CUED - Engineering Tripos Part IIB 2021-2022 **Module Coursework**

Module	4M24	Title of report	High Dimensional MCMC
Date submitted: 10/12/2021		Assessment for this module is ☐ 100% / ☒ 25% coursework of which this assignment forms 100 %	

UNDERGRADUATE STUDENTS ONLY		POST GRADUATE STUDENTS ONLY			
Candidate number:	5585G	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:			

Indicative grades are not provided for the FINAL piece of coursework in a module

Assessment (circle one or two grades)	A*	A	B	C	D
Indicative grade guideline	>75%	65-75%	55-65%	40-55%	<40%
Penalty for lateness:	*20% of maximum achievable marks per week or part week that the work is late.*				

Marker: Date:

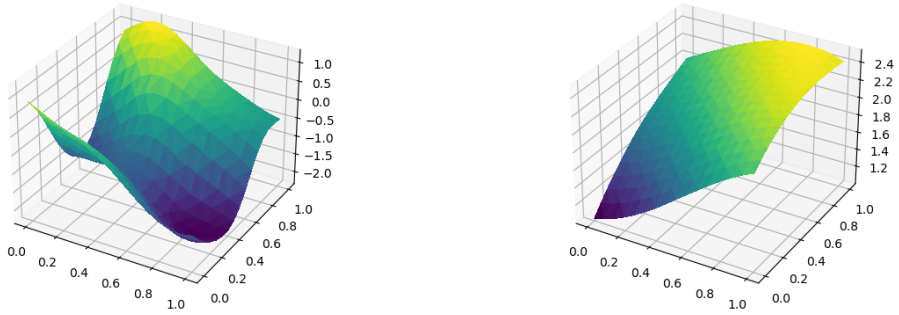
1 Part I: Simulation

1.1 Part (a):

In this first section we sample from the GP data prior with a Gaussian covariance function, $k(\boldsymbol{x}, \boldsymbol{x}')$, over a $D \times D$ grid. These prior $\boldsymbol{u} \in \mathbb{R}^2$ are then subsampled by a factor 4 and corrupted by unit Gaussian noise to formulate our observations, $\boldsymbol{v}$. The Gaussian covariance function is defined as

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

which means that two nearby points are more similar than two points further apart. The concept of *nearby* is encoded in the hyperparameter l: for small l this similarity drops off very quickly (and in the limit $l \to 0$ the samples are independent); increasing l leads to a smoother, more slowly varying surface. For sufficiently large length scale l (e.g. $l = 0.3$ as suggested in the coursework handout), the prior surface is smoothly varying with peaks, troughs and saddle points. Two examples are shown in figure (1). The



(a) $l = 0.3$ (b) $l = 1.0$

Figure 1: GP prior surfaces

subsampled and corrupted data for the surface with $l = 0.3$ is shown in figure (2) The observed datapoints

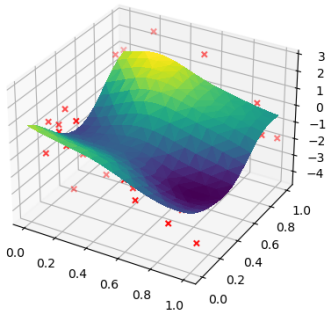


Figure 2: Sampled data for $l = 0.3$

are represented as red crosses. They deviate quite significantly on the vertical axis due to the noise, and represent only a quarter of the total grid generated by the prior.

1.2 Part (b):

In this section we consider the Gaussian Random Walk Metropolis Hastings (GRW-MH) and pre-conditioned Crank Nicholson (pCN) algorithms. Initially, we must specify the log prior and log likelihood functions for this model, which are $\ln p(\boldsymbol{u})$ and $\ln p(\boldsymbol{v}|\boldsymbol{u})$ respectively. The distributions for $\boldsymbol{u}$ and $\boldsymbol{v}|\boldsymbol{u}$ are

$$\begin{aligned} \boldsymbol{u} &\sim \mathcal{N}(0, C) \\ \boldsymbol{v}|\boldsymbol{u} &\sim \mathcal{N}(G\boldsymbol{u}, I) \end{aligned} \tag{2}$$

