CUED - Engineering Tripos Part IIB 2019-2020 **Module Coursework**

Module	**4M24**	Title of report	**High-Dimensional MCMC**
Date submitted: **10/12/2020**		Assessment for this module is ☐ 100% / ☑ **25% coursework** of which this assignment forms **100** %	

UNDERGRADUATE STUDENTS O ɴLY			**POST GRADUATE STUDENTS ONLY**			
Candidate number:	**5562E**		Name:		College:	

Feedback to the student

☐ **See also comments in the text**

		Very good	**Good**	Needs improvmt
CONTENT	**Completeness, quantity of content:** Has the report covered all aspects of the lab? Has the analysis been carried out thoroughly?			
	Correctness, quality of content Is the data correct? Is the analysis of the data correct? Are the conclusions correct?			
	Depth of understanding, quality of discussion Does the report show a good technical understanding? Have all the relevant conclusions been drawn?			
	Comments:			
PRESENTATION	**Attention to detail, typesetting and typographical errors** Is the report free of typographical errors? Are the figures/tables/references presented professionally?			
	Comments:			

Overall assessment (circle grade)		A*	**A**	B	C	D
Guideline standard		>75%	**65-75%**	55-65%	40-55%	<40%
Penalty for lateness:		*20% of marks per week or part week that the work is late.*				

Marker: Date:

4M24 CW - High-Dimensional MCMC

Candidate: 5562E

January 10, 2021

Abstract

This report outlines the result of the 4M24 coursework on high-dimensional Markov Chain Monte Carlo (MCMC).

Contents

1 Introduction

2 Simulation

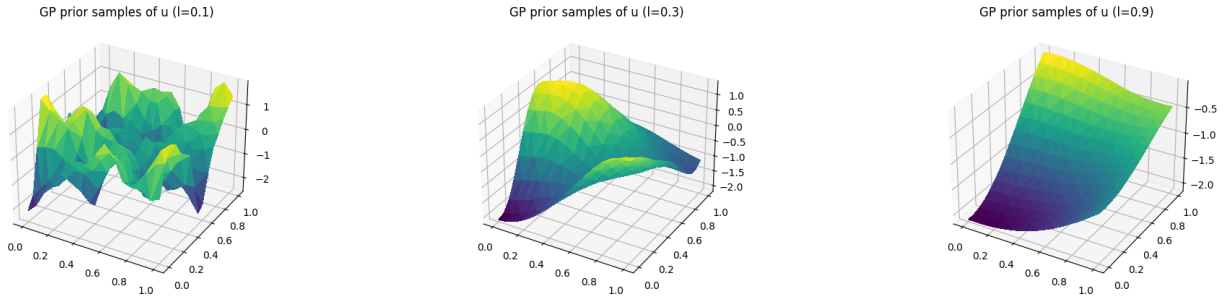
a Sampling from a Gaussian Process

We begin with a Gaussian Process (GP) defined on a 2D domain $\boldsymbol{x} \in [0,1]^2$. The realisations from this process are denoted $\boldsymbol{u} \sim \mathcal{N}(0, C)$ where $C_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and k is the Squared Exponential (SE) covariance function with length parameter l:

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

If we specify the latent variables $\{\boldsymbol{x}_i\}_{i=1}^N$, then we can compute C and hence fully specify the prior on u. We choose to place $\{x_i\}_{i=1}^N$ on a $D \times D$ grid with equal spacing, starting at $(0,0)$ and ending at $(1,1)$. Obviously, we require $N = D^2$.

We can now plot the u-surface atop this grid by ensuring that for each $i \in \{1 \dots N\}$, u_i denotes the Z-position and $\boldsymbol{x}_i$ the X-Y-position. We can then investigate the effect of varying the length-scale parameter l. Three settings of l and the associated plots are given in figure 1 for $D = 16$.



(a) Small $l = 0.1$ (b) Medium $l = 0.3$ (c) Large $l = 0.9$

Figure 1: Samples from GP prior for varying length scales ($D = 16$)

We now proceed to make M random draws (denoted by the $M \times 1$ vector $\boldsymbol{v}$) from these samples $\boldsymbol{u}$ with additive Gaussian noise $\boldsymbol{\epsilon} \sim \mathcal{N}(0, I)$. The subsampling factor f is defined as $f \coloneqq N/M$. The draws can be computed as follows:

$$\boldsymbol{v} = G\boldsymbol{u} + \boldsymbol{\epsilon} \tag{2}$$

Where G is an $M \times N$ matrix with a single one in each row in a random location (without repetition) and rest zeros. The result is that the observations $\boldsymbol{v}$ are a jumbled subsample of $\boldsymbol{u}$ with additive noise $\boldsymbol{\epsilon}$. We can plot the data overlaid on the original prior samples by simply matching each entry of $\boldsymbol{v}$ back to the coordinate it was selected from. The result is plotted on figure 2.

