degrees of freedom and so a sample generated by either proposal is less likely to be in a direction of either higher posterior (GRW) or likelihood (pCN) density, and so is not favoured by the respective acceptance probabilities—a case of *the curse of dimensionality*. Under pCN, only the M dimensions of a proposed sample $\mathbf{u}^{(i)}$ that correspond to the dimensions of $\mathbf{v}$ (datapoint locations) contribute to computing the likelihood for the acceptance probability, while for GRW all N dimensions of a proposed sample are needed to compute the non-normalised posterior for the acceptance probability. As such, when $N > M$ (more inferred points than datapoints), the above curse of dimensionality is more prevalent under GRW than pCN, and so the acceptance probability degenerates at a greater rate.

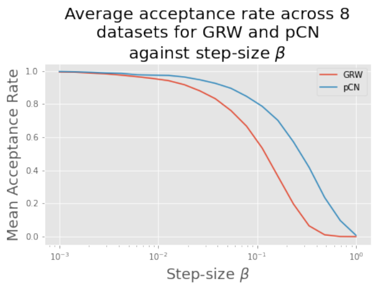

Figure 7: Average sample acceptance rates for 10,000 iteration MCMC runs on 8 random datasets with $l = 0.3$ against step-size parameter β for $D = 16$

Looking now at Figure 7, we see that the acceptance rate is non-increasing as the step-size β is increased. For a step-size of zero the newly proposed sample is the previous sample, which cannot lie in a region of lower posterior density than itself, and so is always accepted. For a very small step-size, the samples are almost all unique, but they are highly correlated as discussed under Figure 3, and successive samples traverse the support of the posterior distribution very slowly, and so *very* many are needed to accurately represent the full posterior distribution. Furthermore, a small step-size is likely to result in samples that are from one particular mode of the posterior, and both algorithms will struggle to generate samples from different modes if the distribution is multimodal. A larger step-size corresponds to proposed samples that are more likely to be in regions of lower posterior (GRW) or likelihood (pCN) density and so are not favoured by the corresponding acceptance probabilities, as explained in the latter half of the discussion of Figure 3. For a step-size of one, a newly proposed sample is a prior sample superposed on the previous sample under GRW, and a prior sample under pCN. This results in an acceptance rate of zero since the prior is wider than the posterior, and so a new sample under either algorithm is extremely unlikely to lie in a region of high posterior density.

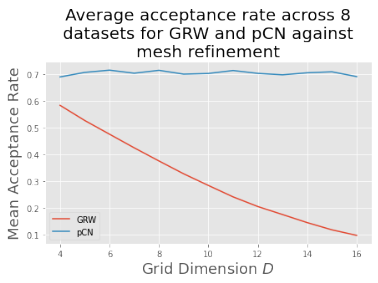

Figure 8: Average sample acceptance rates for 10,000 iteration MCMC runs on 8 random datasets with $l = 0.3$ as a function of mesh refinement

As mentioned in the discussion of Figure 6, the acceptance rate under pCN only degenerates if the number of datapoints M increases, while for GRW it also degenerates if the number of inferred points N increases. And so for a constant number of datapoints but increasing number of inferred points, as in Figure 8, we see that the acceptance rate is constant under pCN, but decreasing for GRW.

Task C

Given that the probit likelihood for a single binary datapoint t_i takes the form $p(t_i = 1|\mathbf{u}) = \Phi\left([\mathbf{Gu}]_i\right)$, and letting $\tilde{\mathbf{v}} = \mathbf{Gu}$, the full log-likelihood is derived as

$$\begin{aligned} p(t_i|\mathbf{u}) &= \Phi\left(\tilde{v}_i\right)^{t_i}\left(1 - \Phi\left(\tilde{v}_i\right)\right)^{1-t_i} \\ p(\mathbf{t}|\mathbf{u}) &= \prod_i \Phi\left(\tilde{v}_i\right)^{t_i}\left(1 - \Phi\left(\tilde{v}_i\right)\right)^{1-t_i} \\ \log p(\mathbf{t}|\mathbf{u}) &= \sum_i t_i \log \Phi\left(\tilde{v}_i\right) + (1 - t_i)\log\left(1 - \Phi\left(\tilde{v}_i\right)\right) \end{aligned}$$