Where $(C)_{ij} = k(\boldsymbol{x_i}, \boldsymbol{x_j})$. Then

$$\begin{aligned} \ln p(\boldsymbol{u}) &= -\ln 2\pi + \frac{1}{2}\ln|C^{-1}| - \frac{1}{2}\boldsymbol{u}^T C^{-1}\boldsymbol{u} \\ \ln p(\boldsymbol{v}|\boldsymbol{u}) &= -\ln 2\pi - \frac{1}{2}(\boldsymbol{v} - G\boldsymbol{u})^T(\boldsymbol{v} - G\boldsymbol{u}) \end{aligned} \tag{3}$$

The log determinant term in $\ln p(\boldsymbol{u})$ was implemented using `np.linalg.slogdet(K_inverse)` to avoid possible overflow problems. The lines of code to be completed in the GRW and pCN functions are shown in figure (3).

```
# compute the inverse of K using its Cholesky decomposition
K_inverse = Kc_inverse.T @ Kc_inverse

# GRW propose new sample
u_new = u_prev + beta*Kc@np.random.normal(size=N)

# pCN propose new sample
u_new = np.sqrt(1-beta**2)*u_prev + beta*Kc@np.random.normal(size=N)

# GRW log acceptance probability
log_alpha = min(lt_new - lt_prev, 0.)

# pCN log acceptance probability
log_alpha = min(ll_new - ll_prev, 0.)

# accept/reject sample
accept = (log_alpha > log_u)
```

Figure 3: Code for part (b)

For GRW, the proposal is

$$x^{(i+1)} = x^{(i)} + \beta\xi \quad \text{with} \quad \xi \sim \mathcal{N}(0, C)$$

and for pCN it is

$$x^{(i+1)} = \sqrt{1-\beta^2}x^{(i)} + \beta\xi \quad \text{with} \quad \xi \sim \mathcal{N}(0, C)$$

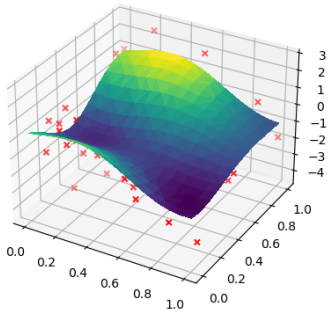
where β is the step size parameter. Further, for GRW the acceptance probability is defined in terms of the log posterior as

$$\ln \alpha_{GRW}(x^{(i)}, x^{(i+1)}) = \min\big(0,\ \Phi(x^{(i)}) + \frac{1}{2}||C^{-\frac{1}{2}}x^{(i)}||^2 - \Phi(x^{(i+1)}) - \frac{1}{2}||C^{-\frac{1}{2}}x^{(i+1)}||^2\big)$$

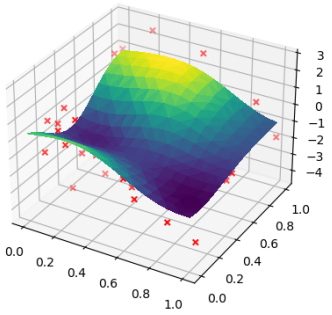
Where $\Phi(\cdot)$ is the negative log likelihood and the norm term is due to the prior. In contrast, for pCN this acceptance probability is just

$$\ln \alpha_{pCN}(x^{(i)}, x^{(i+1)}) = \min\big(0,\ \ \Phi(x^{(i)}) - \Phi(x^{(i+1)})\big)$$

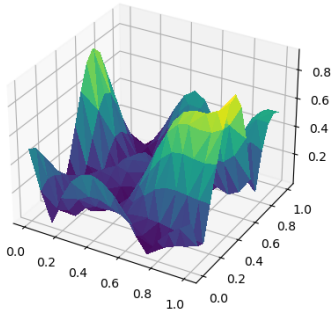
The results of running both of these MCMC algorithms for $D = 16$, $n = 10,000$ and $\beta = 0.2$ are shown in (4). The corresponding acceptance rates for GRW and pCN were 0.0827 and 0.4592 respectively. The chains were initialised using a sample from the prior $p(\boldsymbol{u})$.