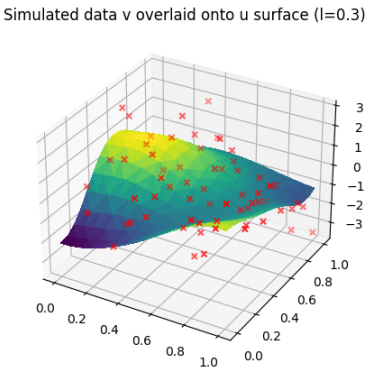


Figure 2: $\boldsymbol{v}$ (red crosses) overlaid on $\boldsymbol{u}$-surface ($f = 4, D = 16, l = 0.3$)

We observe $M = N/f = 16^2/4 = 64$ samples contained in the $\boldsymbol{v}$ vector. These are equally likely to appear above or below the $\boldsymbol{u}$-surface as the noise has zero mean. The noise variance for each data-point is of similar magnitude to the variation in

the surface ($\sigma^2 = 1$) so some crosses appear relatively far away from the surface. Moreover, as the subsampling is random, the $\boldsymbol{v}$-points appear at randomly chosen (but distinct) locations X-Y plane.

b Log probabilities and MCMC

We assume that we have realised a set of observations $\boldsymbol{v}$ and it is now our job to determine probability distributions for $\boldsymbol{u}$ based on this information. As a matter of notation, we define the prior function $\pi(\cdot) \coloneqq p(\boldsymbol{u} = \cdot)$ and likelihood function $\lambda(\cdot) \coloneqq \ln p(\boldsymbol{v}|\boldsymbol{u} = \cdot)$. Likewise, we define the posterior $\rho(\cdot) \coloneqq p(\boldsymbol{u} = \cdot|\boldsymbol{v})$.

The log prior can be calculated simply, through manipulation of the Gaussian pdf:

$$\begin{aligned}\ln \pi(\boldsymbol{w}) &= \ln \mathcal{N}(\boldsymbol{w}; 0, C) \\ &= \ln \frac{1}{(2\pi)^{N/2}|C|^{1/2}} \exp\left(-\frac{1}{2}\boldsymbol{w}^T C^{-1}\boldsymbol{w}\right) \\ &= -\left(\frac{N}{2}\ln 2\pi + \frac{1}{2}\ln|C| + \frac{1}{2}\boldsymbol{w}^T C^{-1}\boldsymbol{w}\right)\end{aligned} \tag{3}$$

Likewise, $\boldsymbol{v}|\boldsymbol{u}$ is also a Gaussian such that $\boldsymbol{v}|\boldsymbol{u} \sim \mathcal{N}(G\boldsymbol{u}, I)$ (see equation 2). By comparison with the form of equation 3, we can jump straight to the log-likelihood, noting that $\ln|I| = 0$:

$$\ln \lambda(\boldsymbol{w}) = -\left(\frac{M}{2}\ln 2\pi + \frac{1}{2}(\boldsymbol{v} - G\boldsymbol{w})^T(\boldsymbol{v} - G\boldsymbol{w})\right) \tag{4}$$

From these it is trivial to determine the log-posterior from Bayes' rule:

$$\begin{aligned}\rho(\boldsymbol{w}) &\propto \pi(\boldsymbol{w}) \cdot \lambda(\boldsymbol{w}) \\ \therefore \ln \rho(\boldsymbol{w}) &= \ln \pi(\boldsymbol{w}) + \ln \lambda(\boldsymbol{w}) + \text{const}\end{aligned} \tag{5}$$

It is important to note that, $\boldsymbol{w}$ is simply a dummy variable. We can simplify this notation further by defining $\mathcal{P} \coloneqq \ln \rho + \text{const}$, $\Pi \coloneqq \ln \pi$ and $\Lambda \coloneqq \ln \lambda$. The constant can be chosen to give us:

$$\mathcal{P}(\boldsymbol{\omega}) = \Pi(\boldsymbol{\omega}) + \Lambda(\boldsymbol{\omega}) \tag{6}$$

b.1 Gaussian Random Walk - Metropolis Hastings (GRW-MH)

