Embarrassingly Parallel MCMC for Fast and Flexible Analysis of Spatiotemporal Data

Willie Neiswanger WILLIE@CS.CMU.EDU
Machine Learning Department, Carnegie Mellon University

Abstract

Background. Large datasets are often collected or stored in a distributed fashion over a number of machines. We would like to develop scalable Bayesian inference algorithms for these datasets. Most existing algorithms require all data to be sent to a central node for processing, or regular communication between machines during parallel processing (both of which require a great deal of communication of information or parameters between machines). Additionally, we often want inference results on many different subsets of these datasets, for example, when analyzing local portions of large spatiotemporal data; most existing methods must rerun inference algorithms for each queried subset, which can take a great deal of time when the number of queries grows large.

Aim. The aim of this project is to develop an "embarrassingly parallel" method of Bayesian inference, in which each machine performs inference on a subset of data without any communication to other machines. By doing this, we hope to be able to perform statistical learning of large, distributed datasets without needing the regular transfer of data or parameters.

Data. We demonstrate the advantages of this method on multiple real datasets. We first show the ability of this method to increase the speed of inference in Bayesian regression and mixture models. We then show the ability to use this technique to analyze spatiotemporal data; in particular, our method is able to yield inference results on arbitrarily selected subsets of data without rerunning any of our initial inference algorithms. Here, we design a generative model for taxi trip data, and run our inference method on a set of 14 million taxi trips in New York City.

Methods. We work in a Bayesian framework; hence the goal of the learning methods will be to compute a posterior distribution over model parameters. The strategy here is to perform Bayesian inference on each machine (on its subset of data) in parallel. After inference is complete, we transfer all inference results to a single machine and combine the local results to infer a global result (i.e. the posterior distribution given the entire, global dataset). More specifically, we perform Markov chain Monte Carlo (MCMC), a sampling algorithm, to conduct local inference. To combine the local results, we develop a sample combination algorithm that combines the local sample sets to produce samples from the full-data (global) posterior distribution.

Results. We develop a method to perform inference in an embarrassingly parallel manner: on each node we conduct local inference on a subset of data, and afterwards, we combine the local results to yield a global inference result. We show that our combination procedure yields asymptotically exact global inference results (i.e. our results converge to the same value as if we had used the full data on a single machine), and we use our method to analyze several large real-world datasets.

Conclusions. The goal of our project was to develop an embarrassingly parallel method of Bayesian inference, in which each machine performs inference on a subset of data without any communication to other machines. This method allows us to perform statistical learning on large, distributed datasets (e.g. spatiotemporal data, which may be split over time and space) without needing the regular transfer of data or parameters.

1. Introduction

Many large, modern datasets are collected and stored in a distributed fashion by multiple sensors or data-collecting agents. Examples of this include medical data recorded in hospitals throughout a country, weather data gathered by a collection of sensors, cell phone data collected on users' phones, and social network data generated by each member (node) of a social network.

This paper is concerned with statistical inference algorithms that can operate in these data-distributed settings. These algorithms, which operate by processing subsets of data separately and in parallel, are particularly advantageous. This is because they mitigate the need for transferring data to a central location for analysis, reduce both the memory usage and computation time of inference [14, 16], allow for continuous data collection from independently operating agents [4], and allow for sensitive data to be processed independently in secure locations (which can yield privacy guarantees [18]).

Here, we are primarily concerned with general inference procedures for latent variable (i.e. "graphical") models. For example, Markov chain Monte Carlo (MCMC) methods are popular tools for performing approximate Bayesian inference via posterior sampling. One major benefit of these techniques is that they guarantee asymptotically exact recovery of the posterior distribution as the number of posterior samples grows. However, MCMC methods may take a prohibitively long time, since for N data points, most methods must perform $O(N)$ operations to draw a sample. Furthermore, MCMC methods might require a large number of "burn-in" steps before beginning to generate representative samples. Further complicating matters is the issue that, for many big data applications, it is necessary to store and process data on multiple machines, and so MCMC must be adapted to run in these data-distributed settings.

Researchers currently tackle these problems independently, in two primary ways. To speed up sampling, multiple independent chains of MCMC can be run in parallel [24, 12, 15]; however, each chain is still run on the entire dataset, and there is no speed-up of the burn-in process (as each chain must still complete the full burn-in before generating samples). To run MCMC when data is partitioned among multiple machines, each machine can perform computation that involves a subset of the data and exchange information at each iteration to draw a sample [11, 17, 22]; however, this requires a significant amount of communication between machines, which can greatly increase computation time when machines wait for external information [1, 8].

We aim to develop a procedure to tackle both problems simultaneously, to allow for quicker burn-in and sampling in settings where data are partitioned among machines. To accomplish this, we propose the following: on each machine, run MCMC on only a subset of the data (independently, without communication between machines), and then combine the samples from each machine to algorithmically construct samples from the full-data posterior distribution. We'd like our procedure to satisfy the following four criteria:

1. Each machine only has access to a portion of the data.
2. Each machine performs MCMC independently, without communicating (i.e. the procedure is "embarrassingly parallel").
3. Each machine can use any type of MCMC to generate samples.
4. The combination procedure yields provably asymptotically exact samples from the full-data posterior.

The third criterion allows existing MCMC algorithms or software packages to be run directly on subsets of the data—the combination procedure then acts as a post-processing step to transform the samples to the correct distribution. Note that this procedure is particularly suitable for use in a MapReduce [5] framework. Also note that, unlike current strategies, this procedure does not involve multiple "duplicate" chains (as each chain uses a different portion of the data and samples from a different posterior distribution), nor does it involve parallelizing a single chain (as there are multiple chains operating independently). We will show how this allows our method to, in fact, parallelize and greatly reduce the time required for burn-in.

In this paper we will (1) introduce and define the *subposterior* density—a modified posterior given a subset of the data—which will be used heavily, (2) present methods for the embarrassingly parallel MCMC and combination procedure, (3) prove theoretical guarantees about the samples generated from our algorithm, (4) describe the current scope of the presented method (i.e. where and when it can be applied), and (5) show empirical results demonstrating that we can achieve speed-ups for burn-in and sampling while meeting the above four criteria.

2. Embarrassingly Parallel MCMC

We draw from [16] for the description of our method. The basic idea behind our method is to partition a set of N i.i.d. data points $x^N = \{x_1, \cdots, x_N\}$ into M subsets, sample from the *subposterior*—the posterior given a data subset with an underweighted prior—in parallel, and then combine the resulting samples to form samples from the full-data posterior $p(\theta|x^N)$, where $\theta \in \mathbb{R}^d$ and $p(\theta|x^N) \propto p(\theta)p(x^N|\theta) = p(\theta)\prod_{i=1}^N p(x_i|\theta)$.

More formally, given data x^N partitioned into M subsets $\{x^{n_1}, \dots, x^{n_M}\}$, the procedure is:

1. For $m = 1, \dots, M$ (in parallel):
   Sample from the subposterior p_m, where

$$p_m(\theta) \propto p(\theta)^{\frac{1}{M}} p(x^{n_m}|\theta). \tag{1}$$

2. Combine the subposterior samples to produce samples from an estimate of the subposterior density product $p_1 \cdots p_M$, which is proportional to the full-data posterior, i.e. $p_1 \cdots p_M(\theta) \propto p(\theta|x^N)$.

We want to emphasize that we do not need to iterate over these steps and the combination stage (step 3) is the only step that requires communication between machines. Also note that sampling from each subposterior (step 2) can typically be done in the same way as one would sample from the full-data posterior. For example, when using the Metropolis-Hastings algorithm, one would compute the likelihood ratio as $\frac{p(\theta^*)^{\frac{1}{M}} p(x^{n_m}|\theta^*)}{p(\theta)^{\frac{1}{M}} p(x^{n_m}|\theta)}$ instead of $\frac{p(\theta^*)p(x^N|\theta^*)}{p(\theta)p(x^N|\theta)}$, where θ^* is the proposed move. In the next section, we show how the combination stage (step 3) is carried out to generate samples from the full-data posterior using the subposterior samples.

3. Combining Subposterior Samples

Our general idea is to combine the subposterior samples in such a way that we are implicitly sampling from an estimate of the subposterior density product function $\widehat{p_1 \cdots p_M}(\theta)$. If our density product estimator is consistent, then we can show that we are drawing asymptotically exact samples from the full posterior. Further, by studying the estimator error rate, we can explicitly analyze how quickly the distribution from which we are drawing samples is converging to the true posterior (and thus compare different combination algorithms).

In the following three sections we present procedures that yield samples from different estimates of the density product. Our first example is based on a simple parametric estimator motivated by the Bernstein-von Mises theorem [13]; this procedure generates approximate (asymptotically biased) samples from the full posterior. Our second example is based on a nonparametric estimator, and produces asymptotically exact samples from the full posterior. Our third example is based on a semiparametric estimator, which combines beneficial aspects from the previous two estimators while also generating asymptotically exact samples.