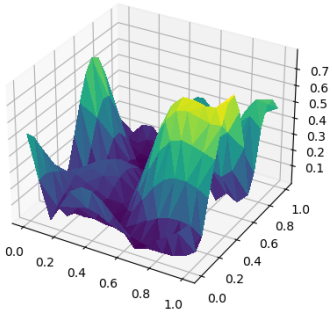
(a) Inferred GRW field



(b) Inferred pCN field



(c) GRW absolute error field



(d) pCN absolute error field

Figure 4: Inferences and error fields

The *inferred fields* are just the mean values of the chain across all iterations. The fields seem reasonable compared to the ground truth, particularly considering the scale of the noise variance relative to the variation in the prior field. As a sanity check, I ran the MCMC algorithms with no subsampling and no noise added - as expected, they seemed to converge to the ground truth field with low mean absolute error (< 0.1).

Further, I tested for burn-in by removing the first 1000 samples and noticed that the inferred fields were qualitatively the same and generally did not improve the mean absolute error, hence the chain is very quickly moving into the high probability regions of space and I decided not to account for burn-in in the following sections.

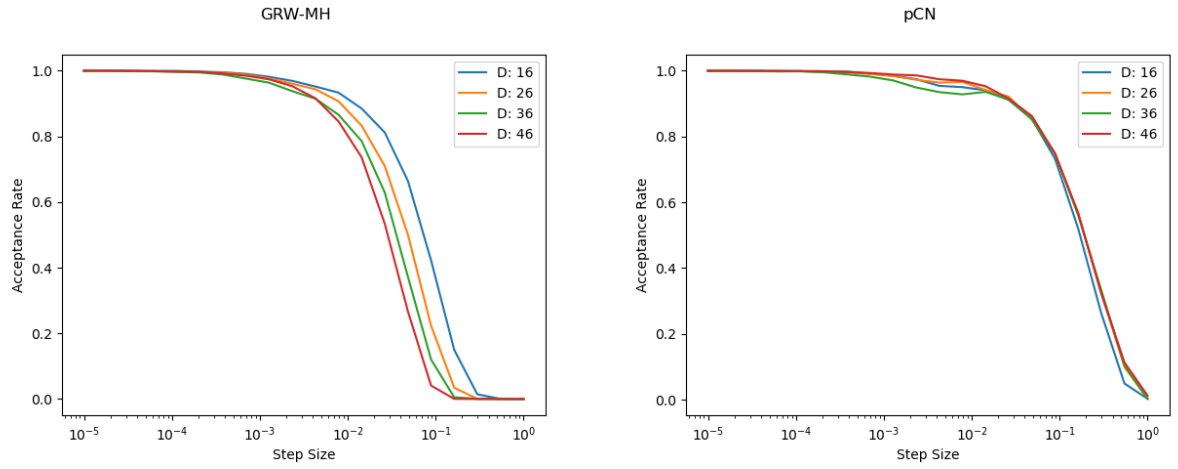
The absolute error fields are very similar for both algorithms which suggests the chains are exploring the same posterior, and each error field display some correlation across space - this makes sense considering the

correlated structure of the underlying and inferred fields. pCN provides a marginally lower average error than GRW for this problem (0.2701 vs 0.2511), and consistently did so across several trials - I also observed a wide range of mean errors so these values are not indicative of the expected performance of these algorithms. The reason for this slight improvement may be the faster convergence of pCN owing to it's higher acceptance rate.

As a test, a lower dimension chain was considered with $D = 4$. This corresponds to very few observed data points, which was reflected in the average errors (GRW: 0.9404 and pCN: 1.0959), however the acceptance rates were much higher than the higher dimensional case - 0.6656 for GRW and 0.8772 for pCN. This is as expected considering the degeneracy of probability mass to very concentrated areas in higher dimensions.

Similarly, β was varied from 0 to 1 to see the effect of step size. For very small $\beta = 0.01$, acceptance rates are high (GRW: 0.9233 and pCN: 0.9642) but the steps are so small that convergence becomes unnecessarily slow - with mean errors 1.2597 and 1.3094. For large $\beta = 0.9$, the acceptance rates become wastefully small (GRW: 0.0007 and pCN: 0.0023), but convergence is much faster than for the short steps - with mean errors 0.3568 and 0.3736.

Very small β leads to strong correlation between samples, since the random component of the proposal is vanishingly small in comparison to the deterministic part. To counter this we would need to thin the samples significantly, which would increase the length of time that the chain must be run for. No thinning was used in the later sections of this coursework on the assumption that β is large enough and that we have enough samples so that this correlation is not a problem - particularly as the order of the sequence of samples is not relevant, only their mean.