Likewise, a Monte Carlo estimate of the predictive distribution is derived as

$$\begin{aligned} p(t_i^* = 1|\mathbf{t}) &= \int p(t_i^* = 1|\mathbf{u})p(\mathbf{u}|\mathbf{t})d\mathbf{u} \\ &= \mathbb{E}_{p(\mathbf{u}|\mathbf{t})} p(t_i^* = 1|\mathbf{u}) \\ &\simeq \frac{1}{n}\sum_{j=1}^{n} p\left(t_i^* = 1|\mathbf{u}^{(j)}\right), \qquad \mathbf{u}^{(j)} \sim p(\mathbf{u}|\mathbf{t}) \\ &= \frac{1}{n}\sum_{j=1}^{n} \Phi\left(v_i^{(j)}\right), \qquad \mathbf{v}^{(j)} = \mathbf{Gu}^{(j)} \end{aligned}$$

where n is the total number of independent posterior samples at our disposal.

In Figure 9, given how much the data has been corrupted by subsampling and noise, the extent to which the predictive distribution matches the unadulterated class assignments is highly impressive. An obvious desideratum for any model, especially a probabilistic one, is for predictions to capture the uncertainty of the model. In the top right plot we see that the probabilities of a point belonging to class 1 vary between roughly 0.8 and 0.2, indicating that there is no point for which the model is certain of

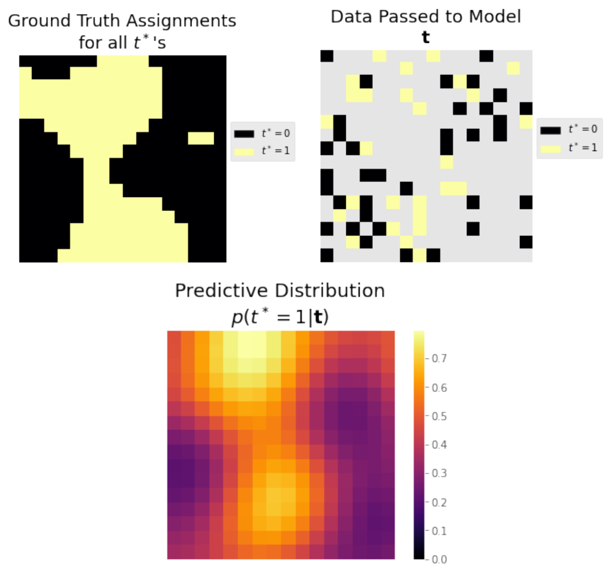

Figure 9: Original class assignments (top left), corrupted data supplied to the model (top right), predictive distribution via pCN posterior samples (bottom)

the class. Better still is the fact that for many points the probability is near to 0.5, indicating the model is not certain at all—this is particularly true at locations where lots of data is missing and where the class boundaries of the original assignments lie, reflecting how we as humans might make predictions given the corrupted data.

Task D

Optimising Prediction Error

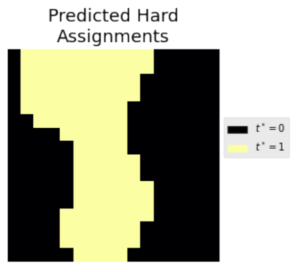

Figure 10: Predicted hard assignments

Applying a 0.5 threshold to the predicted probabilities of Figure 9, we acquire the hard class assignment predictions shown in Figure 10. In doing so, we relinquish the accurate uncertainty estimates of our probabilistic predictions, but there is now a straightforward way to quantify model performance—computing a mean prediction error against the original class assignments. This is useful because it provides us with a framework for model selection, which is necessary since in all but the most contrived of applications model hyperparameters such as the prior length-scale l would not be known *a priori*. We can then continue to use the full predictive distribution of the model that minimises mean prediction error.

