Abstract

This paper presents a novel algorithm, based upon the dependent Dirichlet process mixture model (DDPMM), for clustering batch-sequential data containing an unknown number of evolving clusters. The algorithm is derived via a low-variance asymptotic analysis of the Gibbs sampling algorithm for the DDPMM, and provides a hard clustering with convergence guarantees similar to those of the k-means algorithm. Empirical results from a synthetic test with moving Gaussian clusters and a test with real ADS-B aircraft trajectory data demonstrate that the algorithm requires orders of magnitude less computational time than contemporary probabilistic and hard clustering algorithms, while providing higher accuracy on the examined datasets.

1 Introduction

The Dirichlet process mixture model (DPMM) is a powerful tool for clustering data that enables the inference of an unbounded number of mixture components, and has been widely studied in the machine learning and statistics communities [1–4]. Despite its flexibility, it assumes the observations are exchangeable, and therefore that the data points have no inherent ordering that influences their labeling. This assumption is invalid for modeling temporally/spatially evolving phenomena, in which the order of the data points plays a principal role in creating meaningful clusters. The dependent Dirichlet process (DPD), originally formulated by MacEacern [5], provides a prior over such evolving mixture models, and is a promising tool for incrementally monitoring the dynamic evolution of the cluster structure within a dataset. More recently, a construction of the DPD built upon completely random measures [6] led to the development of the dependent Dirichlet process Mixture model (DDPMM) and a corresponding approximate posterior inference Gibbs sampling algorithm. This model generalizes the DPMM by including birth, death and transition processes for the clusters in the model.

The DDPMM is a Bayesian nonparametric (BNP) model, part of an ever-growing class of probabilistic models for which inference captures uncertainty in the both the number of parameters and their values. While these models are powerful in their capability to capture complex structures in data without requiring explicit model selection, they suffer some practical shortcomings. Inference techniques for BNPs typically fall into two classes: sampling methods (e.g. Gibbs sampling [2] or particle learning [4]) and optimization methods (e.g. variational inference [3] or stochastic variational inference [7]). Current methods based on sampling do not scale well with the size of the
dataset [8]. Most optimization methods require analytic derivatives and the selection of an upper bound on the number of clusters a priori, where the computational complexity increases with that upper bound [3][7]. State-of-the-art techniques in both classes are not ideal for use in contexts where performing inference quickly and reliably on large volumes of streaming data is crucial for timely decision-making, such as autonomous robotic systems [2][11]. On the other hand, many classical clustering methods [12][14] scale well with the size of the dataset and are easy to implement, and advances have recently been made to capture the flexibility of Bayesian nonparametrics in such approaches [15]. However, as of yet there is no classical algorithm that captures dynamic cluster structure with the same representational power as the DDP mixture model.

This paper discusses the Dynamic Means algorithm, a novel hard clustering algorithm for spatio-temporal data derived from the low-variance asymptotic limit of the Gibbs sampling algorithm for the dependent Dirichlet process Gaussian mixture model. This algorithm captures the scalability and ease of implementation of classical clustering methods, along with the representational power of the DDP prior, and is guaranteed to converge to a local minimum of a k-means-like cost function. The algorithm is significantly more computationally tractable than Gibbs sampling, particle learning, and variational inference for the DDP mixture model in practice, while providing equivalent or better clustering accuracy on the examples presented. The performance and characteristics of the algorithm are demonstrated in a test on synthetic data, with a comparison to those of Gibbs sampling, particle learning and variational inference. Finally, the applicability of the algorithm to real data is presented through an example of clustering a spatio-temporal dataset of aircraft trajectories recorded across the United States.