(a) GRW-MH (b) pCN

Figure 5: Effect of increasing dimension on acceptance.

The robustness of pCN to mesh refinement is demonstrated in figure (5). As observed before, the acceptance rate of both MCMC algorithms is dependent on the step size and chain dimension. However, for a fixed observation set, $\boldsymbol{v} \in \mathbb{R}^M$, we can consider chains on increasingly finer meshes in order to obtain better approximations to the infinite dimensional limiting space. The variable D on the plots is the number of grid points on each axis, hence the chain is $\boldsymbol{u} \in \mathbb{R}^{D^2}$. These plots show that to maintain a sensible acceptance rate in GRW-MH we must drop the step size as the dimension increases, but this is not the case for pCN whose acceptance is unaffected by the change in dimension. This is incredibly valuable as we can maintain a moderate value of β in these higher dimensional spaces, meaning less samples are required.

1.3 Part (c):

The model is now extended to probit classification. The observed data $\boldsymbol{v}$ are classified with $v_i \leqslant 0$ mapping to 0 and $v_i > 0$ mapping to 1, leading to $p(t_i = 1|\boldsymbol{u}) = \Phi([G\boldsymbol{u}]_i)$, where $\Phi(\cdot)$ is the standard normal cdf. For M iid observations, the likelihood is

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

Hence

$$\begin{aligned} \ln p(\boldsymbol{t}|\boldsymbol{u}) &= \sum_{i=1}^{M} t_i \ln p(t_i = 1|\boldsymbol{u}) + (1 - t_i) \ln p(t_i = 0|\boldsymbol{u}) \\ &= \sum_{i=1}^{M} t_i \ln \Phi([G\boldsymbol{u}]_i) + (1 - t_i) \ln \Phi(-[G\boldsymbol{u}]_i) \end{aligned} \tag{4}$$

Since $1 - \Phi(x) = \Phi(-x)$. The predictive distribution for binary classifications over this space is then

$$p(t^* = 1|\boldsymbol{t}) = \int p(t^* = 1|\boldsymbol{u}) p(\boldsymbol{u}|\boldsymbol{t}) d\boldsymbol{u}$$

Observing that

$$p(t_i^* = 1|\boldsymbol{t}) = \mathbb{E}_{p(\boldsymbol{u}|\boldsymbol{t})} p(t_i^* = 1|\boldsymbol{u})$$

We can estimate this in a Monte Carlo scheme using a set of S posterior samples $\{\boldsymbol{u}^{(s)}\}_{s=1}^{S}$ as

$$\mathbb{E}_{p(t_i^*=1|\boldsymbol{t})} t_i^* \approx \frac{1}{S} \sum_{s=1}^{S} \Phi([\boldsymbol{u}^{(s)}]_i)$$

We can generate samples from the probit posterior $p(\boldsymbol{u}|\boldsymbol{t})$ by using the probit likelihood in our pCN function from part (b). The code to compute this MC estimate of the predictive distribution given the posterior samples is shown in figure (6).

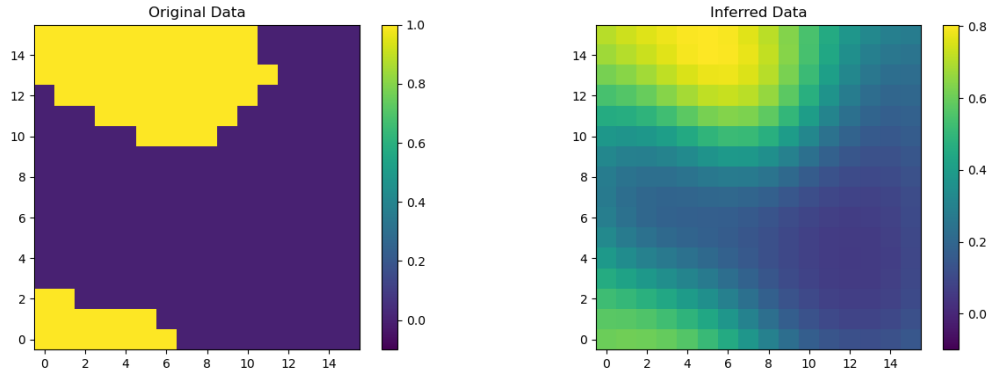
```
def predict_t(samples):
    # Calculate Phi(u)
    cdffield = norm.cdf(samples)
    # Monte carlo estimate
    return np.mean(cdffield, axis=0)
```