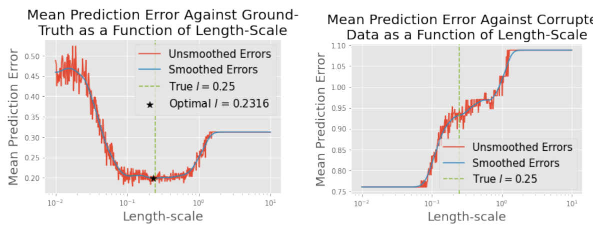

Figure 11: Mean predicted error for models with varying prior length-scale parameter l on original data (left) and corrupted data (right)

Since error plots tend to by noisy, it is useful to apply Gaussian smoothing with a suitable scale and select the model that minimises the smoothed error curve. The left-hand plot in Figure 11 shows that if we have access to the original assignments, then we can recover a good estimate of the true length scale parameter l. This is very well, but this approach assumes that we have access to the untarnished data, and so by extension that the only reason we corrupted the dataset in the first place was to carry out this cross-validation-esque procedure for model selection. If that is the case, then we would surely improve model performance by applying a less substantial degree of subsampling and refraining from adding noise—it would be better to supply the model with more of the original data during training (computation of the posterior distribution) and use a smaller amount of held-out data on which to validate.

It is perhaps more realistic to assume that we do not have access to the original assignments, but rather only the noisy and incomplete dataset. The task of inferring all class assignments then becomes a more useful, difficult, and indeed interesting one. As seen in the right-hand plot of Figure 11, the mean prediction error against the corrupted dataset provides an ineffective metric for model selection—the model that performs best under this metric is one with a small enough length-scale to capture the noise in the data and therefore make predictions that overfit to the training data. The reason for this is that the model *sees* all of the data on which it is validated during training, and as such the validation is no longer a good measure of how well the model generalises to unseen data. Instead, the validation is a measure of how well the model can fit to the dataset—in essence it is a similar procedure to model selection by maximising the likelihood, hence why it encourages overfitting. If we only have access to the corrupted dataset but would like to carry out a simple cross-validation-esque procedure for model selection, then we would need to further subsample the dataset so that we have a training dataset and a *validation* dataset whose union is the entire corrupted dataset.

Optimising Marginal Likelihood

For a Bayesian, the obvious approach to model selection is the *evidence framework*—choose the model that maximises the marginal likelihood (also termed the model evidence). For the uninitiated, the marginal likelihood is the name of the denominator in *Bayes' Theorem*, which, in the context of machine learning where we have a model $\mathcal{M}$ with parameters θ and a dataset $\mathcal{D}$, can be written

$$p(\theta|\mathcal{D},\mathcal{M}) = \frac{p(\theta|\mathcal{M})p(\mathcal{D}|\theta,\mathcal{M})}{p(\mathcal{D}|\mathcal{M})}$$

Note that usually the dependence on the choice of model $\cdot|\mathcal{M}$ is omitted for brevity of notation. The marginal likelihood is the normalising constant for the product of the prior and likelihood (the "non-normalised" posterior), and is computed as

$$p(\mathcal{D}|\mathcal{M}) = \int p(\theta|\mathcal{M})p(\mathcal{D}|\theta,\mathcal{M})d\theta$$

It represents the probability of a dataset occurring assuming it was generated by a particular class of model, irrespective of model parameters. In general it is intractable and so is why posterior distributions tend to be intractable too. From now on we omit the dependence on the model class $\mathcal{M}$. One way we can estimate it is by applying a simple Monte Carlo estimator derived as

$$\begin{aligned} p(\mathcal{D}) &= \int p(\theta)p(\mathcal{D}|\theta)d\theta \\ p(\mathcal{D}) &= \mathbb{E}_{p(\theta)}p(\mathcal{D}|\theta) \\ &\simeq \frac{1}{n}\sum_{i=1}^{n} p(\mathcal{D}|\theta^{(i)}), \qquad \theta^{(i)} \sim p(\theta) \end{aligned}$$