3.1 Approximate posterior sampling with a parametric estimator

The first method for forming samples from the full posterior given subposterior samples involves using an approximation based on the Bernstein-von Mises (Bayesian central limit) theorem, an important result in Bayesian asymptotic theory. Assuming that a unique, true data-generating model exists and is denoted θ_0, this theorem states that the posterior tends to a normal distribution concentrated around θ_0 as the number of observations grows. In particular, under suitable regularity conditions, the posterior $P(\theta|x^N)$ is well approximated by $\mathcal{N}_d(\theta_0, F_N^{-1})$ (where F_N is the fisher information of the data) when N is large [13]. Since we aim to perform posterior sampling when the number of observations is large, a normal parametric form often serves as a good posterior approximation. A similar approximation was used in [2] in order to facilitate fast, approximately correct sampling. We therefore estimate each subposterior density with $\widehat{p}_m(\theta) = \mathcal{N}_d(\theta|\widehat{\mu}_m, \widehat{\Sigma}_m)$ where $\widehat{\mu}_m$ and $\widehat{\Sigma}_m$ are the sample mean and covariance, respectively, of the subposterior samples. The product of the M subposterior densities will be proportional to a Gaussian pdf, and our estimate of the density product function $p_1 \cdots p_M(\theta) \propto p(\theta|x^N)$ is

$$\widehat{p_1 \cdots p_M}(\theta) = \widehat{p}_1 \cdots \widehat{p}_M(\theta) \propto \mathcal{N}_d\left(\theta|\widehat{\mu}_M, \widehat{\Sigma}_M\right),$$

where the parameters of this distribution are

$$\widehat{\Sigma}_M = \left(\sum_{m=1}^{M} \widehat{\Sigma}_m^{-1}\right)^{-1} \tag{2}$$

$$\widehat{\mu}_M = \widehat{\Sigma}_M \left(\sum_{m=1}^{M} \widehat{\Sigma}_m^{-1} \widehat{\mu}_m\right). \tag{3}$$

These parameters can be computed quickly and, if desired, online (as new subposterior samples arrive).

3.2 Asymptotically exact posterior sampling with nonparametric density product estimation

In the previous method we made a parametric assumption based on the Bernstein-von Mises theorem, which allows us to generate approximate samples from the full posterior. Although this parametric estimate has quick convergence, it generates asymptotically biased samples, especially in cases where the posterior is particularly non-Gaussian. In this section, we develop a procedure that implicitly samples from the product of nonparametric density estimates, which allows us to produce asymptotically exact samples from the full posterior. By constructing a consistent density product estimator from which we can generate samples, we ensure that the distribution from which we are sampling converges to the full posterior.

Given T samples[1] $\{\theta_{t_m}^m\}_{t_m=1}^T$ from a subposterior p_m, we can write the kernel density estimator $\widehat{p}_m(\theta)$ as,

$$\begin{aligned}\widehat{p}_m(\theta) &= \frac{1}{T}\sum_{t_m=1}^{T} \frac{1}{h^d} K\left(\frac{\|\theta - \theta_{t_m}^m\|}{h}\right) \\ &= \frac{1}{T}\sum_{t_m=1}^{T} \mathcal{N}_d(\theta|\theta_{t_m}^m, h^2 I_d),\end{aligned}$$

where we have used a Gaussian kernel with bandwidth parameter h. After we have obtained the kernel density estimator $\widehat{p}_m(\theta)$ for M subposteriors, we define our nonparametric density product

1. For ease of description, we assume each machine generates the same number of samples, T. In practice, they do not have to be the same.

estimator for the full posterior as

$$
\begin{aligned}
\widehat{p_1 \cdots p_M}(\theta) &= \widehat{p}_1 \cdots \widehat{p}_M(\theta) \\
&= \frac{1}{T^M} \prod_{m=1}^{M} \sum_{t_m=1}^{T} \mathcal{N}_d(\theta | \theta^m_{t_m}, h^2 I_d) \\
&\propto \sum_{t_1=1}^{T} \cdots \sum_{t_M=1}^{T} w_{t\cdot} \mathcal{N}_d \left(\theta \middle| \bar{\theta}_{t\cdot}, \frac{h^2}{M} I_d \right). \qquad (4)
\end{aligned}
$$

This estimate is the probability density function (pdf) of a mixture of T^M Gaussians with *unnormalized* mixture weights $w_{t\cdot}$. Here, we use $t\cdot = \{t_1, \ldots, t_M\}$ to denote the set of indices for the M samples $\{\theta^1_{t_1}, \ldots, \theta^M_{t_M}\}$ (each from a separate machine) associated with a given mixture component, and we define

$$
\bar{\theta}_{t\cdot} = \frac{1}{M} \sum_{m=1}^{M} \theta^m_{t_m} \qquad (5)
$$

$$
w_{t\cdot} = \prod_{m=1}^{M} \mathcal{N}_d \left(\theta^m_{t_m} | \bar{\theta}_{t\cdot}, h^2 I_d \right). \qquad (6)
$$

Although there are T^M possible mixture components, we can efficiently generate samples from this mixture by first sampling a mixture component (based on its unnormalized component weight $w_{t\cdot}$) and then sampling from this (Gaussian) component. In order to sample mixture components, we use an independent Metropolis within Gibbs (IMG) sampler. This is a form of MCMC, where at each step in the Markov chain, a single dimension of the current state is proposed (i.e. sampled) independently of its current value (while keeping the other dimensions fixed) and then is accepted or rejected. In our case, at each step, a new mixture component is proposed by redrawing one of the M current sample indices $t_m \in t\cdot$ associated with the component uniformly and then accepting or rejecting the resulting proposed component based on its mixture weight. We give the IMG algorithm for combining subposterior samples in Algorithm 1.[2]

In certain situations, Algorithm 1 may have a low acceptance rate and therefore may mix slowly. One way to remedy this is to perform the IMG combination algorithm multiple times, by first applying it to groups of $\tilde{M} < M$ subposteriors and then applying the algorithm again to the output samples from each initial application. For example, one could begin by applying the algorithm to all $\frac{M}{2}$ pairs (leaving one subposterior alone if M is odd), then repeating this process—forming pairs and applying the combination algorithm to pairs only—until there is only one set of samples remaining, which are samples from the density product estimate.

3.3 Asymptotically exact posterior sampling with semiparametric density product estimation

Our first example made use of a parametric estimator, which has quick convergence, but may be asymptotically biased, while our second example made use of a nonparametric estimator, which is asymptotically exact, but may converge slowly when the number of dimensions is large. In this example, we implicitly sample from a semiparametric density product estimate, which allows us to leverage the fact that the full posterior has a near-Gaussian form when the number of observations is large, while still providing an asymptotically unbiased estimate of the posterior density, as the number of subposterior samples $T \to \infty$.

We make use of a semiparametric density estimator for p_m that consists of the product of a parametric estimator $\widehat{f}_m(\theta)$ (in our case $\mathcal{N}_d(\theta | \widehat{\mu}_m, \widehat{\Sigma}_m)$ as above) and a nonparametric estimator $\widehat{r}(\theta)$ of the correction function $r(\theta) = p_m(\theta)/\widehat{f}_m(\theta)$ [7]. This estimator gives a near-Gaussian estimate when the number of samples is small, and converges to the true density as the number of samples

2. Again for simplicity, we assume that we generate T samples to represent the full posterior, where T is the number of subposterior samples from each machine.