Armed with the log-posterior (equation 6), observations $\boldsymbol{v}$ and observation matrix G, we can now apply the Gaussian Random Walk Metropolis-Hastings algorithm to draw samples of $\boldsymbol{u}$ from the posterior. We start with an initial estimate drawn from the prior $\boldsymbol{u}^{(0)} \sim p(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{u}; 0, C)$. For simplicity we use the symbol $\boldsymbol{\zeta}^*$ to denote a fresh sample drawn from the prior $\pi \sim \mathcal{N}(0, C)$. It is computed by applying a Cholesky decomposition to the covariance matrix C and multiplying a standard Gaussian vector $\boldsymbol{z} \sim \mathcal{N}(0, I)$. As such:

$$\boldsymbol{\zeta}^* = C^{1/2}\boldsymbol{z} \tag{7}$$

To emphasise, every time $\boldsymbol{\zeta}^*$ appears in an equation we draw a fresh sample according to equation 7. We then pick a symmetric proposal distribution to sequentially generate samples. Given a sample $\boldsymbol{u}^t$ we generate a proposal $\boldsymbol{w}^{(t)}$ as follows:

$$\boldsymbol{w}^{(t)} \coloneqq \boldsymbol{u}^{(t)} + \beta\boldsymbol{\zeta}^* \tag{8}$$

Where β is a hyperparameter that controls the step-size of each iteration. As our proposal distribution is symmetric, the acceptance probability $\alpha^{(t)} \coloneqq \alpha(\boldsymbol{u}^{(t)}, \boldsymbol{w}^{(t)})$ simplifies to the ratio of posteriors (with an upper bound of 1):

$$\alpha(\boldsymbol{u}, \boldsymbol{w}) \coloneqq \min\left(\frac{\rho(\boldsymbol{w})}{\rho(\boldsymbol{u})}, 1\right) \tag{9}$$

It may be more natural to deal in log of this value:

$$\ln \alpha^{(t)} = \min\left(\mathcal{P}(\boldsymbol{w}^{(t)}) - \mathcal{P}(\boldsymbol{u}^{(t)}), 0\right) \tag{10}$$

Note that the constant term in the $\mathcal{P}$ definition cancels. Naturally, we can sample a uniform random variable $U \sim \mathcal{U}(0, 1)$ and compare to $\alpha^{(t)}$:

$$\begin{aligned}p(U < \alpha) &= \alpha \\ \therefore p(\ln U < \ln \alpha) &= \alpha\end{aligned} \tag{11}$$

Algorithm 1 Gaussian Random Walk - Metropolis Hastings

$\boldsymbol{u}^{(0)} \leftarrow \boldsymbol{\zeta}^*$
for $t \in \{0, 1 \cdots T-1\}$ **do**
  $\boldsymbol{w}^{(t)} \leftarrow \boldsymbol{u}^{(t)} + \beta\boldsymbol{\zeta}^*$ ▷ Generate proposal
  $\ln \alpha^{(t)} \leftarrow \min\left(\mathcal{P}(\boldsymbol{w}^{(t)}) - \mathcal{P}(\boldsymbol{u}^{(t)}), 0\right)$
  $U^{(t)} \leftarrow\sim \mathcal{U}(0,1)$

  if $\ln U^{(t)} \leq \ln \alpha^{(t)}$ **then**
    $\boldsymbol{u}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)}$ ▷ Proposal accepted
  else
    $\boldsymbol{u}^{(t+1)} \leftarrow \boldsymbol{u}^{(t)}$ ▷ Proposal rejected
  end if
end for

The algorithm for GRW-MH is given in algorithm 1.

As our algorithm depends only on the difference of the posteriors, there is no need to compute the constant term. This massively speeds up computation.

b.2 Preconditioned Crank-Nicholson (pCN)

Preconditioned Crank-Nicholson is rather similar, except we choose a subtly different proposal distribution:

$$\boldsymbol{\omega}^{(t)} \coloneqq \sqrt{1-\beta^2}\boldsymbol{u}^{(t)} + \beta\boldsymbol{\zeta}^* \tag{12}$$

This changes the form of our acceptance probability subtly by exchanging the log-posterior $\mathcal{P}$ for the log-likelihood Λ. The rest of the algorithm is broadly unchanged (see algorithm 2).

Algorithm 2 preconditioned Crank-Nicholson

$\boldsymbol{u}^{(0)} \leftarrow \boldsymbol{\zeta}^*$
for $t \in \{0, 1 \cdots T-1\}$ **do**
  $\boldsymbol{w}^{(t)} \leftarrow \sqrt{1-\beta^2}\boldsymbol{u}^{(t)} + \beta\boldsymbol{\zeta}^*$ ▷ Generate proposal
  $\ln \alpha^{(t)} \leftarrow \min\left(\Lambda(\boldsymbol{w}^{(t)}) - \Lambda(\boldsymbol{u}^{(t)}), 0\right)$
  $U^{(t)} \leftarrow\sim \mathcal{U}(0,1)$

  if $\ln U^{(t)} \leq \ln \alpha^{(t)}$ **then**
    $\boldsymbol{u}^{(t+1)} \leftarrow \boldsymbol{w}^{(t)}$ ▷ Proposal accepted
  else
    $\boldsymbol{u}^{(t+1)} \leftarrow \boldsymbol{u}^{(t)}$ ▷ Proposal rejected
  end if
end for