2 Background

The Dirichlet process (DP) is a prior over mixture models, where the number of mixture components is not known a priori[16]. In general, we denote \(D \sim \text{DP}(\mu)\), where \(\alpha_\mu \in \mathbb{R}_+\) and \(\mu : \Omega \to \mathbb{R}_+\), \(\int_\Omega \mu = \alpha_\mu\) are the concentration parameter and base measure of the DP, respectively. If \(D \sim \text{DP}\), then \(D = \{(\theta_k, \pi_k)\}_{k=0}^\infty \subset \Omega \times \mathbb{R}_+\), where \(\theta_k \in \Omega\) and \(\pi_k \in \mathbb{R}_+\) [17]. The reader is directed to [1] for a more thorough coverage of Dirichlet processes.

The dependent Dirichlet process (DDP)[5], an extension to the DP, is a prior over evolving mixture models. Given a Poisson process construction[6], the DDP essentially forms a Markov chain of DPs (\(D_1, D_2, \ldots\)), where the transitions are governed by a set of three stochastic operations: Points \(\theta_k\) may be added, removed, and may move during each step of the Markov chain. Thus, they become parameterized by time, denoted by \(\theta_{kt}\). In slightly more detail, if \(D_t\) is the DP at time step \(t\), then the following procedure defines the generative model of \(D_t\) conditioned on \(D_{t-1} \sim \text{DP}(\mu_{t-1})\):

1. **Subsampling**: Define a function \(q : \Omega \to [0, 1]\). Then for each point \((\theta, \pi) \in D_{t-1}\), sample a Bernoulli distribution \(b_\theta \sim \text{Be}(q(\theta))\). Set \(D'_t\) to be the collection of points \((\theta, \pi)\) such that \(b_\theta = 1\), and renormalize the weights. Then \(D'_t \sim \text{DP}(q\mu_{t-1})\), where \((q\mu)(A) = \int_A q(\theta)\mu(d\theta)\).

2. **Transition**: Define a distribution \(T : \Omega \times \Omega \to \mathbb{R}_+\). For each point \((\theta, \pi) \in D'_t\), sample \(\theta' \sim T(\theta|\theta)\), and set \(D''_t\) to be the collection of points \((\theta', \pi)\). Then \(D''_t \sim \text{DP}(qT\mu_{t-1})\), where \((T\mu)(A) = \int_A \int_\Omega T(\theta'|\theta)\mu(d\theta)d\theta)\).

3. **Superposition**: Sample \(F \sim \text{DP}(\nu)\), and sample \((c_D, c_F) \sim \text{Dir}(Tq\mu_{t-1}(\Omega), \nu(\Omega))\). Then set \(D_t\) to be the union of \((\theta, c_D \pi)\) for all \((\theta, \pi) \in D''_t\) and \((\theta, c_F \pi)\) for all \((\theta, \pi) \in F\). Thus, \(D_t\) is a random convex combination of \(D''_t\) and \(F\), where \(D_t \sim \text{DP}(Tq\mu_{t-1} + \nu)\).

If the DDP is used as a prior over a mixture model, these three operations allow new mixture components to arise over time, and old mixture components to exhibit dynamics and perhaps disappear over time. As this is covered thoroughly in [6], the mathematics of the underlying Poisson point process construction are not discussed in more depth in this work. However, an important result of using such a construction is the development of an explicit posterior for \(D_t\) given observations of the points \(\theta_{kt}\) at timestep \(t\). For each point \(k\) that was observed in \(D_{\tau}\) for some \(1 \leq \tau \leq t\), define:

- \(n_{kt} \in \mathbb{N}\) as the number of observations of point \(k\) in timestep \(t\);
- \(c_{kt} \in \mathbb{N}\) as the number of past observations of point \(k\) prior to timestep \(t\), i.e. \(c_{kt} = \sum_{\tau=1}^{t-1} n_{kt} \); and
- \(q_{kt} \in (0, 1)\) as the subsampling weight on point \(k\) at timestep \(t\); and \(\Delta t_k\) as the number of time steps that have elapsed since point \(k\).
label assignment step yields meaningful assignments as long as

\[ \frac{c_{kt}}{\rho} \approx \nu \begin{cases} 1, & \text{if } \theta_{kt} \text{ is first observed during timestep } t. \\
0, & \text{otherwise}
\end{cases} \]

This posterior leads directly to the development of a Gibbs sampling algorithm for the DDP, whose low-variance asymptotics are discussed further below.