Figure 6: Code for part (b)

This maps our samples values on the real line to soft class assignments in the interval $[0, 1]$, which quantify how uncertain the classifier is about assigning the point t^* to class 1. This is visualised in figure (7); there are clustered regions of high probability of belonging to class 1 which describe the true class assignments well, and similarly for class 0. Between these regions the model gradually transitions between classes. Notably, even in the highest confidence regions the probabilities are around 0.8 (or just under 0.2 for regions of class 0) - this suggests the model is not overfitting, which makes sense due to our regularising GP prior on the underlying field $\boldsymbol{u}$.

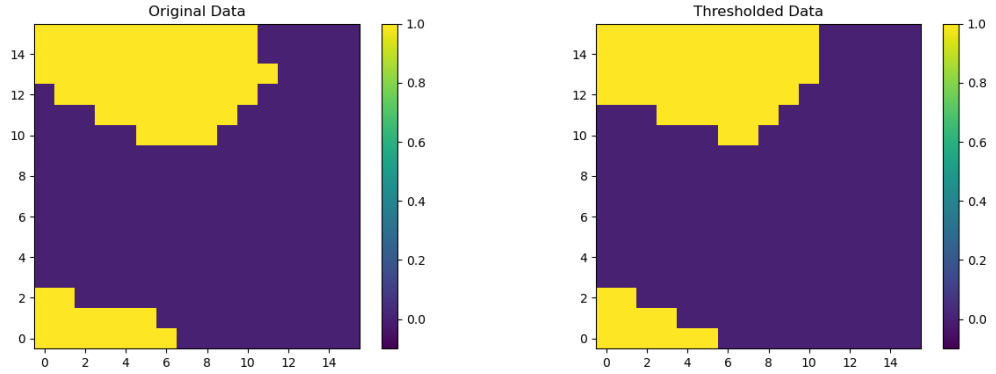
1.4 Part (d):

The result of hard assignments at $p(t^* = 1|\boldsymbol{t}) = 0.5$ are shown in figure (8)



(a) t_{true} (b) Soft assignments

Figure 7: Soft class assignments along with the underlying field

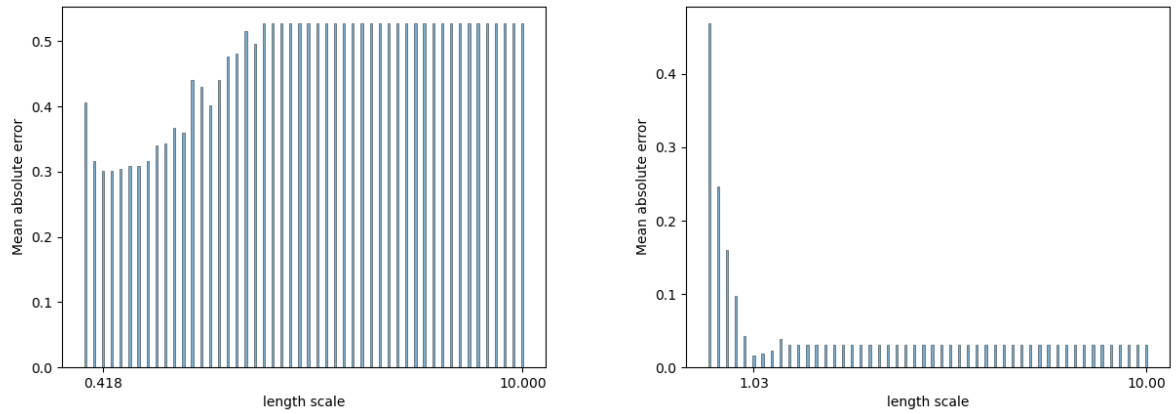


(a) t_{true} (b) Hard assignments

Figure 8: Hard class assignments along with the underlying field

At first glance this is a good estimate of the class assignments, and indeed this is confirmed by the mean absolute error (or equivalently for probit classification, the mean square error) of 0.03516 - i.e. 9 incorrect predictions. There was some variation in this error value: some fields belonged entirely to one class and were very easily predicted while others were slightly more complicated. I ran this prediction process on 500 different datasets generated with $l = 0.3$, for which the mean error was 0.1550 - hence the model is predicting to about 85% accuracy.