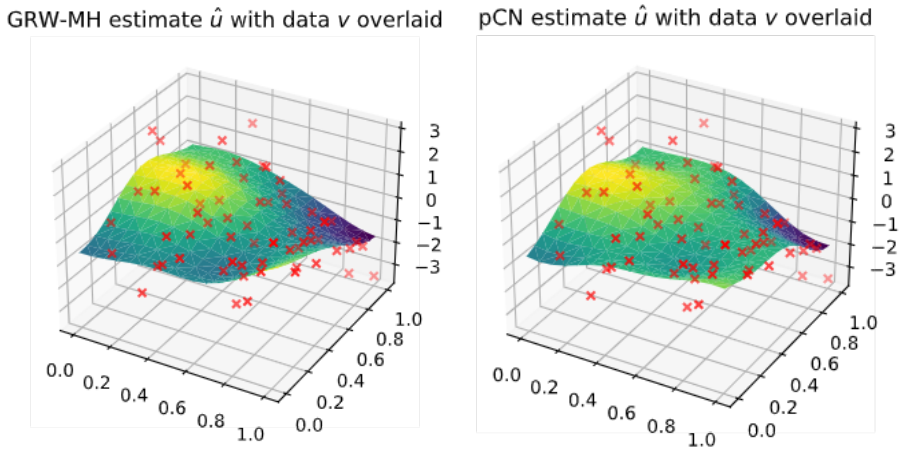
A huge advantage of pCN compared to GRW-MH is that we do not need to compute the prior probability for each proposal. This massively speeds up computation.

b.3 Method comparison

We run both algorithms for $T = 10^4$ iterations and $\beta = 0.2$. The algorithm returns the set $\{\boldsymbol{u}^{(t)}\}_{t=1}^{T}$ which can be averaged to return a Monte-Carlo estimate of $\boldsymbol{u}$ denoted $\hat{\boldsymbol{u}}$:

$$\hat{\boldsymbol{u}} = \frac{1}{T}\sum_{t=1}^{T}\boldsymbol{u}^{(t)} \tag{13}$$

Formally we would apply burn-in B and thinning H but that is not the focus of this report. Choosing these parameters optimally is a report in itself. B is chosen to be at least greater than the time it takes the Markov chain to reach a stationary distribution (can be estimated by plotting the sample magnitude with respect to iteration). H is chosen by looking at the autocorellation of the generated samples and choosing an offset for which it is close to zero.



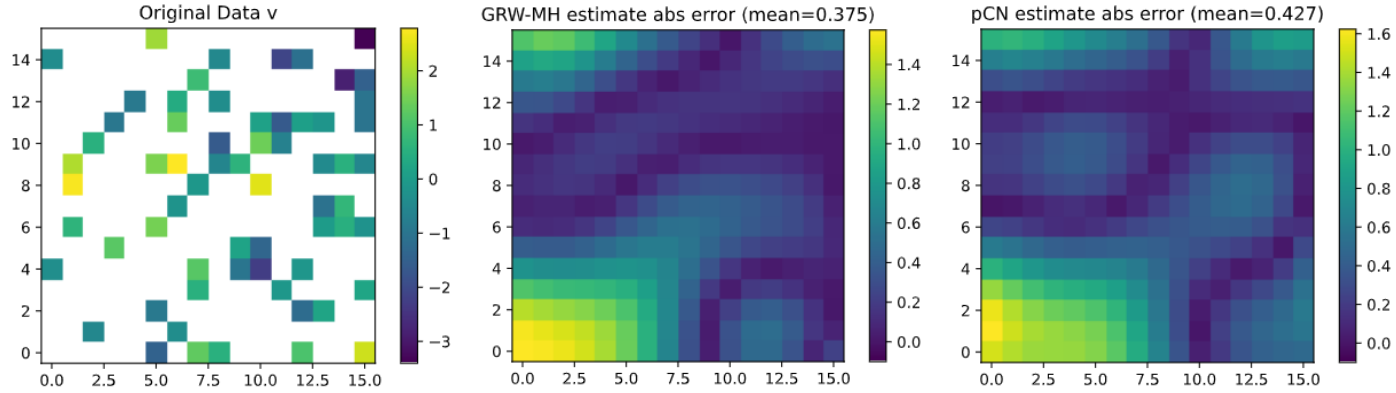
(a) GRW-MH estimate

(b) pCN estimate

Figure 3: Monte Carlo estimate errors ($D = 16, T = 10^4, \beta = 0.2$)

We plot the estimated surface $\hat{\boldsymbol{u}}$ predicted by each algorithm in figure 3. However, this plot is not very easy to interpret as both look broadly similar. Instead, we prefer to plot the absolute error surface $|\hat{\boldsymbol{u}} - \boldsymbol{u}|$ as that better visualises the accuracy of the predictions.