3 Asymptotic Analysis of the DDP Mixture

The dependent Dirichlet process Gaussian mixture model (DDP-GMM) serves as the foundation upon which the present work is built. The generative model of a DDP-GMM at time step \( t \) is

\[
\begin{align*}
\{\theta_{kt}, \pi_{kt}\}_{k=1}^{\infty} & \sim \text{DP}(\mu_t) \\
\{z_{it}\}_{i=1}^{N_t} & \sim \text{Categorical}\{\pi_{kt}\}_{k=1}^{\infty} \\
y_{it} & \sim \mathcal{N}(\theta_{zt}, \sigma I)
\end{align*}
\]

where \( \theta_{kt} \) is the measure for unobserved points at time step \( t \). Throughout the rest of this paper, the subscript \( kt \) refers to quantities related to cluster \( k \) at time step \( t \), and subscript \( it \) refers to quantities related to observation \( i \) at time step \( t \).

The Gibbs sampling algorithm for the DDP-GMM iterates between sampling labels \( z_{it} \) for datapoints \( y_{it} \) given the set of parameters \( \{\theta_{kt}\} \), and sampling parameters \( \theta_{kt} \) given each group of data \( \{y_{it} : z_{it} = k\} \). Assuming the transition model \( T \) is Gaussian, and the subsampling function \( q \) is constant, the functions and distributions used in the Gibbs sampling algorithm are: the prior over cluster parameters, \( \theta \sim \mathcal{N}(\phi, \rho I) \); the likelihood of an observation given its cluster parameter, \( y_{it} \sim \mathcal{N}(\theta_{kt}, \sigma I) \); the distribution over the transitioned cluster parameter given its last known location after \( \Delta t_k \) time steps, \( \theta_{kt} \sim \mathcal{N}(\theta_{kt(-\Delta t_k)}, \xi \Delta t_k I) \); and the subsampling function \( q(\theta) = q \in (0, 1) \). Given these functions and distributions, the low-variance asymptotic limits (i.e. \( \sigma \to 0 \)) of these two steps are discussed in the following sections.

3.1 Setting Labels Given Parameters

In the label sampling step, a datapoint \( y_{it} \) can either create a new cluster, join a current cluster, or revive an old, transitioned cluster. Using the distributions defined previously, the label assignment probabilities are

\[
p(z_{it} = k | \ldots) \propto \begin{cases} \\
\alpha_L (2\pi(\sigma + \rho))^{-d/2} \exp \left( \frac{-||y_{it} - \phi||^2}{2(\sigma + \rho)} \right), & k = K + 1 \\
(c_{kt} + n_{kt})(2\pi\sigma)^{-d/2} \exp \left( \frac{-||y_{it} - \theta_{kt}\|}{2\sigma} \right), & n_{kt} > 0 \\
q_{kt} c_{kt}(2\pi(\sigma + \xi \Delta t_k))^{-d/2} \exp \left( \frac{-||y_{it} - \theta_{kt(-\Delta t_k)}\|^2}{2(\sigma + \xi \Delta t_k)} \right), & n_{kt} = 0
\end{cases}
\]

where \( q_{kt} = \rho^{\Delta t_k} \) due to the fact that \( q(\theta) \) is constant over \( \Omega \), and \( \alpha_L = \alpha_\nu \frac{1-\xi}{1-\rho} \) where \( \alpha_\nu \) is the concentration parameter for the innovation process, \( F_t \). The low-variance asymptotic limit of this label assignment step yields meaningful assignments as long as \( \alpha_L, \alpha_\nu, \xi, \text{ and } q \) vary appropriately with \( \sigma \); thus, setting \( \alpha_L, \xi, \text{ and } q \) as follows (where \( \lambda, \tau \text{ and } Q \) are positive constants):