In practice, we are unlikely to know the true value of l used to generate the data. Up until now we have used the true value for training, but we now consider a 1D grid-search across a range of l values to optimise the mean prediction error.



(a) $l_{\text{true}} = 0.3$ (b) $l_{\text{true}} = 0.9$

Figure 9: Heat maps for two separate runs

The grid search could quite reliably predict close to the true l value for short length scales, but became much less reliable when l was greater than 2 or so.

2 Part II: Spatial

2.1 Part (e):

These models are now applied to some real data. We have subsampled 2D spatial count data $\{\boldsymbol{u}\}$, for which we would like to infer the underlying field. We enforce positivity in the predicted counts using the exponential function - we define $\theta_i = \exp([G\boldsymbol{u}]_i)$, and the corresponding Poisson likelihood with rate parameter $\boldsymbol{\theta}$

$$\begin{aligned} p(\boldsymbol{c}|\boldsymbol{\theta}) &= \prod_{i=1}^{M} \frac{e^{-\theta_i}\theta_i^{c_i}}{c_i!} \\ \ln p(\boldsymbol{c}|\boldsymbol{\theta}) &= \sum_{i=1}^{M} -\theta_i + c_i \ln \theta_i - \ln c_i! \end{aligned} \tag{5}$$

$$\ln p(\boldsymbol{c}|\boldsymbol{u}) = \sum_{i=1}^{M} -e^{u_i} + c_i u_i - \ln c_i! \tag{6}$$

This form of the likelihood can be used in the pCN method (without requiring explicit calculation of

```
def log_poisson_likelihood(u, c, G):
    gu = G @ u
    # work in terms of u (not theta)
    summand = -np.exp(gu) + c*gu
    return np.sum(summand)
```

Figure 10: Code for part (e)

$\ln c_i!$, since this is fixed with respect to the samples $\boldsymbol{u}$ and cancels out in the expression for the acceptance probability) to generate samples from $p(\boldsymbol{u}|\boldsymbol{c})$.

2.2 Part (f):

Since our count distribution is discrete, the inferred counts are calculated as an infinite sum

$$\mathbb{E}_{p(c^*|\boldsymbol{c})}c^* = \sum_{k=0}^{\infty} kp(c^* = k|\boldsymbol{c})$$

Where

$$p(c^* = k|\boldsymbol{c}) = \int p(c^* = k|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{c})d\boldsymbol{\theta}$$

Hence

$$\mathbb{E}_{p(c^*|\boldsymbol{c})}c^* = \sum_{k=0}^{\infty} k \int p(c^* = k|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{c})d\boldsymbol{\theta} = \int p(\boldsymbol{\theta}|\boldsymbol{c}) \sum_{k=0}^{\infty} kp(c^* = k|\boldsymbol{\theta})d\boldsymbol{\theta}$$

Now we observe that the internal sum is just an expected value

$$\sum_{k=0}^{\infty} kp(c^* = k|\boldsymbol{\theta}) = \mathbb{E}_{p(c^*|\boldsymbol{\theta})}c^* = \boldsymbol{\theta}$$