This Monte Carlo estimator for the marginal likelihood is usually described as the *naive* Monte Carlo or *arithmetic mean* estimator (Llorente et al., 2020). For us, the marginal likelihood is estimated as

$$p(\mathcal{D}) \simeq \frac{1}{n}\sum_{i=1}^{n} p(\mathbf{t}|\mathbf{u}^{(i)}), \qquad \mathbf{u}^{(i)} \sim p(\mathbf{u})$$

where n is the number of prior samples we take. To implement this in a numerically stable way, we need to move into the log domain and utilise the *log-sum-exp* trick as follows

$$\log p(\mathcal{D}) \simeq \mathrm{LSE}\{\ell(\mathbf{t}|\mathbf{u}^{(i)})\}_{i=1}^{n} - \log n$$

where LSE denotes the log-sum-exp operator and $\ell(\mathbf{t}|\mathbf{u})$ denotes the log of the probit likelihood.

Figure 12 shows that the naive Monte Carlo estimate to the marginal likelihood can be used effectively to select a prior length-scale that well-estimates the true one, especially after Gaussian smoothing. Importantly, this provides us with a principled approach to selecting model hyperparameters that works well under the assumption that we only have access to the corrupted data, as is the case for Figure 12. To achieve lower-variance estimates for the marginal likelihood, *Chib's method* (Chib and Jeliazkov, 2001) is perhaps the most suitable for this task.

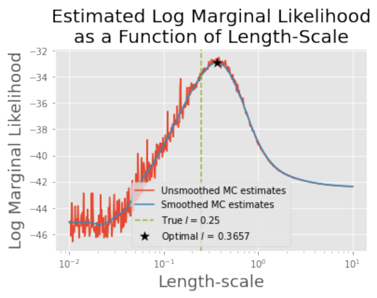

Figure 12: Log marginal likelihood for different values of prior length-scale parameter l

Spatial Data

Task E

We now let $\boldsymbol{\theta}$ denote the transform of the subsampled data such that:

$$\theta_i = \exp\left([\mathbf{G}\mathbf{u}]_i\right) = \exp(\tilde{v}_i)$$

and we interpret it as the Poisson rate parameter corresponding to a count in our dataset. The full Poisson likelihood $p(\mathbf{c}|\mathbf{u})$ for bike theft count data $\mathbf{c}$ is derived as

$$\begin{aligned}\log p(\mathbf{c}|\boldsymbol{\theta}) &= \sum_{i=1}^{M} \log \frac{e^{-\theta_i}\theta_i^{c_i}}{c_i!}\\ &= \sum_{i=1}^{M} -\theta_i + c_i \log\theta_i - \log c_i!\\ \therefore \quad \log p(\mathbf{c}|\mathbf{u}) &= \sum_{i=1}^{M} -\exp(\tilde{v}_i) + c_i\tilde{v}_i - \log c_i!\end{aligned}$$

Note that only the difference of sample likelihoods is used in the pCN algorithm, and so for added efficiency we can omit the log factorial term from the likelihood since it would only cancel. We can then proceed with pCN as before to generate posterior samples, but first we need to investigate the burn-in time and required degree of thinning since we are working with a very different dataset.

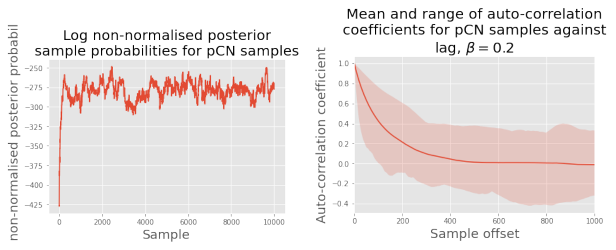

Figure 13: Sample log non-normalised posterior probabilities (left), mean and range of sample auto-correlation coefficients for each dimension of the parameter $\mathbf{u}$ against lag (right)

This time we use the log non-normalised posterior sample probabilities to investigate the initial transient period since they happen to show the transition more clearly. The left-hand plot in Figure 13 shows that the initial transient period is only around 200 samples long, as so we discard the first 500 samples. The right-hand plot indicates that t should be taken to be 500. These values are used for the rest of this section, and the total number of pCN iterations is increased from 10,000 to 500,000 to ensure that there are enough thinned and burned in samples