Algorithm 1 Asymptotically Exact Sampling via Nonparametric Density Product Estimation

```
Input: Subposterior samples: $\{\theta^1_{t_1}\}_{t_1=1}^T \sim p_1(\theta)$, ..., $\{\theta^M_{t_M}\}_{t_M=1}^T \sim p_M(\theta)$
Output: Posterior samples (asymptotically, as $T \to \infty$): $\{\theta_i\}_{i=1}^T \sim p_1 \cdots p_M(\theta) \propto p(\theta|x^N)$
1:  Draw $t\cdot = \{t_1, \ldots, t_M\} \overset{\text{iid}}{\sim} \text{Unif}(\{1, \ldots, T\})$
2:  for $i = 1$ to $T$ do
3:      Set $h \leftarrow i^{-1/(4+d)}$
4:      for $m = 1$ to $M$ do
5:          Set $c\cdot \leftarrow t\cdot$
6:          Draw $c_m \sim \text{Unif}(\{1, \ldots, T\})$
7:          Draw $u \sim \text{Unif}([0, 1])$
8:          if $u < w_{c\cdot}/w_{t\cdot}$ then
9:              Set $t\cdot \leftarrow c\cdot$
10:         end if
11:     end for
12:     Draw $\theta_i \sim \mathcal{N}_d(\bar{\theta}_{t\cdot}, \frac{h^2}{M} I_d)$
13: end for
```

grows. Given T samples $\{\theta^m_{t_m}\}_{t_m=1}^T$ from a subposterior p_m, we can write the estimator as

$$\begin{aligned}
\widehat{p}_m(\theta) &= \widehat{f}_m(\theta)\widehat{r}(\theta) \\
&= \frac{1}{T}\sum_{t_m=1}^T \frac{1}{h^d} K\left(\frac{\|\theta - \theta^m_{t_m}\|}{h}\right) \frac{\widehat{f}_m(\theta)}{\widehat{f}_m(\theta^m_{t_m})} \\
&= \frac{1}{T}\sum_{t_m=1}^T \frac{\mathcal{N}_d(\theta|\theta^m_{t_m}, h^2 I_d)\mathcal{N}_d(\theta|\widehat{\mu}_m, \widehat{\Sigma}_m)}{\mathcal{N}_d(\theta^m_{t_m}|\widehat{\mu}_m, \widehat{\Sigma}_m)},
\end{aligned}$$

where we have used a Gaussian kernel with bandwidth parameter h for the nonparametric component of this estimator. Therefore, we define our semiparametric density product estimator to be

$$\begin{aligned}
\widehat{p_1 \cdots p_M}(\theta) &= \widehat{p}_1 \cdots \widehat{p}_M(\theta) \\
&= \frac{1}{T^M}\prod_{m=1}^M \sum_{t_m=1}^T \frac{\mathcal{N}_d(\theta|\theta^m_{t_m}, hI_d)\mathcal{N}_d(\theta|\widehat{\mu}_m, \widehat{\Sigma}_m)}{h^d \mathcal{N}_d(\theta^m_{t_m}|\widehat{\mu}_m, \widehat{\Sigma}_m)} \\
&\propto \sum_{t_1=1}^T \cdots \sum_{t_M=1}^T W_{t\cdot} \mathcal{N}_d\left(\theta|\mu_{t\cdot}, \Sigma_{t\cdot}\right).
\end{aligned}$$

This estimate is proportional to the pdf of a mixture of T^M Gaussians with unnormalized mixture weights,

$$W_{t\cdot} = \frac{w_{t\cdot}\, \mathcal{N}_d\left(\bar{\theta}_{t\cdot}|\widehat{\mu}_M, \widehat{\Sigma}_M + \frac{h}{M} I_d\right)}{\prod_{m=1}^M \mathcal{N}_d(\theta^m_{t_m}|\widehat{\mu}_m, \widehat{\Sigma}_m)},$$

where $\bar{\theta}_{t\cdot}$ and $w_{t\cdot}$ are given in Eqs. 5 and 6. We can write the parameters of a given mixture component $\mathcal{N}_d(\theta|\mu_{t\cdot}, \Sigma_{t\cdot})$ as

$$\begin{aligned}
\Sigma_{t\cdot} &= \left(\frac{M}{h} I_d + \widehat{\Sigma}_M^{-1}\right)^{-1}, \\
\mu_{t\cdot} &= \Sigma_{t\cdot}\left(\frac{M}{h} I_d \bar{\theta}_{t\cdot} + \widehat{\Sigma}_M^{-1}\widehat{\mu}_M\right),
\end{aligned}$$

where $\widehat{\mu}_M$ and $\widehat{\Sigma}_M$ are given by Eq. 2 and 3. We can sample from this semiparametric estimate using the IMG procedure outlined in Algorithm 1, replacing the component weights $w_{t\cdot}$ with $W_{t\cdot}$ and the component parameters $\bar{\theta}_{t\cdot}$ and $\frac{h}{M}I_d$ with $\mu_{t\cdot}$ and $\Sigma_{t\cdot}$.

We also have a second semiparametric procedure that may give higher acceptance rates in the IMG algorithm. We follow the above semiparametric procedure, where each component is a normal distribution with parameters $\mu_{t\cdot}$ and $\Sigma_{t\cdot}$, but we use the nonparametric component weights $w_{t\cdot}$ instead of $W_{t\cdot}$. This procedure is also asymptotically exact, since the semiparametric component parameters $\mu_{t\cdot}$ and $\Sigma_{t\cdot}$ approach the nonparametric component parameters $\bar{\theta}_{t\cdot}$ and $\frac{h}{M}I_d$ as $h \to 0$, and thus this procedure tends to the nonparametric procedure given in Algorithm 1.

4. Method Complexity and Scope

Given M data subsets, to produce T samples in d dimensions with the nonparametric or semiparametric asymptotically exact procedures (Algorithm 1) requires $O(dTM^2)$ operations. The variation on this algorithm that performs this procedure $M-1$ times on pairs of subposteriors (to increase the acceptance rate; detailed in Section 3.2) instead requires only $O(dTM)$ operations.

We have presented our method as a two step procedure, where first parallel MCMC is run to completion, and then the combination algorithm is applied to the M sets of samples. We can instead perform an online version of our algorithm: as each machine generates a sample, it immediately sends it to a master machine, which combines the incoming samples[3] and performs the accept or reject step (Algorithm 1, lines 3-12). This allows the parallel MCMC phase and the combination phase to be performed in parallel, and does not require transfering large volumes of data, as only a single sample is ever transferred at a time.

The total communication required by our method is transferring $O(dTM)$ scalars (T samples from each of M machines), and as stated above, this can be done online as MCMC is being carried out. Further, the communication is unidirectional, and each machine does not pause and wait for any information from other machines during the parallel sampling procedure.

The algorithms in this paper hold for posterior distributions over finite-dimensional real spaces. These include generalized linear models (e.g. linear, logistic, or Poisson regression), mixture models with known weights, hierarchical models, and (more generally) finite-dimensional graphical models with unconstrained variables. This also includes both unimodal and multimodal posterior densities (such as in Section 6.4). However, the methods and theory presented here do not yet extend to cases such as infinite dimensional models (e.g. nonparametric Bayesian models [6]) nor to distributions over the simplex (e.g. topics in latent Dirichlet allocation [3]). In the future, we hope to extend this work to these domains.

5. Related Work

In [23, 2, 20], the authors develop a way to sample approximately from a posterior distribution when only a small randomized mini-batch of data is used at each step. In [10], the authors used a hypothesis test to decide whether to accept or reject proposals using a small set of data (adaptively) as opposed to the exact Metropolis-Hastings rule. This reduces the amount of time required to compute the acceptance ratio. Since all of these algorithms are still sequential, they can be directly used in our algorithm to generate subposterior samples to further speed up the entire sampling process.

Several parallel MCMC algorithms have been designed for specific models, such as for topic models [22, 17] and nonparametric mixture models [25]. These approaches still require synchronization to be correct (or approximately correct), while ours aims for more general model settings and does not need synchronization until the final combination stage.

Consensus Monte Carlo [21] is perhaps the most relevant work to ours. In this algorithm, data is also portioned into different machines and MCMC is performed independently on each machine. Thus, it roughly has the same time complexity as our algorithm. However, the prior is not explicitly

3. For the semiparametric method, this will involve an online update of mean and variance Gaussian parameters.

reweighted during sampling as we do in Eq 1, and final samples for the full posterior are generated by averaging subposterior samples. Furthermore, this algorithm has few theoretical guarantees. We find that this algorithm can be viewed as a relaxation of our nonparametric, asymptotically exact sampling procedure, where samples are generated from an evenly weighted mixture (instead of each component having weight $w_{t\cdot}$) and where each sample is set to $\bar{\theta}_{t\cdot}$ instead of being drawn from $\mathcal{N}\left(\bar{\theta}_{t\cdot}, \frac{h}{M} I_d\right)$. This algorithm is one of our experimental baselines.

6. Experiments

In the following sections, we demonstrate empirically that our method allows for quicker, MCMC-based estimation of a posterior distribution, and that our consistent-estimator-based procedures yield asymptotically exact results. We furthere show that our methods allow for analysis of selected subsets of data, which is of particular use when applied to spatiotemporal data.

6.1 Data and Models

We apply our method to a few Bayesian models using both synthetic and real data, where we show both quantitative comparisons of the accuracy and performance of our method, as well as exploratory analysis, showing insights into real-world large-scale datasets. We perform experiments on the following models and data:

1. A **Bayesian logistic regression model** applied to synthetic data drawn according to the generative process of the assumed model.
2. A **Bayesian logistic regression model** applied to data from a forest covertype prediction challenge.
3. A **hierarchical Poisson-gamma regression model** applied to synthetic data drawn according to the generative process of the assumed model.
4. A **Gaussian mixture model** applied to synthetic data drawn according to the generative process of the assumed model.
5. A **hidden segments mixture model** of taxi records applied to 14 million New York City taxi trip records.