The absolute error is plotted on figures 4b and 4c. The error surface for each algorithm has a broadly similar shape (the mean absolute error is marginally higher for pCN). Indeed, by comparison with figure 4a, the regions of high error are those in areas of sparse available data $\boldsymbol{v}$. This is to be expected as we cannot make a confident prediction in areas where we have few data-points.



(a) Data v projected onto X-Y plane

(b) GRW-MH absolute error $|\hat{\boldsymbol{u}} - \boldsymbol{u}|$

(c) pCN absolute error $|\hat{\boldsymbol{u}} - \boldsymbol{u}|$

Figure 4: Monte Carlo estimate errors ($D = 16, T = 10^4, \beta = 0.2$)

The acceptance rate is an important measure of the wastefulness of the algorithm. We tabulate it in table 1 for a range of grid-sizes D and step-sizes β.

Acceptance Rate (GRW-MH | pCN)		β = 0.04	β = 0.2	β = 1.0
D	4	0.932 | 0.973	0.692 | 0.886	0.054 | 0.302
	16	0.694 | 0.851	0.097 | 0.498	0.000 | 0.010

Table 1: Acceptance rate for GRW-MH | pCN algorithms for varying β, D

Comparing the two algorithms, pCN has a higher acceptance for all tested scenarios. Higher dimensional spaces (larger D) necessarily have lower acceptance rates as there are more degrees of freedom so it is less likely that a random walk will move in a direction that increases the posterior (or likelihood for pCN). The acceptance rate for both algorithms decreases as β increases. Nevertheless, having β too low runs the risk of falling into a local optimum as there is insufficient energy to escape shallow wells. Therefore, it is less likely to converge to the true global optimum.

c Probit classification

The model is now extended to work on a probit classification problem. The data $\boldsymbol{v}$ is put through a sign function to give the vector $\boldsymbol{t}$, such that $t_i \coloneqq \text{sign}(v_i) \in \{-1, +1\}$. As such the likelihood has the following form:

$$\begin{aligned} p(t_i = 1|\boldsymbol{u}) &= p(v_i > 0|\boldsymbol{u}) \\ &= p([G\boldsymbol{u}]_i + \epsilon_i > 0) \\ &= p(-\epsilon_i < [G\boldsymbol{u}]_i) \\ &= \Phi([G\boldsymbol{u}]_i) \end{aligned} \tag{14}$$

Where $\Phi(\cdot)$ is the standard Gaussian CDF (as $-\epsilon_i \sim \epsilon_i \sim \mathcal{N}(0, 1)$). Conversely, for the case $t_i = -1$ the likelihood is given by:

$$\begin{aligned} p(t_i = -1|\boldsymbol{u}) &= 1 - p(t_i = 1|\boldsymbol{u}) \\ &= 1 - \Phi([G\boldsymbol{u}]_i) \\ &= \Phi(-1 \cdot [G\boldsymbol{u}]_i) \end{aligned} \tag{15}$$

This means we can simplify the expression for $t_i \in -1, 1$, leading to:

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

We can extend this easily to find the likelihood $\lambda(\cdot)$[1] of the latent variables $\boldsymbol{u}$ given the whole vector of observations $\boldsymbol{t}$:

$$\begin{aligned} \lambda(\boldsymbol{u}) &= p(\boldsymbol{t}|\boldsymbol{u}) \\ &= \prod_{i=1}^{M} p(t_i|\boldsymbol{u}) \\ &= \prod_{i=1}^{M} \Phi(t_i \cdot [G\boldsymbol{u}]_i) \end{aligned} \tag{17}$$

The second line arises from the fact that $t_i \perp\!\!\!\perp t_j|\boldsymbol{u}$ or more specifically given $G\boldsymbol{u}$. This a direct result of equation 2 as the noise terms ϵ_i are mutually independent. However, instead we find it easier to deal with the log-likelihood $\Lambda(\cdot) \coloneqq \ln \lambda(\cdot)$ as this turns the multiplication into a summation:

$$\begin{aligned} \Lambda(\boldsymbol{u}) &= \sum_{i=1}^{M} \ln \Phi(t_i \cdot [G\boldsymbol{u}]_i) \\ &= \mathbf{1}^T \left[\ln \Phi(\boldsymbol{t} \odot G\boldsymbol{u})\right] \end{aligned} \tag{18}$$