Task F

Predictive Distribution Derivations & Initial Predictions

The predictive distribution $p(c^* = k|\mathbf{c})$ over the true underlying bike-theft count values is intractable. However, we can estimate the expectation of the distribution using a Monte Carlo estimate derived as

$$\begin{aligned}p(c^* = k|\mathbf{c}) &= \int p(c^* = k|\mathbf{u})p(\mathbf{u}|\mathbf{c})d\mathbf{u}\\ &\simeq \frac{1}{n}\sum_{j=1}^{n} p(c^* = k|\mathbf{u}^{(j)}),\ \mathbf{u}^{(j)} \sim p(\mathbf{u}|\mathbf{c})\\ \therefore \quad \mathbb{E}_{p(c^*|\mathbf{c})}(c^*) &\simeq \sum_{k=0}^{\infty} k \left(\frac{1}{n}\sum_{j=1}^{n} f\left(c^* = k|\theta^{*(j)}\right)\right)\\ &= \frac{1}{n}\sum_{j=1}^{n}\sum_{k=0}^{\infty} k f\left(c^* = k|\theta^{*(j)}\right)\\ &= \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}_{f\left(c^*|\theta^{*(j)}\right)}(c^*)\\ &= \frac{1}{n}\sum_{j=1}^{n} \theta^{*(j)})\end{aligned}$$

where $f(c|\theta)$ is the Poisson distribution $\frac{e^{-\theta}\theta^c}{c!}$.

With no information about what the prior length-scale l should be, we initially use a length-scale of $l = 1.0$. In Figure 14 we see that the mean predictions look fairly different to the original dataset. Though areas which generally have higher or lower theft counts in the original dataset can still be identified in the mean predicted counts, the count values of the latter tend to vary more gradually—adjacent cells correlate more strongly than in the original dataset. To reason about models with different length-scales, we need a way to investigate the full predictive distribution rather than just the expectation thereof. We can estimate the predictive distribution $p(c^* = k|\mathbf{c})$ for a true underlying count c^* using

$$\begin{aligned}p(c^* = k|\mathbf{c}) &\simeq \frac{1}{n}\sum_{j=1}^{n} p(c^* = k|\mathbf{u}^{(j)}), \qquad \mathbf{u}^{(j)} \sim p(\mathbf{u}|\mathbf{c})\\ &= \exp\left(\text{LSE}\{\ell(c^* = k|\mathbf{u}^{(i)})\}_{j=1}^{n} - \log n\right)\\ &= \hat{p}(c^* = k|\mathbf{c})\end{aligned}$$

meaning we can estimate the variance of the predictive distribution over counts as follows:

$$\begin{aligned}
\mathbb{V}_{p(c^*|\mathbf{c})}\left[c^*\right] &= \mathbb{E}_{p(c^*|\mathbf{c})}\left[{c^*}^2\right] - \mathbb{E}_{p(c^*|\mathbf{c})}\left[c^*\right]^2 \\
&= \sum_{k=0}^{\infty} k^2 p(c^* = k|\mathbf{c}) - \left(\sum_{k=0}^{\infty} k p(c^* = k|\mathbf{c})\right)^2 \\
&\simeq \sum_{k=0}^{\infty} k^2 \hat{p}(c^* = k|\mathbf{c}) - \left(\sum_{k=0}^{\infty} k \hat{p}(c^* = k|\mathbf{c})\right)^2 \\
&\simeq \sum_{k=0}^{d} k^2 \hat{p}(c^* = k|\mathbf{c}) - \left(\sum_{k=0}^{d} k \hat{p}(c^* = k|\mathbf{c})\right)^2
\end{aligned}$$

where d is some integer for which $p(c^* = d + z|\mathbf{c}) \simeq 0$, $\forall z \in \mathbb{Z}^+$. The standard deviation is then just the square-root of the variance.