6.2 Comparison Methods and Performance Evaluation

In each experiment, we compare the following strategies for parallel, communication-free sampling:[4]

- **Single chain full-data posterior samples** (`regularChain`)—Typical, single-chain MCMC for sampling from the full-data posterior.
- **Parametric subposterior density product estimate** (`parametric`)—For M sets of subposterior samples, the combination yielding samples from the parametric density product estimate.
- **Nonparametric subposterior density product estimate** (`nonparametric`)—For M sets of subposterior samples, the combination yielding samples from the nonparametric density product estimate.
- **Semiparametric subposterior density product estimate** (`semiparametric`)—For M sets of subposterior samples, the combination yielding samples from the semiparametric density product estimate.
- **Subposterior sample average** (`subpostAvg`)—For M sets of subposterior samples, the average of M samples consisting of one sample taken from each subposterior.
- **Subposterior sample pooling** (`subpostPool`)—For M sets of subposterior samples, the union of all sets of samples.

4. We did not directly compare with the algorithms that require synchronization since the setup of these experiments can be rather different.

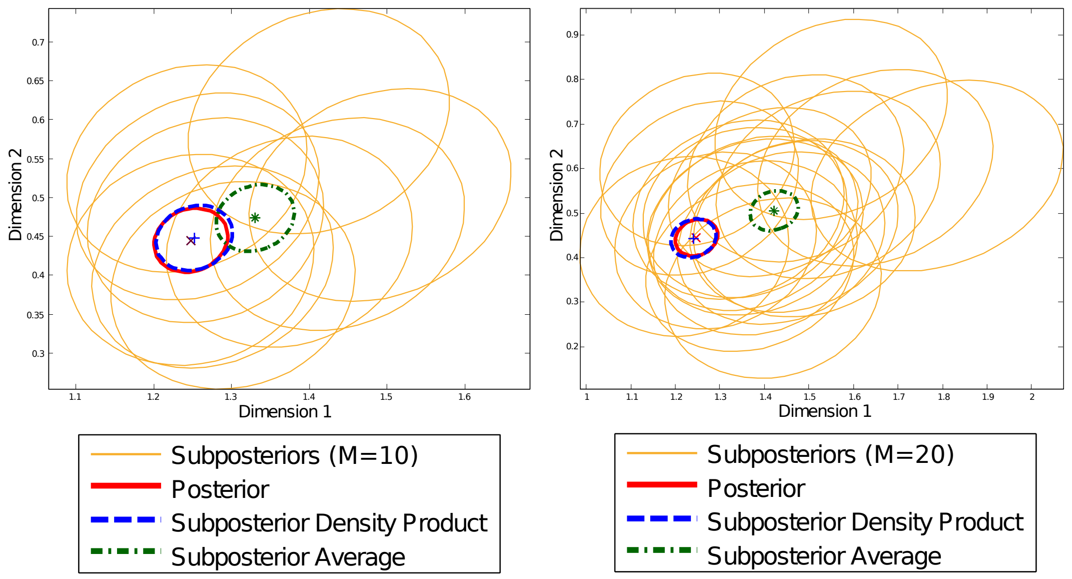

Figure 1: Bayesian logistic regression posterior ovals. We show the posterior 90% probability mass ovals for the first 2-dimensional marginal of the posterior, the M subposteriors, the subposterior density product (via the `parametric` procedure), and the subposterior average (via the `subpostAvg` procedure). We show $M{=}10$ subsets (left) and $M{=}20$ subsets (right). The subposterior density product generates samples that are consistent with the true posterior, while the `subpostAvg` produces biased results, which grow in error as M increases.

- **Duplicate chains full-data posterior sample pooling** (`duplicateChainsPool`)—For M sets of samples from the full-data posterior, the union of all sets of samples.

To assess the performance of our sampling and combination strategies, we ran a single chain of MCMC on the full data for 500,000 iterations, removed the first half as burn-in, and considered the remaining samples the "groundtruth" samples for the true posterior density. We then needed a general method to compare the distance between two densities given samples from each, which holds for general densities (including multimodal densities, where it is ineffective to compare moments such as the mean and variance[5]). Following work in density-based regression [19], we use an estimate of the L_2 distance, $d_2(p,\hat{p})$, between the groundtruth posterior density p and a proposed posterior density $\widehat{p}$, where $d_2(p,\hat{p}) = \|p-\widehat{p}\|_2 = \left(\int(p(\theta)-\widehat{p}(\theta))^2 d\theta\right)^{1/2}$.

In the following experiments involving timing, to compute the posterior L_2 error at each time point, we collected all samples generated before a given number of seconds, and added the time taken to transfer the samples and combine them using one of the proposed methods. In all experiments and methods, we followed a fixed rule of removing the first $\frac{1}{6}$ samples for burn-in (which, in the case of combination procedures, was applied to each set of subposterior samples before the combination was performed).

Experiments were conducted with a standard cluster system. We obtained subposterior samples by submitting batch jobs to each worker since these jobs are all independent. We then saved the results to the disk of each worker and transferred them to the same machine which performed the final combination.

6.3 Generalized Linear Models

Generalized linear models are widely used for many regression and classification problems. Here we conduct experiments, using logistic regression as a test case, on both synthetic and real data to demonstrate the speed of our parallel MCMC algorithm compared with typical MCMC strategies.

5. In these cases, dissimilar densities might have similar low-order moments. See Section 6.4 for an example.

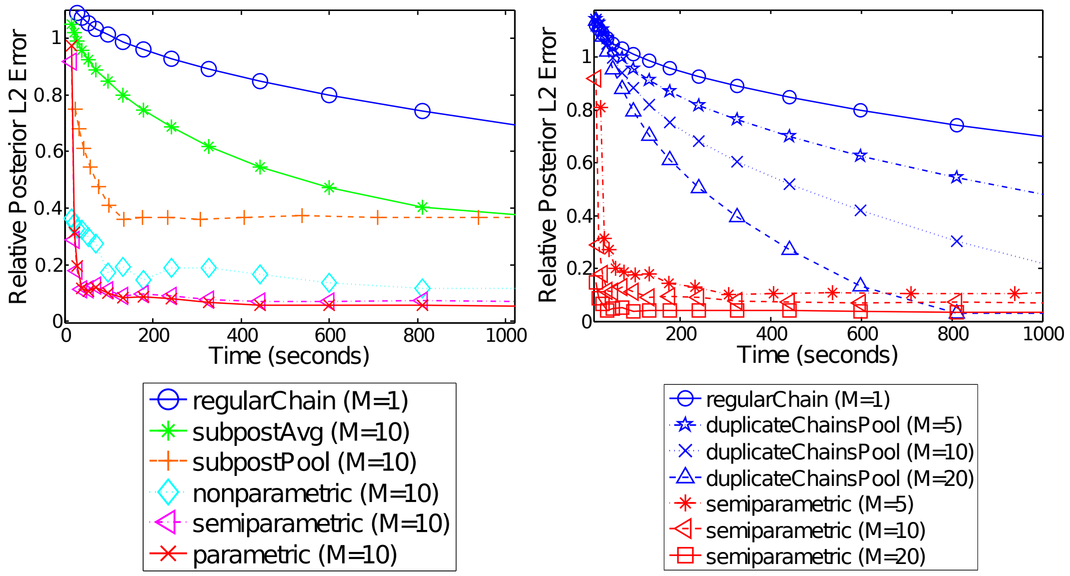

Figure 2: Posterior L_2 error vs time for logistic regression. Left: the three combination strategies proposed in this paper (`parametric`, `nonparametric`, and `semiparametric`) reduce the posterior error much more quickly than a single full-data Markov chain; the `subpostAvg` and `subpostPool` procedures yield biased results. Right: we compare with multiple full-data Markov chains (`duplicateChainsPool`); our method yields faster convergence to the posterior even though only a fraction of the data is being used by each chain.

6.3.1 Synthetic data

Our synthetic dataset contains 50,000 observations in 50 dimensions. To generate the data, we drew each element of the model parameter β and data matrix X from a standard normal distribution, and then drew each outcome as $y_i \sim \text{Bernoulli}(\text{logit}^{-1}(X_i\beta))$ (where X_i denotes the i^{th} row of X)[6]. We use Stan, an automated Hamiltonian Monte Carlo (HMC) software package,[7] to perform sampling for both the true posterior (for groundtruth and comparison methods) and for the subposteriors on each machine. One advantage of Stan is that it is implemented with C++ and uses the No-U-Turn sampler for HMC, which does not require any user-provided parameters [9].

In Figure 1, we illustrate results for logistic regression, showing the subposterior densities, the subposterior density product, the subposterior sample average, and the true posterior density, for the number of subsets M set to 10 (left) and 20 (right). Samples generated by our approach (where we draw samples from the subposterior density product via the `parametric` procedure) overlap with the true posterior much better than those generated via the `subpostAvg` (subposterior sample average) procedure— averaging of samples appears to create systematic biases. Futher, the error in averaging appears to increase as M grows. In Figure 2 (left) we show the posterior error vs time. A regular full-data chain takes much longer to converge to low error compared with our combination methods, and simple averaging and pooling of subposterior samples gives biased solutions.