\[
\alpha_L = (1 + \rho/\tau)^{d/2} \exp \left( -\frac{\Lambda}{2\tau} \right), \quad \xi = \tau \sigma, \quad q = \exp \left( -\frac{Q}{2\tau} \right)
\]

yields the following assignments in the limit as \( \sigma \to 0 \):

\[
z_{it} = \arg \min_k \{J_k\}, \quad J_k = \begin{cases} ||y_{it} - \theta_{kt}\|^2 & \text{if } \theta_{kt} \text{ instantiated} \\
Q \Delta t_k + ||y_{it} - \theta_{kt(-\Delta t_k)}\|^2 - \xi \Delta t_k + 1 & \text{if } \theta_{kt} \text{ old, uninstantiated} \\
\lambda & \text{if } \theta_{kt} \text{ new}
\end{cases}
\]

In this assignment step, \( Q \Delta t_k \) acts as a cost penalty for reviving old clusters that increases with the time since the cluster was last seen, \( \tau \Delta t_k \) acts as a cost reduction to account for the possible motion of clusters since they were last instantiated, \( \lambda \) acts as a cost penalty for introducing a new cluster.
3.2 Setting Parameters given Labels

In the parameter sampling step, the parameters are sampled using the distribution

\[ p(\theta_{kt} | \{y_{it} : z_{it} = k\}) \propto p(\{y_{it} : z_{it} = k\} | \theta_{kt}) p(\theta_{kt}) \]  

(6)

There are two cases to consider when setting a parameter \( \theta_{kt} \). Either \( \Delta t_k = 0 \) and the cluster is new in the current time step, or \( \Delta t_k > 0 \) and the cluster was previously created, disappeared for some amount of time, and then was revived in the current time step.

**New Cluster** Suppose cluster \( k \) is being newly created. In this case, \( \theta_{kt} \sim \mathcal{N}(\phi, \rho) \). Using the fact that a normal prior is conjugate a normal likelihood, we have a closed-form posterior for \( \theta_{kt} \):

\[ \theta_{kt} | \{y_{it} : z_{it} = k\} \sim \mathcal{N}(\theta_{\text{post}}, \sigma_{\text{post}}) \]

\[ \theta_{\text{post}} = \sigma_{\text{post}} \left( \frac{\phi}{\rho} + \frac{\sum_{i=1}^{n_k} y_{it}}{\sigma} \right), \sigma_{\text{post}} = \left( \frac{1}{\rho} + \frac{n_k}{\sigma} \right)^{-1} \]  

(7)

Then letting \( \sigma \to 0 \), we get

\[ \theta_{kt} = \left( \sum_{i=1}^{n_k} y_{it} \right) \frac{1}{n_k} \equiv m_{kt} \]  

(8)

where \( m_{kt} \) is the mean of the observations in the current timestep.

**Revived Cluster** Suppose there are \( \Delta t_k \) time steps where cluster \( k \) was not observed, but there are now \( n_{kt} \) data points with mean \( m_{kt} \) assigned to it in this time step. In this case,

\[ p(\theta_{kt}) = \int_\theta T(\theta_{kt} | \theta) p(\theta) d\theta, \theta \sim \mathcal{N}(\theta', \sigma'). \]  

(9)

Again using conjugacy of normal likelihoods and priors,

\[ \theta_{kt} | \{y_{it} : z_{it} = k\} \sim \mathcal{N}(\theta_{\text{post}}, \sigma_{\text{post}}) \]

\[ \theta_{\text{post}} = \sigma_{\text{post}} \left( \frac{\theta'}{\xi \Delta t_k + \sigma'} + \frac{\sum_{i=1}^{n_k} y_{it}}{\sigma} \right), \sigma_{\text{post}} = \left( \frac{1}{\xi \Delta t_k + \sigma'} + \frac{n_k}{\sigma} \right)^{-1} \]  