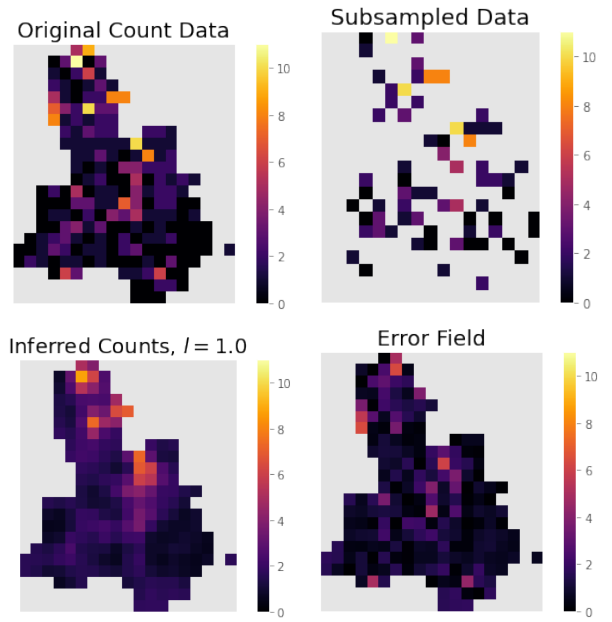

Figure 14: Original count data (top left), subsampled data (top right), the mean inferred counts under $l = 1.0$ (bottom left), the absolute difference between the original counts and the mean inferred counts (bottom right)

Extreme Prior Length-Scales

Looking at Figure 15, for a very large prior length-scale we see that the mean predicted counts vary *very* gradually with the only significant variation being a slight increase in theft counts towards the north of the borough. The standard deviation of the distribution is similarly smooth and increases in a similar manner to reflect an increase in uncertainty for more extreme predictions. The model attributes any finer variation in theft counts, which in this case is almost all of the variation, to randomness—a hallmark of underfitting. Under a very small length-scale the model behaves differently for cells that were in the training dataset and those that were not. For cells for which the model has seen data, the mean predictions match the data almost exactly and the standard deviation is small, but again increases for larger mean inferred counts. The mean predictions in the rest of the cells are mostly the same as each other at some reasonable value ($\simeq 1.64$) which is not too different from the dataset mean ($= 1.63$). The predictive standard deviation for these points is also roughly uniform, but noticeably larger than for the points for which the model has seen data. This is because, under a small length-scale, the model assumes very little covariance between locations, and as such the predictions for each location are almost entirely independent—predictions for cells for which there is data are centered on the corresponding datapoint value with generally low uncertainty (except for extreme values which have higher uncertainty), while for the rest of the cells the model reverts to the prior. The model overfits to the data, hence the overconfidence for predictions on points which had data and lack of confidence otherwise.

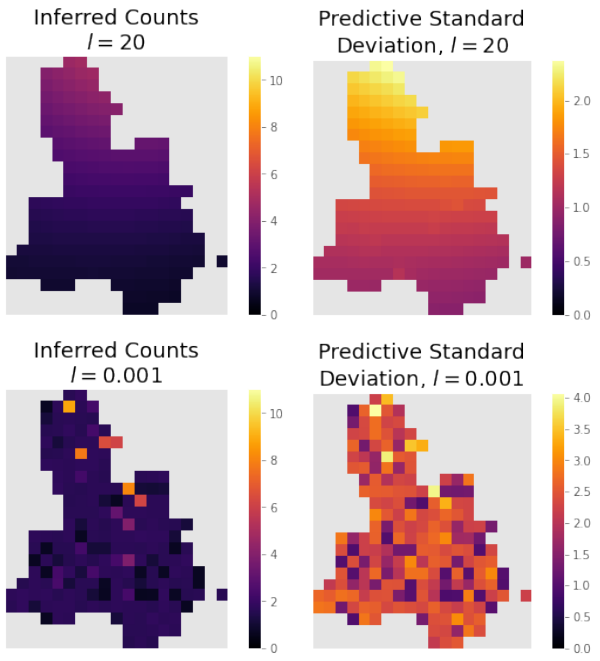
Inferred Counts $l = 20$
Predictive Standard Deviation, $l = 20$
Inferred Counts $l = 0.001$
Predictive Standard Deviation, $l = 0.001$
0
2
4
6
8
10
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