We next compare our combination methods with multiple independent "duplicate" chains each run on the full dataset. Even though our methods only require a fraction of the data storage on each machine, we are still able to achieve a significant speed-up over the full-data chains. This is primarily because the duplicate chains cannot parallelize burn-in (i.e. each chain must still take some n steps before generating reasonable samples, and the time taken to reach these n steps does not decrease as more machines are added). However, in our method, each subposterior sampler can take each step more quickly, effectively allowing us to decrease the time needed for burn-in as we increase M. We

6. Note that we did not explicitly include the intercept term in our logistic regression model.
7. http://mc-stan.org

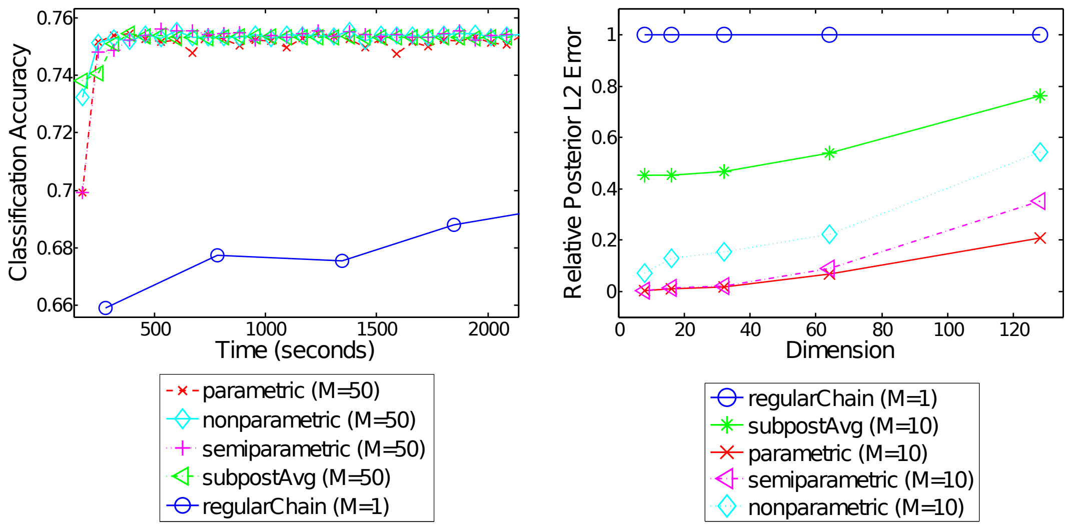

Figure 3: Left: Bayesian logistic regression classification accuracy vs time for the task of predicting forest cover type. Right: Posterior error vs dimension on synthetic data at 1000 seconds, normalized so that `regularChain` error is fixed at 1.

show this empirically in Figure 2 (right), where we plot the posterior error vs time, and compare with full duplicate chains as M is increased.

Using a Matlab implementation of our combination algorithms, all (batch) combination procedures take under twenty seconds to complete on a 2.5GHz Intel Core i5 with 16GB memory.

6.3.2 Real-world data

Here, we use the *covtype* (predicting forest cover types)[8] dataset, containing 581,012 observations in 54 dimensions. A single chain of HMC running on this entire dataset takes an average of 15.76 minutes per sample; hence, it is infeasible to generate groundtruth samples for this dataset. Instead we show classification accuracy vs time. For a given set of samples, we perform classification using a sample estimate of the posterior predictive distribution for a new label y with associated datapoint x, i.e.

$$\begin{aligned} P(y|x, y^N, x^N) &= \int P(y|x, \beta, y^N, x^N) P(\beta|x^N, y^N) \\ &\approx \frac{1}{S} \sum_{s=1}^{S} P(y|x, \beta_s) \end{aligned}$$

where x^N and y^N denote the N observations, and $P(y|x, \beta_s) = \text{Bernoulli}(\text{logit}^{-1}(x^\top \beta_s))$. Figure 3 (left) shows the results for this task, where we use M=50 splits. The parallel methods achieve a higher accuracy much faster than the single-chain MCMC algorithm.

6.3.3 Scalability with dimension

We investigate how the errors of our methods scale with dimensionality, to compare the different estimators implicit in the combination procedures. In Figure 3 (right) we show the relative posterior error (taken at 1000 seconds) vs dimension, for the synthetic data with M=10 splits. The errors at each dimension are normalized so that the `regularChain` error is equal to 1. Here, the `parametric` (asymptotically biased) procedure scales best with dimension, and the `semiparametric` (asymptotically exact) procedure is a close second. These results also demonstrate that, although the

8. http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets

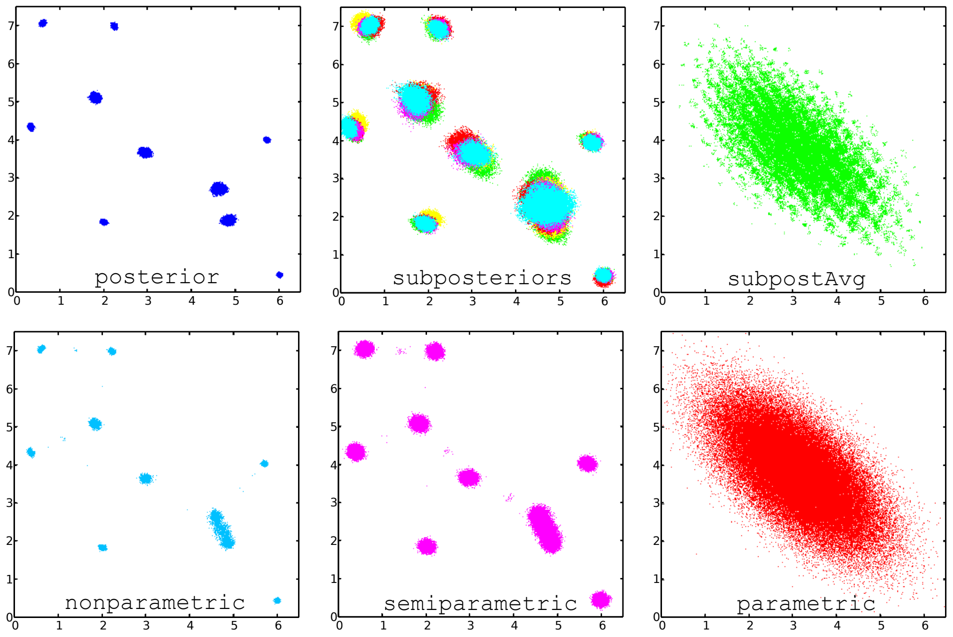

Figure 4: Gaussian mixture model posterior samples. We show 100,000 samples from a single 2-d marginal (corresponding to the posterior over a single mean parameter) of the full-data posterior (top left), all subposteriors (top middle—each one is given a unique color), the subposterior average via the `subpostAvg` procedure (top right), and the subposterior density product via the `nonparametric` procedure (bottom left), `semiparametric` procedure (bottom middle), and `parametric` procedure (bottom right).

`nonparametric` method can be viewed as implicitly sampling from a nonparametric density estimate (which is usually restricted to low-dimensional densities), the performance of our method does not suffer greatly when we perform parallel MCMC on posterior distributions in much higher-dimensional spaces.

6.4 Gaussian mixture models

In this experiment, we aim to show correct posterior sampling in cases where the full-data posterior, as well as the subposteriors, are multimodal. We will see that the combination procedures that are asymptotically biased suffer greatly in these scenarios. To demonstrate this, we perform sampling in a Gaussian mixture model. We generate 50,000 samples from a ten component mixture of 2-d Gaussians. The resulting posterior is multimodal; this can be seen by the fact that the component labels can be arbitrarily permuted and achieve the same posterior value. For example, we find after sampling that the posterior distribution over each component mean has ten modes. To sample from this multimodal posterior, we used the Metropolis-Hastings algorithm, where the component labels were permuted before each step (note that this permutation results in a move between two points in the posterior distribution with equal probability).

In Figure 4 we show results for $M=10$ splits, showing samples from the true posterior, overlaid samples from all five subposteriors, results from averaging the subposterior samples, and the results after applying our three subposterior combination procedures. This figure shows the 2-d marginal of the posterior corresponding to the posterior over a single mean component. The `subpostAvg` and `parametric` procedures both give biased results, and cannot capture the multimodality of the posterior. We show the posterior error vs time in Figure 5 (left), and see that our asymptotically exact methods yield quick convergence to low posterior error.