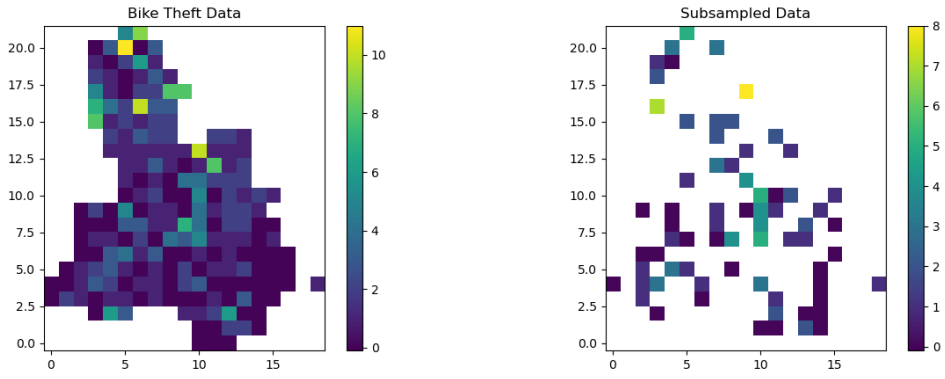
So our predictive distribution is now

$$\mathbb{E}_{p(c^*|\boldsymbol{c})}c^* = \int p(\boldsymbol{\theta}|\boldsymbol{c})\boldsymbol{\theta}d\boldsymbol{\theta}$$

Which can be estimated in an MC scheme using our posterior samples $\{\boldsymbol{u}^{(s)}\}$ for the whole predictive field $\boldsymbol{c}^*$

$$\mathbb{E}_{p(\boldsymbol{c}^*|\boldsymbol{c})}\boldsymbol{c}^* \approx \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{\theta}^{(s)} = \frac{1}{S}\sum_{s=1}^{S} \exp(\boldsymbol{u}^{(s)})$$

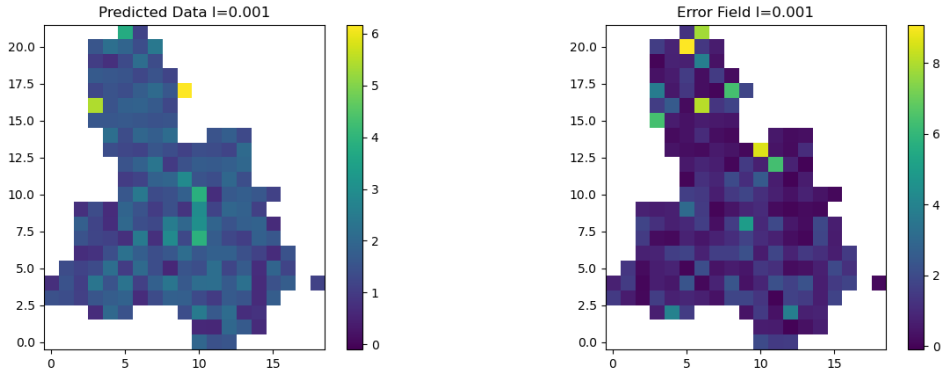
With these inferred fields in hand, we can experiment with different length scales for the GP prior. Firstly, two extreme values relative to how discretised the space is were considered ($l = 0.01$, $l = 10$). The plots in figures (11), (12) and (13) show the true bike theft data, the subsampled data used for training, as well as the predicted fields and their respective errors for each length scale.



(a) True Data

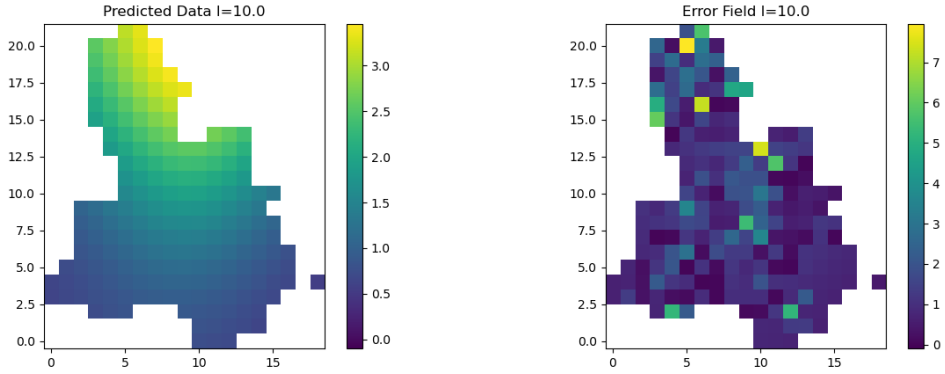
(b) Subsampled Data

Figure 11: True Data



(a) Predicted field (b) Error field

Figure 12: $l = 0.01$



(a) Predicted field (b) Error field

Figure 13: $l = 10$

For very short length scales, the predicted field is mostly just noise - there are peaks that correspond to observed peaks in the subsampled data, but outside of this the surface is largely just flat with slight fluctuations. For long length scales, the model has found the largest observed count in the subsampled data and learnt a downward sloping field away from this point - however the length scale is too long, and so the field varies so slowly that the model can't model the variation away from this point.