(10)

Similarly to the label assignment step, let \( \xi = \tau \sigma \). Then as long as \( \sigma' = \sigma/w, w > 0 \) (which holds if equation (10) is used to recursively keep track of the parameter posterior), taking the asymptotic limit of this as \( \sigma \to 0 \) yields:

\[ \theta_{kt} = \frac{\theta'(w^{-1} + \Delta t_k \tau)^{-1} + n_{kt} m_{kt}}{(w^{-1} + \Delta t_k \tau)^{-1} + n_k} \]  

(11)

that is to say, the revived \( \theta_{kt} \) is a weighted average of estimates using current timestep data and previous timestep data. \( \tau \) controls how much the current data is favoured - as \( \tau \) increases, the weight on current data increases, which is explained by the fact that our uncertainty in where the old cluster transitioned to increases with \( \tau \). It is also noted that if \( \tau = 0 \), this reduces to a simple weighted average using the amount of data collected as weights.

**Combined Update** Combining the updates for new cluster parameters and old transitioned cluster parameters yields a recursive update scheme:

\[ \begin{align*}
\theta_{k0} &= m_{k0} \\
\theta_{kt} &= \theta_{k0(t-\Delta t_k)} + \theta_{k0} \gamma_{kt} + n_{kt} m_{kt} \\
w_{kt} &= \gamma_{kt} + n_{kt}
\end{align*} \]  

(12)

where time step 0 here corresponds to when the cluster is first created. An interesting interpretation of this update is that it behaves like a standard Kalman filter, in which \( w_{kt}^{-1} \) serves as the current estimate variance, \( \tau \) serves as the process noise variance, and \( n_{kt} \) serves as the inverse of the measurement variance.
Theorem 1. Each iteration in Algorithm 2 monotonically decreases the cost function $L_t$, where

$$L_t = \sum_{k \in K_t} \left( \frac{\text{New Cost}}{\lambda [\Delta t_k = 0]} + \frac{\text{Revival Cost}}{\Delta t_k} + \frac{\text{Weighted-Prior Sum-Squares Cost}}{\varphi} \right) \sum_{y_{it} \in Y_t} \kappa_{it}^2 \right)$$

The cost function is comprised of a number of components for each currently active cluster $k \in K_t$: A penalty for new clusters based on $\lambda$, a penalty for old clusters based on $Q$ and $\Delta t_k$, and finally a prior-weighted sum of squared distance cost for all the observations in cluster $k$. It is noted that
for new clusters, $\theta_{kt} = \theta_{k(t-\Delta t_k)}$ since $\Delta t_k = 0$, so the least squares cost is unweighted. The ASSIGNPARAMS function calculates this cost function in each iteration of Algorithm 2 and the algorithm terminates once the cost function does not decrease during an iteration.

4.2 Reparameterizing the Algorithm

In order to use the dynamic means algorithm, there are three free parameters to select: $\lambda$, $Q$, and $\tau$. While $\lambda$ represents how far an observation can be from a cluster before it is placed in a new cluster, and thus can be tuned intuitively, $Q$ and $\tau$ are not so straightforward. The parameter $Q$ represents a conceptual added distance from any data point to a cluster for every time step that the cluster is not observed. The parameter $\tau$ represents a conceptual reduction of distance from any data point to a cluster for every time step that the cluster is not observed. How these two quantities affect the algorithm, and how they interact with the setting of $\lambda$, is hard to judge.