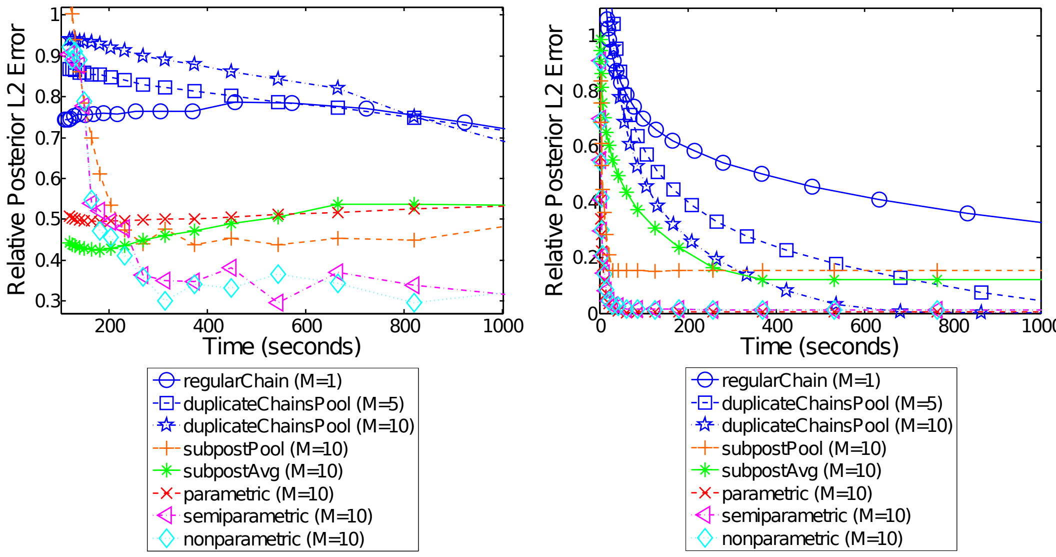

Figure 5: Left: Gaussian mixture model posterior error vs time results. Right: Poisson-gamma hierarchical model posterior error vs time results.

6.5 Hierarchical models

We show results on a hierarchical Poisson-gamma model of the following form

$$\begin{aligned}
a &\sim \text{Exponential}(\lambda) \\
b &\sim \text{Gamma}(\alpha, \beta) \\
q_i &\sim \text{Gamma}(a, b) \text{ for } i = 1, \ldots, N \\
x_i &\sim \text{Poisson}(q_i t_i) \text{ for } i = 1, \ldots, N
\end{aligned}$$

for N=50,000 observations. We draw $\{x_i\}_{i=1}^N$ from the above generative process (after fixing values for a, b, λ, and $\{t_i\}_{i=1}^N$), and use M=10 splits. We again perform MCMC using the Stan software package.

In Figure 5 (right) we show the posterior error vs time, and see that our combination methods complete burn-in and converge to a low posterior error very quickly relative to the `subpostAvg` and `subpostPool` procedures and full-data chains.

7. Analyzing New York City Taxi Records

Our final experiment involves a large-scale exploratory analysis of spatiotemporal transportation-flow data, which allows users to flexibly analyze traffic patterns at arbitrarily chosen portions of, for example, the day, week, and year. To achieve this analysis, we design a graphical model for this data and use it in a large scale distributed inference experiment.

Our data consisted of New York City taxi records from the year 2015 [9]. Each taxi record consists of a time and location for both pickup (i.e. start of the trip) and dropoff (i.e. end of the trip.). There are roughly 14 million taxi trips each month and 170 million total trips in 2015. Taxi locations range over the five New York City Boroughs, as well as some nearby locations in New Jersey (and, in rare cases, other states).

In this experiment we aim to demonstate the ability of our method to allow for flexible analysis of spatiotemporal data. In particular, we aim to show that our embarrassingly parallel MCMC methods can allow for analysis of arbitrarily chosen subsets of a large datset *after* inference has already been completed. As an example use case, consider the taxi records datset. This is a time-varying dataset,

9. `http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml`

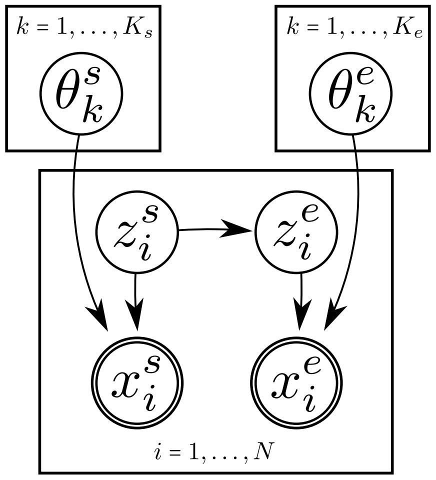

Figure 6: Graphical model for the hidden segments mixture model (HSMM) for taxi records.

and we can use this temporal component to split up the data into groups. In this case, we split our data into fine grained groups of taxi trips with pickup times occuring in the same hour (for example, in the Month of February of 2015 alone, this yields 672 groups of data). We can then perform MCMC on each subset (i.e. each hour) of data. In previous experiments, we have been interested in combining all locally inferred models. Here, we are interested in taking any subset of models (i.e. any arbitrary group of hours) and selectively combining those. By doing this, we can yield an inference result on queries such as the flow of traffic during weekdays versus weekends, or in morning rush hours versus evening rush hours; we can also aim to analyze traffic patterns on days with certain, for example, weather events (i.e. hours where it was sunny versus stormy) or in the presence of other covariates.

We next describe our model. Denote the i^{th} taxi trip pickup location as x_i^s and dropoff location as x_i^e. In this model, we assume that pickup and dropoff locations tend to cluster at common locations in the city; we will refer to these clusters as "hubs" or "transportation hubs". Our model includes an independent set of hubs for both pickup locations and dropoff locations. Let z_i^s be an assignment variable for pickup location x_i^s, which labels the identity of the associated pickup hub for the i^{th} record, and let z_i^e be a similar assignment variable for dropoff x_i^e. Further, assume there are K_s pickup hubs (i.e. $z_i^s \in \{1, \ldots, K_s\}$) and K_e dropoff hubs (i.e. $z_i^e \in \{1, \ldots, K_e\}$). Additionally, let θ_k^s denote the parameters of the emission distribution f_s for the k^{th} pickup hub (i.e. parameters of the distribution f_s that generates the x_i^s assigned to pickup hub k) and let θ_k^e denote the parameters of the emission distribution f_e for the k^{th} dropoff hub (i.e. parameters of the distribution f_e that generates the x_i^e associated with pickup hub k). Finally, assume there is an appropriately normalized transition matrix T that dictates the transition, or relationship, between a taxi record's pickup hub and its dropoff hub; more specifically, the entry T_{k_1,k_2} in this matrix will dictate the probability of transitioning to dropoff hub k_2 from pickup hub k_1.

The generative process of our model, which we call a hidden segments mixture model (HSMM) for taxi records $i = 1, \ldots, N$, can be written as following:

$$\begin{aligned}
\theta_k^s &\sim g_s(\alpha_s), \text{ for } k = \{1, \ldots, K\} \\
\theta_k^e &\sim g_e(\alpha_e), \text{ for } k = \{1, \ldots, K\} \\
z_i^s &\sim \text{Uniform}(\{1, \ldots, K\}) \\
z_i^e &\sim \text{Categorical}(T_{z_i^s,:}) \\
x_i^s &\sim f_s\left(\theta_{z_i^s}^s\right) \\
x_i^e &\sim f_e\left(\theta_{z_i^s}^s\right)
\end{aligned}$$

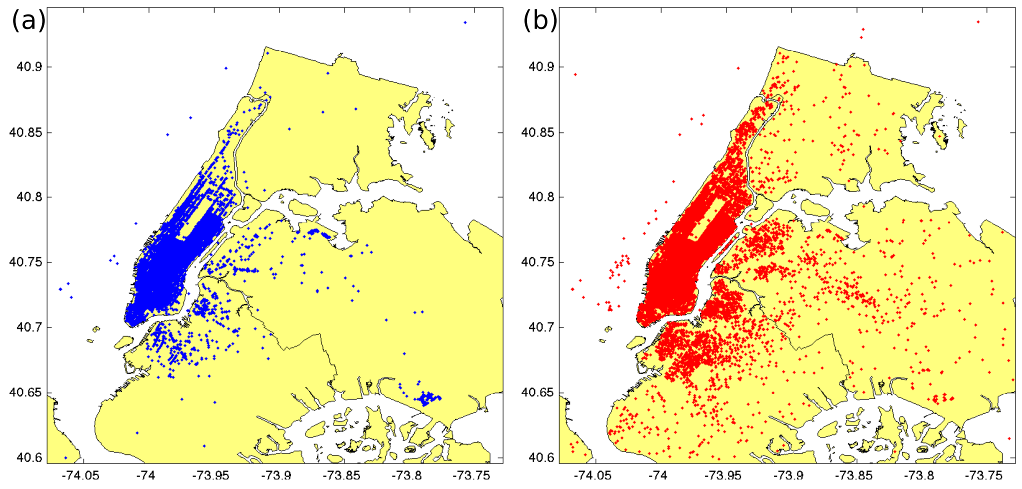

Figure 7: One hour of (a) taxi pickup locations in blue, and (b) taxi dropoff locations in red.

where g_s and g_e are, respectively, prior distributions over pickup hub and dropoff hub parameters (and are parameterized by α_s and α_e) and $T_{k,:}$ denotes the k^{th} row of transition matrix T. In the following experiments, we will assume very simple emission distributions for both taxi pickups and dropoffs: we choose f_s and f_e to be Gaussian, and hub parameter priors g_s and g_e to be Normal-Wishart. Furthermore, we fix $K = 30$ through all experiments; this value was chosen via visual inspection of inference results on a subset of the data. A graphical depiction of this model is drawn in Figure 6. When performing the combination procedure in this model, we use the parametric combination strategy (Section 3.1), which yields better results given the highly multimodal posterior density landscape in this latent variable model.