Where $\odot$ denotes element-wise multiplication of vectors, ln and Φ are extended to operate element-wise also and $\mathbf{1}$ is simply the vector of all-ones. This form in equation 18 is very easy to implement using numpy as it is vectorised. With this log-likelihood function we can sample from the posterior $p(\boldsymbol{u}|\boldsymbol{t})$ using pCN. From these samples we can perform a Monte Carlo estimate of the true class assignments for any t_j^* not restricted to the subsampling imposed by G. The Monte-Carlo estimate is computed as follows:

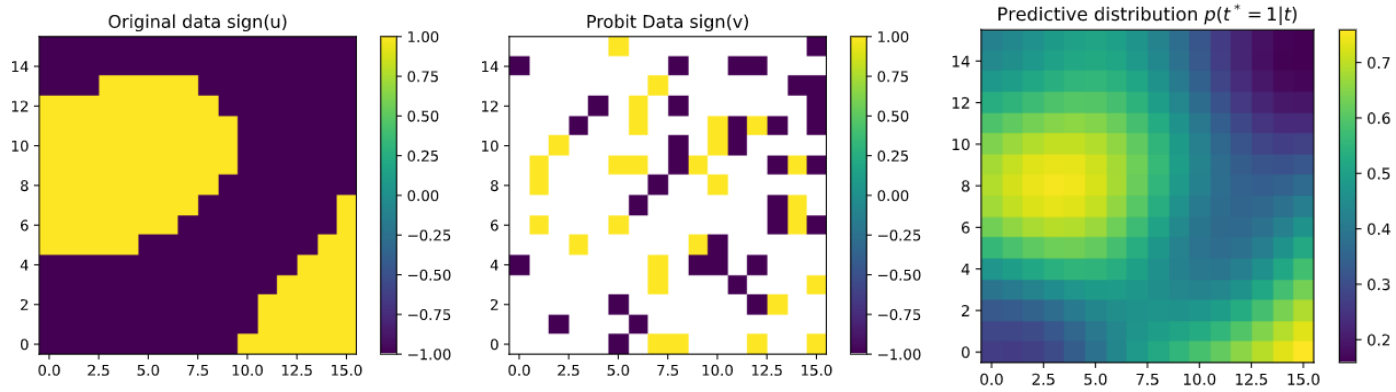
$$\begin{aligned} p(t_j^* = 1|\boldsymbol{t}) &= \int p(t^* = 1, \boldsymbol{u}|\boldsymbol{t}) d\boldsymbol{u} \\ &= \int p(t^* = 1|\boldsymbol{u}) p(\boldsymbol{u}|\boldsymbol{t}) d\boldsymbol{u} \\ &\approx \frac{1}{T} \sum_{t=1}^{T} p\left(t^* = 1|\boldsymbol{u}^{(t)}\right) \\ &= \frac{1}{T} \sum_{t=1}^{T} \Phi\left(u_j^{(t)}\right) \end{aligned} \tag{19}$$

[1] Note that for simplicity we are not changing notation and all previous symbols in the $\boldsymbol{v}$ problem will retain their meaning for the $\boldsymbol{t}$ problem

Where each $\boldsymbol{u}^{(t)}$ denotes a sample drawn from the posterior $p(\boldsymbol{u}|\boldsymbol{t})$ as is the case under pCN. As we are bypassing the subsampling matrix G both the $\boldsymbol{t}^*$ and $\boldsymbol{u}^*$ vectors are indexed by the same variable j, labelling position on the finite 2-D grid on the X-Y plane. We can quickly compute the vector of probabilities by dropping the j dependence to give equation 20:

$$p(\boldsymbol{t}^*|\boldsymbol{t}) \approx \frac{1}{T}\sum_{t=1}^{T} \Phi\left(\boldsymbol{u}^{(t)}\right) \tag{20}$$

This is a Monte-Carlo estimate, using the generated samples. We can plot the predictive probabilities on figure 5c. We see that the predictive distribution broadly follows the shape of figure 5a. This is particularly impressive given the corruption apparent in figure 5b.



(a) True values $\boldsymbol{t}^* = sign(\boldsymbol{u})$ (b) Subsampled values after noise $\boldsymbol{t}$ (c) Predictive distribution $p(t^* = 1|\boldsymbol{t})$