Instead of picking $Q$ and $\tau$ directly, the algorithm may be reparameterized by picking $N_Q, k_\tau \in \mathbb{R}^+$, $N_Q > 1$, $k_\tau \geq 1$, and given a choice of $\lambda$, setting

$$Q = \lambda/N_Q, \quad \tau = \frac{N_Q(k_{\tau} - 1) + 1}{N_Q - 1}.$$  \hfill (16)

If $Q$ and $\tau$ are set in this manner, $N_Q$ represents the number (possibly fractional) of time steps a cluster can be unobserved before the label update (5) will never revive that cluster, and $k_\tau \lambda$ represents the maximum squared distance away from a cluster center such that after a single time step, the label update (5) will revive that cluster. As $N_Q$ and $k_{\tau}$ are specified in terms of concrete algorithmic behavior, they are intuitively easier to set than $Q$ and $\tau$.

5 Related Work

Prior k-means clustering algorithms that determine the number of clusters present in the data have primarily involved a method for iteratively modifying $k$ using various statistical criteria [13, 14, 18]. In contrast, this work derives this capability from a Bayesian nonparametric model, similarly to the DP-Means algorithm [15]. In this sense, the relationship between the Dynamic Means algorithm and the dependent Dirichlet process [6] is exactly that between the DP-Means algorithm and Dirichlet process [16], where the Dynamic Means algorithm may be seen as an extension to the DP-Means that handles sequential data with time-varying cluster parameters. Other clustering methods such as MONIC [19] and MC3[20] have the capability to monitor time-varying clusters; however, these methods require datapoints to be identifiable across timesteps, and determine cluster similarity across timesteps via the commonalities between label assignments. The Dynamic Means algorithm does not require such information, and tracks clusters essentially based on similarity of the parameters across timesteps.

There are a number of sequential Monte-Carlo methods that have capabilities similar to the Dynamic Means algorithm. In particular, particle learning [21] is a general sequential Bayesian inference technique based on auxiliary particle filtering [22] that can be adapted for use in the present clustering problem. Similar methods in the target tracking community for multitarget tracking with noisy observations exist based on particle filtering [23, 24], of which particle learning may be seen as a generalization.

6 Applications

6.1 Synthetic Gaussian Motion Data

In this experiment, moving Gaussian clusters on $[0, 1] \times [0, 1]$ were generated synthetically over a period of 100 time steps. In each step, there was some number of clusters, each having 15 data points. The data points were sampled from a symmetric Gaussian distribution with a standard deviation of 0.05. Between time steps, the cluster centers moved randomly, with displacements sampled from the same distribution. At each time step, each cluster had a 0.05 probability of being destroyed.

This data was clustered with Dynamic Means (with 3 random assignment ordering restarts), DDP-GMM Gibbs sampling, variational inference, and particle learning on a computer with an Intel i7
processor and 16GB of memory. First, the number of clusters was fixed to 5, and the parameter space of each algorithm was searched for the best possible performance in terms of cluster label accuracy (taking into account label consistency across time steps). For the Dynamic Means algorithm the space \((\lambda, T_Q, k_\tau) \in [0, 0.16] \times [0, 10] \times [1, 6]\) was tested, while for the comparison algorithms the space \((\log_{10}(\alpha), q) \in [-4, 2] \times [0, 1]\) was tested, with 50 trials at each parameter setting. The results of this parameter sweep for the Dynamic Means algorithm are shown in Figures 1a - 1c. Figures 1d and 1e show how the average clustering accuracy varies with the parameters after fixing either \(k_\tau\) or \(T_Q\) to their values at the maximum accuracy parameter setting over the full space. The Dynamic Means algorithm had a similar robustness with respect to variations in its parameters as the comparison algorithms, when considering parameter deviations in terms of percentages of the maximum accuracy parameter setting. The histogram in Figure 1f demonstrates that the clustering speed is robust to the setting of parameters.