We plot one hour of this data in Figure 7. Plot (a) shows all pickup locations in blue, and plot (b) shows all dropoff locations in red. There are slightly over 30,000 taxi records in this hour alone. On individual groups of data (i.e. over individual hours), we perform inference in this model using Gibbs sampling. We run each of these chains of MCMC for 50,000 steps, and then thin and randomly permute the resulting samples. Our Gibbs sampling implementation takes on the order of one hour to yield these samples. We also run our combination algorithm for 50,000 steps; in the following visualizations, we plot the parameters yielded by the final step of this algorithm (we are therefore plotting an approximate point estimate, as it is difficult to visualize a posterior distribution over the parameters of this model).

In this set of experiments, we combine selective subsets of the inferred local (hourly) results to analyze taxi traffic flow; in particular, we determine pickup hubs, dropoff hubs, and transition probabilities between the hubs, at different times of the day and week, over the course of the month of February 2015. We show results in Figure 8. In each plot, we overlay the found pickup hubs (blue crosses) and dropoff hubs (red circles) on top of the outlines of the five New York City boroughs (note that some of the hubs are positioned outside of the boroughs in New Jersey). To show the transition probabilities between the hubs, we plot arrows between hubs for the 40 highest weighted entries of the transition matrix T; these arrows are colored based on their weights, with a darker arrow corresponding to a stronger transition between hubs. As an example, we show an inference result for a single hour (17:00-18:00 on Thursday, February 2015) in Figure 8 (d); Note that this result is yielded directly from local inference, without any futher combination.

In Figure 8, plot (a) shows the inference result of combining all possible hours (i.e. the model posterior over the full dataset). We see here that the hubs are fairly evenly distributed over Manhattan, Brooklyn, and Queens (with a concentration along the East and West side of Manhattan), and a small collection of pickup hubs in New Jersey. We furthermore see that the transitions between hubs concentrate in Manhattan and East Queens, but are all fairly homogeneous (without a large pattern

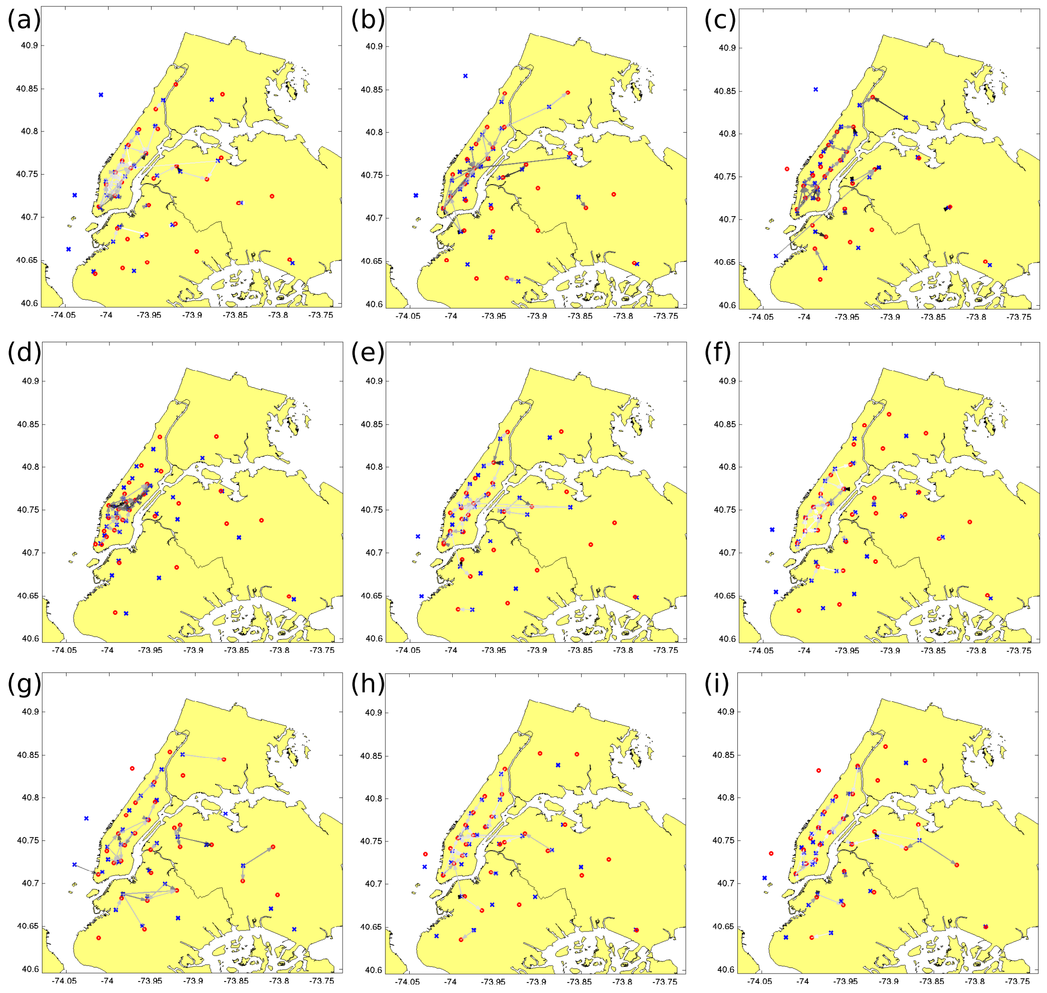

Figure 8: Combined inference results for the HSMM applied to one month of taxi data, showing taxi traffic patterns at different times of the day and week. Each plot shows the inferred pickup hubs (blue crosses), dropoff hubs (red circles), and highest-weighted transitions between pickup and dropoff hubs (directed edges denote the largest 40 elements of T, where a darker edge corresponds with a stronger weight). See text for details on individual plots.

of specific highly weighted transitions). Plots (b) and (c) show the traffic of pattern during morning rush hour (06:00-09:00) and evening rush hour (16:00-19:00), respectively. In (b), we see a strong concentration of traffic going to midtown Manhattan (particularly, midtown east) and the financial district (South Manhattan). In these morning hours, there is relatively little taxi traffic in Brooklyn and Queens (though we do see a major flow from a hub associated with LaGuardia airport, in North Queens, to midtown Manhattan). In (c), during the evening hours, we see major traffic to and between Greenwich Village and East Village in Manhattan, and to the West and East sides of cental park; we also see increased activity in both Queens and Brooklyn. Plots (e), (f), (h), and (i) aim to elicit the differences in traffic patterns between weekdays and weekends. In particular, plots (e) and (f) show combined model results during weekdays (i.e. Mondays through Fridays), where (e) shows results for the 09:00 hour only, and (f) shows results taken over all hours in the day. Likewise, plots (h) and (i) show combined model results during weekends (i.e. Saturday and Sunday), where (h) shows results taken for the 09:00 hour only, and (i) shows results taken over all hours in the day. We see in weekdays, there is a much higher concentration of taxi traffic in Manhattan (and looking at the 09:00 hour, we see a particular concentration in Midtown Manhattan and East Queens near the