Figure 15: Mean inferred counts for $l = 20$ (top left), standard deviation of predictive distribution for $l = 20$ (top right), mean inferred counts for $l = 0.001$ (bottom left), standard deviation of predictive distribution for $l = 0.001$ (bottom right)

Hyperparameter Selection

As in the previous section, the natural next step is to try and find the most suitable prior length-scale. In an attempt to do this we use the same two pieces of machinery that the previous section armed us with; the naive cross-validation-esque method, which we term the *hold-out* method, and the evidence framework. In order not to lose generality to other datasets and applications, we will assume that we only have access to the incomplete dataset. This is done because for both model selection techniques we use, it is the case that if they work on the incomplete dataset then they also work on the complete dataset, but not necessarily the other way round. For the rest of this section, *the dataset* refers to the subsampled dataset.

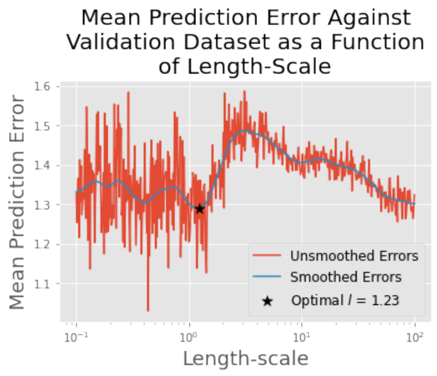

Figure 16: Mean prediction errors against validation set for varying prior length-scale parameter l

After randomly splitting the dataset into a training set and a validation set, in this case in a 70 : 30 proportion, we can compute the mean prediction error against the validation set for models with different prior length-scales trained on the training dataset. Since the validation dataset is relatively small (only 27 count predictions), the mean prediction errors in Figure 16 are noisier than we have seen previously, but it is still possible to recover a meaningful minimum and corresponding optimal length-scale. Note that the holdout method in this section is slightly different to that of the previous section since here we only validate on unseen data, but previously we validated on a dataset consisting either entirely or in some part of the training data.

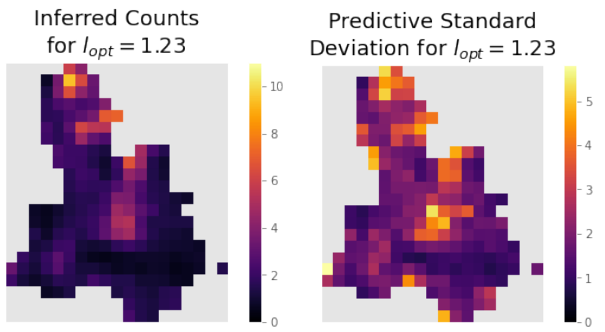
Inferred Counts for $l_{opt} = 1.23$
Predictive Standard Deviation for $l_{opt} = 1.23$

Figure 17: Mean inferred counts and predictive standard deviation for the prior length-scale that is optimal under the holdout method

The holdout-method-optimal (HMO) length-scale is a fairly moderate value, and we see that the predictions capture variation as granular as neighbourhoods of around four cells of higher or lower bike thefts, but attribute any finer variation to randomness. The predictive standard deviation values are also moderate, but large enough to account for any finer variation. The predictive standard deviation values are again larger for more extreme predictions to reflect the increase in uncertainty, and they do not vary particularly smoothly.

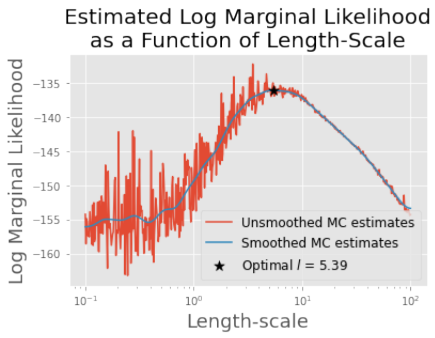

Figure 18: Log marginal likelihood for different values of prior length-scale parameter l

Exactly as in the previous section, we compute estimates for the marginal likelihood using a naive Monte Carlo method and do this for models with different prior length-scales. This provides us with an evidence-framework-optimal (EFO) length-scale.