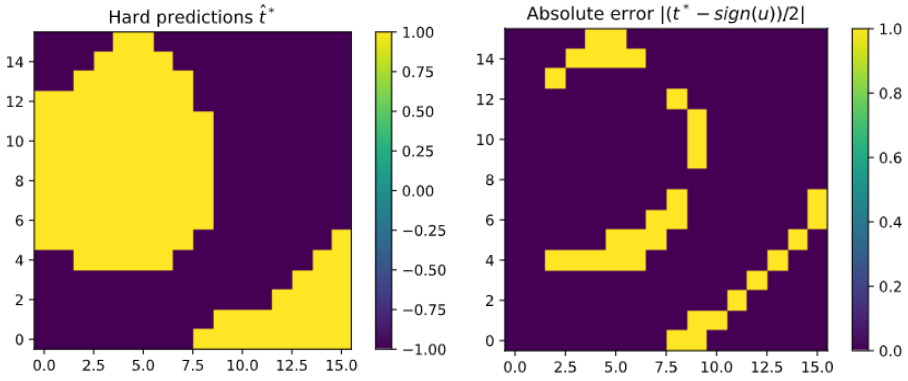
Figure 5: Monte Carlo estimate of probit data ($D = 16, T = 10^4, \beta = 0.2$)

d Length scale inference

We now make hard assignments in the probit classification problem by thresholding at $p(t^* = 1|\boldsymbol{t}) = 0.5$ such that:

$$\hat{t}_j^* = \begin{cases} +1, & \text{if } p(t_j^* = 1|\boldsymbol{t}) \geq 0.5 \\ -1, & \text{otherwise} \end{cases} \tag{21}$$

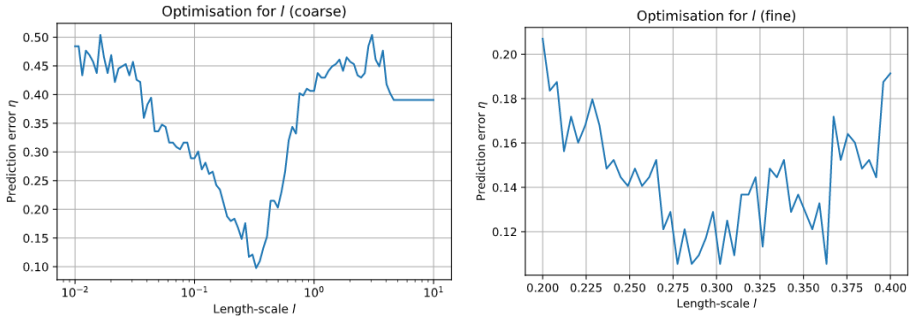
From this we can plot the absolute prediction error $|(\hat{t}_j^* - u_j)/2|$ and compute the mean η over the index j - which is equal to the probability of prediction error $\eta = p(t_j^* \neq \text{sign}(u_j))$ for general j. The hard assignments alongside the absolute errors are plotted in figure 6.



(a) Hard predictions $\hat{\boldsymbol{t}}^*$ (b) Absolute error $|(\hat{t}_j^* - u_j)/2|$

Figure 6: Hard assignment prediction error ($D = 16, T = 10^4, \beta = 0.2, l = 0.3$)

The plot in figure 6b gives a mean absolute error $\eta = 0.125$. However, we wish to infer the length scale parameter l by performing a line-search and minimising the mean absolute error η. This can be illustrated through a coarse logarithmic plot in the range $l \in [0.01, 10]$ and a finer linear plot in the range $l \in [0.2, 0.4]$ as in figures 7a and 7b respectively.



(a) Coarse logarithmic scale

(b) Fine linear scale

Figure 7: Prediction error η against length-scale l

It is rather computationally intensive to generate these plots in figure 7. We could instead perform a golden ratio line search to find the minimum. However, as each η evaluation is non-deterministic the overall function would not be smooth for practical values of n so this method is likely to have high error. By inspecting figure 7b it is hard to obtain a point-estimate due to the uncertainty in each point. The best estimate would be $\hat{l} = 0.325 \pm 0.5$. This range encompasses the true value $l_0 = 0.3$. Therefore, inference in this case has succeeded.

Nevertheless, the method used to infer the estimate is hardly robust. A more generalisable approach can be obtained by returning to Bayes' rule in its full form and making explicit the dependence on length-scale l:

$$p(\boldsymbol{u}|\boldsymbol{t}, l) = \frac{p(\boldsymbol{u}|l) \cdot p(\boldsymbol{t}|\boldsymbol{u}, l)}{p(\boldsymbol{t}|l)} \tag{22}$$

The term $p(\boldsymbol{t}|l)$ is independent of $\boldsymbol{u}$ and called the model evidence. It can be computed through a Monte-Carlo estimate as follows:

$$\begin{aligned} p(\boldsymbol{t}|l) &= \int p(\boldsymbol{t}|\boldsymbol{u}, l) \cdot p(\boldsymbol{u}|l) d\boldsymbol{u} = \int p(\boldsymbol{t}|\boldsymbol{u}) \cdot p(\boldsymbol{u}|l) d\boldsymbol{u} \\ &\approx \frac{1}{T} \sum_{i=1}^{T} p(\boldsymbol{t}|\boldsymbol{u}^{(i)}) \quad \text{for } \boldsymbol{u}^{(i)} \sim p(\boldsymbol{u}|l) \\ &= \frac{1}{T} \sum_{i=1}^{T} \prod_{j=1}^{M} \Phi(t_j \cdot [G\boldsymbol{u}^{(i)}]_j) \end{aligned} \tag{23}$$