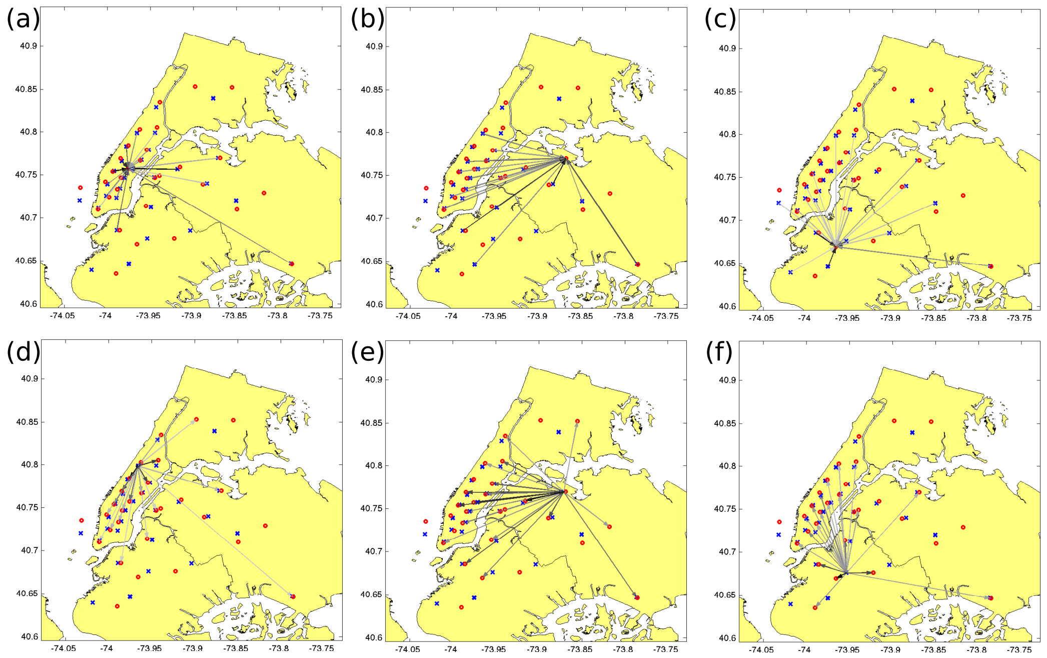

Figure 9: Combined inference results for the HSMM applied to one month of taxi data, showing taxi traffic patterns either to or from a selected individual hub. Each plot shows the inferred pickup hubs (blue crosses), dropoff hubs (red circles), and highest-weighted transitions for a single pickup or dropoff hub (directed edges denote the largest 20 elements of the row or column of T associated with this pickup or dropoff hub, where a darker edge corresponds with a stronger weight). The top row of plots shows the distribution over pickup hubs for three dropoff hubs (one in Manhattan, one in Queens, and one in Brooklyn), while the bottom row of plots shows the distribution over dropoff hubs for three pickup hubs (in three similar locations). See text for more details on individual plots.

border of Manhattan), while in weekends, there is a much higher concentration around the border of central park, throughout Queens, and in Brooklyn. Finally, in plot (g), we show taxi traffic results combined only over late-night hours (00:00-04:00). In this plot, we see far more traffic in small clusters throughout Brooklyn, Queens, and the Bronx, and a cluster near Greenwich Village and East Village in Manhattan.

We can also use the inferred model results to show the transportation patterns for a selected individual hub. For example, we might select the hub closest to the LaGuardia airport in North Queens, and want to see the distribution over hubs that people take the taxi to (from LaGuardia) or the distribution over hubs that people take the taxi from (to LaGuardia). We can get these types of results for any of our inferred hubs. In Figure 9, we show such results for a few selected hubs. Plot (a) shows the distribution over pickup hubs for a single dropoff hub in midtown Manhattan. We see that most taxi trips are from neighboring hubs in Manhattan, and from LaGuardia airport in North Queens and John F. Kennedy airport in Southeast Queens. Similarly, in Plot (b), we show the distribution over dropoff hubs for a single pickup hub in Manhattan, and see a similar transportation pattern. In plots (b) and (e) we show the same type of results for a dropoff hub and pickup hub in North Queens, and in plots (c) and (f) we show the same type of results for a dropoff hub and pickup hub in central Brooklyn. Of note is that plots (b) and (e) correspond to the dropoff and pickup hubs closest to LaGuardia airport in queens; we see that the highest weighted pickup hubs traveling to to this airport are in Queens and Brooklyn (including the hub located at John F. Kennedy Airport), while the highest weighted dropoff hubs traveling from this airport are in West Queens and Manhattan.

8. Conclusion

In this paper, we present an embarrassingly parallel MCMC algorithm and provide theoretical guarantees about the samples it yields. Experimental results demonstrate our method's potential to speed up burn-in and perform faster asymptotically correct sampling. Further, it can be used in settings where data are partitioned onto multiple machines that have little intercommunication—this is ideal for use in a MapReduce setting. Currently, our algorithm works primarily when the posterior samples are real, unconstrained values and we plan to extend our algorithm to more general settings in future work.

References

[1] Alekh Agarwal and John C Duchi, *Distributed delayed stochastic optimization*, Decision and Control (CDC), 2012 IEEE 51st Annual Conference on, IEEE, 2012, pp. 5451–5452.

[2] Sungjin Ahn, Anoop Korattikara, and Max Welling, *Bayesian posterior sampling via stochastic gradient fisher scoring*, Proceedings of the 29th International Conference on Machine Learning, 2012, pp. 1591–1598.

[3] David M Blei, Andrew Y Ng, and Michael I Jordan, *Latent dirichlet allocation*, The Journal of Machine Learning Research **3** (2003), 993–1022.

[4] Trevor Campbell and Jonathan How, *Approximate decentralized bayesian inference*, 2014.

[5] Jeffrey Dean and Sanjay Ghemawat, *Mapreduce: simplified data processing on large clusters*, Communications of the ACM **51** (2008), no. 1, 107–113.

[6] Samuel J Gershman and David M Blei, *A tutorial on bayesian nonparametric models*, Journal of Mathematical Psychology **56** (2012), no. 1, 1–12.

[7] Nils Lid Hjort and Ingrid K Glad, *Nonparametric density estimation with a parametric start*, The Annals of Statistics (1995), 882–904.

[8] Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B. Gibbons, Gregory R. Ganger, Garth Gibson, and Eric P. Xing, *More effective distributed ml via a stale synchronous parallel parameter server*, Advances in Neural Information Processing Systems, 2013.

[9] Matthew D Hoffman and Andrew Gelman, *The no-u-turn sampler: Adaptively setting path lengths in hamiltonian monte carlo*, arXiv preprint arXiv:1111.4246 (2011).

[10] Anoop Korattikara, Yutian Chen, and Max Welling, *Austerity in MCMC land: Cutting the Metropolis-Hastings budget*, arXiv preprint arXiv:1304.5299 (2013).

[11] John Langford, Alex J Smola, and Martin Zinkevich, *Slow learners are fast*, Advances in Neural Information Processing Systems, 2009.

[12] Kathryn Blackmond Laskey and James W Myers, *Population Markov chain Monte Carlo*, Machine Learning **50** (2003), no. 1-2, 175–196.

[13] Lucien Le Cam, *Asymptotic methods in statistical decision theory*, New York (1986).

[14] Yucheng Low, Joseph E Gonzalez, Aapo Kyrola, Danny Bickson, Carlos E Guestrin, and Joseph Hellerstein, *Graphlab: A new framework for parallel machine learning*, arXiv preprint arXiv:1408.2041 (2014).

[15] Lawrence Murray, *Distributed Markov chain Monte Carlo*, Proceedings of Neural Information Processing Systems Workshop on Learning on Cores, Clusters and Clouds, vol. 11, 2010.

[16] Willie Neiswanger, Chong Wang, and Eric Xing, *Asymptotically exact, embarrassingly parallel mcmc*, Proceedings of the 30th Conference on Uncertainty in Artificial Intelligence (UAI 2014), 2014.

[17] David Newman, Arthur Asuncion, Padhraic Smyth, and Max Welling, *Distributed algorithms for topic models*, The Journal of Machine Learning Research **10** (2009), 1801–1828.

[18] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith, *Smooth sensitivity and sampling in private data analysis*, Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, ACM, 2007, pp. 75–84.

[19] Junier Oliva, Barnabás Póczos, and Jeff Schneider, *Distribution to distribution regression*, Proceedings of The 30th International Conference on Machine Learning, 2013, pp. 1049–1057.

[20] Sam Patterson and Yee Whye Teh, *Stochastic gradient riemannian langevin dynamics on the probability simplex*, Advances in Neural Information Processing Systems, 2013.

[21] Steven L. Scott, Alexander W. Blocker, and Fernando V. Bonassi, *Bayes and big data: The consensus monte carlo algorithm*, Bayes 250, 2013.

[22] Alexander Smola and Shravan Narayanamurthy, *An architecture for parallel topic models*, Proceedings of the VLDB Endowment **3** (2010), no. 1-2, 703–710.

[23] Max Welling and Yee W Teh, *Bayesian learning via stochastic gradient Langevin dynamics*, Proceedings of the 28th International Conference on Machine Learning, 2011, pp. 681–688.

[24] Darren J Wilkinson, *Parallel Bayesian computation*, Statistics Textbooks and Monographs **184** (2006), 477.

[25] Sinead Williamson, Avinava Dubey, and Eric P Xing, *Parallel Markov chain Monte Carlo for nonparametric mixture models*, Proceedings of the 30th International Conference on Machine Learning, 2013, pp. 98–106.