Using the best parameter setting with respect to mean cluster label accuracy for each algorithm, the data as described above were clustered in 50 trials with a varying number of clusters present in the data. For the Dynamic Means algorithm, parameter values \(\lambda = 0.04, T_Q = 6.8,\) and \(k_\tau = 1.01\) were used, and the algorithm was again given 3 attempts with random labelling assignment orders, where the lowest cost solution of the 3 was picked to proceed to the next time step. For the other algorithms, the parameter values \(\alpha = 1\) and \(q = 0.05\) were used, with a Gaussian transition distribution variance of 0.05. The number of samples for the Gibbs sampling algorithm was 5000 with one recorded for every 5 samples, the number of particles for the particle learning algorithm was 100, and the variational inference algorithm was run to a tolerance of \(10^{-20}\) with the maximum number of iterations set to 5000.

In Figures 1d and 1e, the labelling accuracy and clustering time (respectively) for the algorithms is shown. Note that the sampling algorithms (Gibbs and PL) were handicapped to generate Figure 1d; the best posterior sample in terms of labelling accuracy was selected at each time step, which required knowledge of the true labelling. This example demonstrates that the Dynamic Means algorithm outperforms other standard inference algorithms in terms of label accuracy, while having the benefit of fast convergence to a solution. Note that the label accuracy computation included enforcing consistency across timesteps, to allow tracking individual cluster trajectories. If this is not enforced (i.e. accuracy considers each time step independently), the other algorithms provide accuracies more comparable to those of the Dynamic Means algorithm.
6.2 Aircraft Trajectory Clustering

In persistent surveillance missions, the ability to autonomously recognize anomalous situations and behaviors allows the intelligent allocation of resources to best address those anomalies. In the case of monitoring mobile agent trajectories, characterizing typical spatial and temporal patterns in their motions is an important step towards the detection of such anomalies. In this experiment, the Dynamic Means algorithm was used to find the typical spatial and temporal patterns in the motions of commercial aircraft, a domain in which anomaly detection is safety-critical.

Automatic dependent surveillance-broadcast (ADS-B) data, including plane identification, timestamp, latitude, longitude, heading and speed, was collected from all transmitting planes across the United States during the week from 2013-3-22 1:30:0 to 2013-3-28 12:0:0 UTC. The following procedure was used to create feature vectors for clustering the data. First, individual ADS-B messages were connected together based on their plane identification and timestamp to form trajectories, and erroneous trajectories were filtered based on reasonable spatial/temporal bounds, yielding 17,895 unique trajectories. Then, for each trajectory, a Gaussian process was trained using the latitude and longitude of each ADS-B point along the trajectory as the inputs and the North and East components of plane velocity at those points as the outputs. Next, the mean velocity from the Gaussian process was queried for each point on a regular lattice across the USA with 10 latitudes and 20 longitudes, and used to create a 400-dimensional feature vector for each trajectory. Finally, the resulting 17,895 feature vectors were clustered using the Dynamic Means algorithm, DP-Means algorithm (run on the entire dataset in a single batch), and the DDP Gibbs sampling algorithm (50 sampling iterations total per time step) for all plane take-offs occurring in a sequence of half hour time windows.

The clustering results of this exercise from the Dynamic Means algorithm are shown in Figure 2. Over 10 trials, the Dynamic Means algorithm took an average of 4.9s to process all the feature vectors, the DP-Means algorithm took an average of 72.3s, while the DDP Gibbs sampling algorithm took an average of 5426.1s. The DP-Means algorithm provided a model with about twice as many clusters (mostly near duplicates of other clusters) due to the inability to capture slight variations through cluster transition processes and the inability to remove erroneous clusters by modeling cluster death processes. The DDP Gibbs sampling algorithm did not give reliable results and proved to be very difficult to tune on this dataset, due to the high dimensionality of the feature vectors and the high runtime of the algorithm on the available hardware. The Dynamic Means algorithm, on the other hand, successfully identified the major flows of commercial aircraft across the United States along with their temporal patterns (the 12 most popular of which are shown in Figure 2) in orders of magnitude less time than both the DP-means and DDP mixture Gibbs sampling algorithms.