With this equation we can compute a Maximum-Likelihood estimate $\hat{l}^{ML} = \arg\max_l p(\boldsymbol{t}|l)$; this is equivalent to maximising the model evidence. If we wanted to give l a full Bayesian treatment, we would define a prior $p(l)$ and compute the posterior $p(l|\boldsymbol{t}) \propto p(l) \cdot p(\boldsymbol{t}|l)$. The posterior would be a complete probability distribution rather than a single point-estimate.

3 Spatial Data

We now turn our attention to real-world data. Specifically, we wish to develop a model to predict bike thefts in Lewisham borough based on spatial theft data. We have a series of 400m^2 cells identified by a (x, y) coordinate and for each cell we have the total number of bike thefts within over 2015. This is denoted by a vector $\boldsymbol{c}$.

a Poisson Likelihood

We now deal with a Poisson likelihood. The field $\boldsymbol{u}$ is mapped to $\boldsymbol{\theta} = \exp(\boldsymbol{u})$ for positivity (there cannot be negative thefts in a region) and θ is used as the Poisson rate for each cell. The overall likelihood function is given below:

$$p(\boldsymbol{c}|\boldsymbol{\theta}) = \prod_{i=1}^{N} p(c_i|\theta_i) = \prod_{i=1}^{N} \frac{\exp(-\theta_i) \cdot \theta_i^{c_i}}{c_i!} \tag{24}$$

As always we prefer to work with the log-likelihood giving us:

$$\ln p(\boldsymbol{c}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \left(-\theta_i + c_i \ln \theta_i - \ln(c_i!)\right) \tag{25}$$

Substituting for $\boldsymbol{\theta} = \exp(u)$ we get:

$$\begin{aligned} \Lambda(\boldsymbol{u}) &= \sum_{i=1}^{N} \left(-\exp(u_i) + c_i \cdot u_i - \ln(c_i!)\right) \\ &= -\mathbf{1}^T \exp(\boldsymbol{u}) + \boldsymbol{c}^T \boldsymbol{u} - f(\boldsymbol{c}) \end{aligned} \tag{26}$$

We can ignore the term $f(\boldsymbol{c})$ for the purposes of pCN as it is independent of $\boldsymbol{u}$ and we are only interested in the difference of Λ. Once we implement $\Lambda(\boldsymbol{u})$ we can use pCN to sample from the posterior $p(\boldsymbol{u}|\boldsymbol{c})$ with our standard operating parameters. Nevertheless, we must begin by guessing the length-scale parameter l as this is not known in advance. This is covered in the subsequent section.

b Bike theft predictions

We wish to estimate the expected counts in each cell from the predictive distribution on a new cell c_j^* given the subsampled counts $\boldsymbol{c}$:

$$\mathbb{E}_{p(c_j^*|\boldsymbol{c})}[c^*] = \sum_{k=0}^{\infty} k \cdot p(c_j^* = k|\boldsymbol{c}) \tag{27}$$

The predictive distribution can be computed through a Monte-Carlo estimate as follows:

$$\begin{aligned} p(c_j^* = k|\boldsymbol{c}) &= \int p(c_j^* = k|\boldsymbol{u}) \cdot p(\boldsymbol{u}|\boldsymbol{c}) d\boldsymbol{u} \\ &\approx \frac{1}{T} \sum_{i=1}^{T} p(c_j^* = k|\boldsymbol{u}^{(i)}) \quad \text{for } \boldsymbol{u}^{(i)} \sim p(\boldsymbol{u}|\boldsymbol{c}) \end{aligned} \tag{28}$$

Substituting into equation 27 we can rearrange to obtain:

$$\begin{aligned} \mathbb{E}_{p(c_j^*|\boldsymbol{c})}[c_j^*] &\approx \sum_{k=0}^{\infty} k \cdot \frac{1}{T} \sum_{i=1}^{T} p(c_j^* = k|\boldsymbol{u}^{(i)}) \\ &= \frac{1}{T} \sum_{i=1}^{T} \sum_{k=0}^{\infty} k \cdot p(c_j^* = k|\theta_j^{(i)}) \\ &= \frac{1}{T} \sum_{i=1}^{T} \mathbb{E}_{p(c_j^*|\theta_j^{(i)})}[c_j^*] \\ &= \frac{1}{T} \sum_{i=1}^{T} \theta_j^{(i)} \end{aligned} \tag{29}$$

Where the last line holds as the expectation of a Poisson variable is simple its parameter.