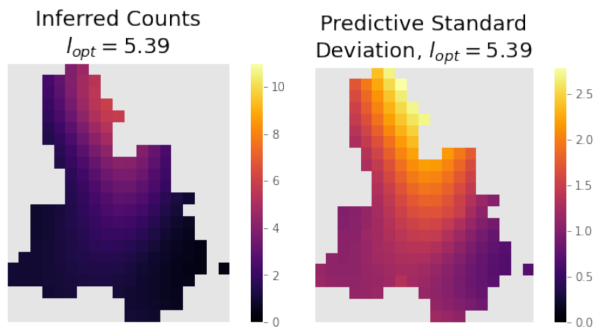
Inferred Counts $l_{opt} = 5.39$
Predictive Standard Deviation, $l_{opt} = 5.39$

Figure 19: Mean inferred counts and predictive standard deviation for the prior length-scale that is optimal under the evidence framework

The EFO length-scale is perhaps surprisingly large, with predictions capturing variation over the scale of approximately half the size of the borough at minimum. The model identifies the northernmost-central region in the borough as a region with high theft rates, while the rest of the borough has theft rates that decrease smoothly and monotonically with distance away from that region. The predictive standard deviations vary similarly smoothly, are relatively low, but increase to reflect an increase in uncertainty for more extreme predictions as with previous cases.

For this dataset, the two model selection methods yield drastically different optimal hyperparameter values. To help reach a conclusion over which method is "right", it is useful to consider:

1. the purpose of our model predictions

2. biases and pathologies present in each model selection technique

It is important to remember that the purpose of our model is to produce a distribution over the underlying Poisson rate parameters for bike thefts in a given $400m^2$ cell. Crucially, this is different to asking the model to give us a distribution over what the bike theft count for a given cell might be after a single trial. We expect the predictions to have a tighter confidence region than if the latter were the case and so bike theft counts collected from a single trial do not necessarily need to lie within the confidence bounds of a prediction. As such, we might expect to see predictions that underfit the dataset because the underlying parameters should have significantly less random variation, and it is possible for much of the variation in the dataset to be attributed to randomness, as is the case for the EFO model

Another important consideration is that our model is probabilistic, but the holdout method is not—it only uses the predictive mean. Ultimately we wish to use the full predictive distribution of the model, and so it is perhaps wrong to assess models purely on how well their predictive mean generalises to new data. This is in contrast to the fact that the evidence framework is the canonical probabilistic approach to model selection for compelling reasons such as the natural trading off between encouraging a good fit to the data and penalising model complexity (Bishop, 2006). Another weakness of the holdout method is that it could be sensitive to the choice of validation set—if the randomly selected validation points happen to be mostly close or even adjacent to points in the training data then the optimal model will show more signs of overfitting and fewer signs of generalising well. On the other hand, Lotfi et al. (2022) argue that the marginal likelihood has a tendency to favour models that underfit, which could be the case here. A final thing to consider is that the choice of prior and indeed class of model might not be the most suitable for this application, and this could degrade the ability of the marginal likelihood to select the best model. To further investigate this notion, one needs to question whether they really expect bike thefts to vary as smoothly as a GP with squared-exponential covariance would dictate—would natural obstacles in the borough such as motorways or larger buildings not contribute towards a strongly discontinuous field of bike theft Poission rates?

References

Fernando Llorente, Luca Martino, David Delgado, and Javier Lopez-Santiago. Marginal likelihood computation for model selection and hypothesis testing: an extensive review. 2020. doi: 10.48550/ARXIV.2005.08334. URL https://arxiv.org/abs/2005.08334.

Siddhartha Chib and Ivan Jeliazkov. Marginal likelihood from the metropolis–hastings output. *Journal of the American Statistical Association*, 96(453):270–281, 2001. doi: 10.1198/016214501750332848. URL https://doi.org/10.1198/016214501750332848.

Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

Sanae Lotfi, Pavel Izmailov, Gregory Benton, Micah Goldblum, and Andrew Gordon Wilson. Bayesian model selection, the marginal likelihood, and generalization, 2022. URL https://arxiv.org/abs/2202.11678.