These observations were true across several attempts using different random seeds for the subsampling - the short length scale only modelled the peaks, and the long length scale only found the largest peak.

Following this, the same grid search method as the first section was performed between $l = 0.01$ and $l = 3$, since a precursor grid search showed that the minimum error was achieved at a length scale below 3. The resulting plot is shown in figure (14) with the lowest error value of l labelled - there is a clear dip in the error around the lowest value, though the location of the dip naturally depended on the specific seed used to subsample the data, and the minimum varied from roughly $l = 0.4$ to $l = 1.6$.

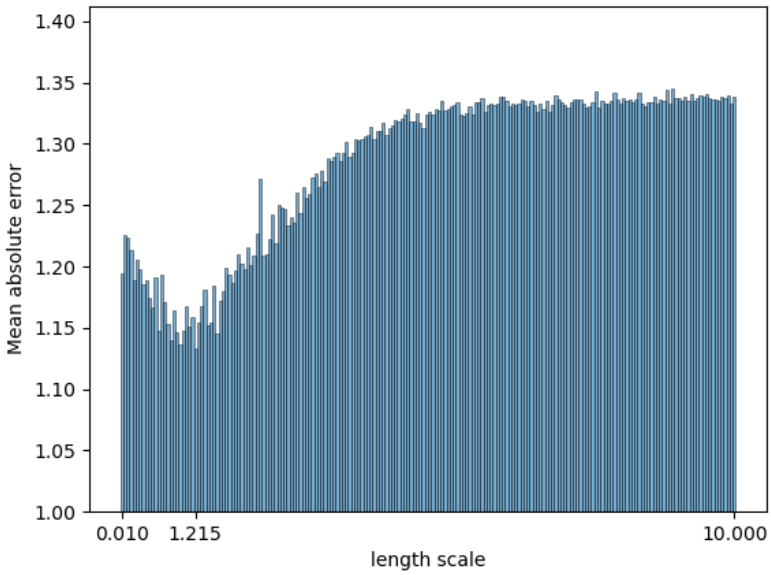
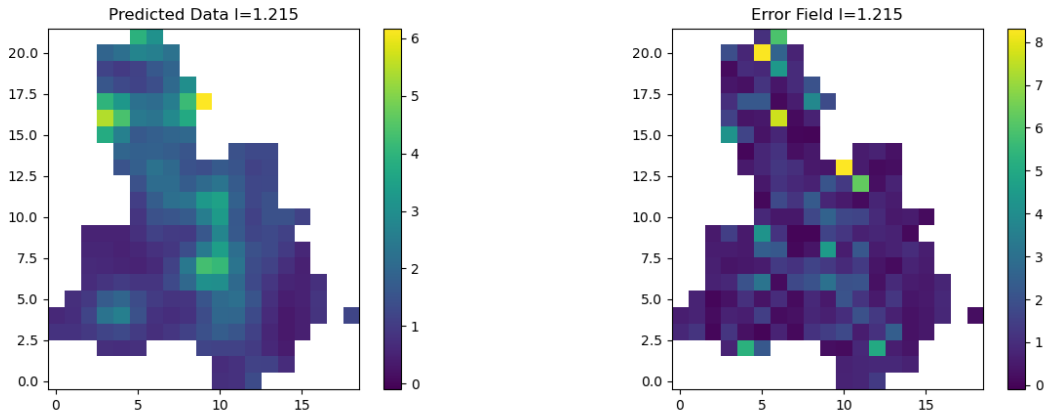


Figure 14: Heat map for spatial data

The corresponding predicted field for an example optimal length scale using seed 12 is shown in figure (15). This inferred field seems quite reasonable as there are common features identifiable between the true and predicted data at length scales larger than one square such as the vertical ridge in the centre, the peak near the top left, and the very low regions in the bottom right and the left hand side. The error field shows that spikes in the true data that were not present in the subsampled data could not be modelled, but this is to be expected as these spikes appear in isolation in the data, and cannot be well modelled by the squared exponential GP prior structure.



(a) Predicted field

(b) Error field

Figure 15: $l = 1.215$