7 Conclusion

This work developed a clustering algorithm for batch-sequential data containing temporally evolving clusters, derived from a low-variance asymptotic analysis of the Gibbs sampling algorithm for the dependent Dirichlet process mixture model. Synthetic and real data experiments demonstrated that the algorithm requires orders of magnitude less computational time than contemporary probabilistic and hard clustering algorithms, while providing higher accuracy on the examined datasets. The speed of inference coupled with the convergence guarantees provided yield an algorithm which is suitable for use in time-critical applications, such as online model-based autonomous planning systems.
References

[1] Yee Whye Teh. Dirichlet processes. In *Encyclopedia of Machine Learning*, Springer, New York, 2010.

[2] Radford M. Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

[3] David M. Blei and Michael I. Jordan. Variational inference for dirichlet process mixtures. *Bayesian Analysis*, 1(1):121–144, 2006.

[4] Carlos M. Carvalho, Hedibert F. Lopes, Nicholas G. Polson, and Matt A. Taddy. Particle learning for general mixtures. *Bayesian Analysis*, 5(4):709–740, 2010.

[5] Steven N. MacEachern. Dependent nonparametric processes. In *Proceedings of the Bayesian Statistical Science Section*. American Statistical Association, 1999.

[6] Dahua Lin, Eric Grimson, and John Fisher. Construction of dependent dirichlet processes based on poisson processes. In *Neural Information Processing Systems*, 2010.

[7] Matt Hoffman, David Blei, Chong Wang, and John Paisley. Stochastic variational inference. *arXiv ePrint* 1206.7051, 2012.

[8] Finale Doshi-Velez and Zoubin Ghahramani. Accelerated sampling for the indian buffet process. In *Proceedings of the International Conference on Machine Learning*, 2009.

[9] Felix Endres, Christian Plagemann, Cyrill Stachniss, and Wolfram Burgard. Unsupervised discovery of object classes from range data using latent dirichlet allocation. In *Robotics Science and Systems*, 2005.

[10] Matthias Luber, Kai Arras, Christian Plagemann, and Wolfram Burgard. Classifying dynamic objects: An unsupervised learning approach. In *Robotics Science and Systems*, 2004.

[11] Zhikun Wang, Marc Deisenroth, Heni Ben Amor, David Vogt, Bernard Schölkopf, and Jan Peters. Probabilistic modeling of human movements for intention inference. In *Robotics Science and Systems*, 2008.

[12] Stuart P. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982.

[13] Dan Pelleg and Andrew Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the 17th International Conference on Machine Learning*, 2000.

[14] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society B*, 63(2):411–423, 2001.

[15] Brian Kulis and Michael I. Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, Edinburgh, Scotland, 2012.

[16] Thomas S. Ferguson. A bayesian analysis of some nonparametric problems. *The Annals of Statistics*, 1(2):209–230, 1973.

[17] Jayaram Sethuraman. A constructive definition of dirichlet priors. *Statistica Sinica*, 4:639–650, 1994.

[18] Tsunenori Ishioka. Extended k-means with an efficient estimation of the number of clusters. In *Proceedings of the 2nd International Conference on Intelligent Data Engineering and Automated Learning*, pages 17–22, 2000.

[19] Myra Spiliopoulou, Irene Ntoutsi, Yannis Theodoridis, and Rene Schult. Monic - modeling and monitoring cluster transitions. In *Proceedings of the 12th International Conference on Knowledge Discovering and Data Mining*, pages 706–711, 2006.

[20] Panos Kalnis, Nikos Mamoulis, and Spiridon Bakiras. On discovering moving clusters in spatio-temporal data. In *Proceedings of the 9th International Symposium on Spatial and Temporal Databases*, pages 364–381. Springer, 2005.

[21] Carlos M. Carvalho, Michael S. Johannes, Hedibert F. Lopes, and Nicholas G. Polson. Particle learning and smoothing. *Statistical Science*, 25(1):88–106, 2010.

[22] Michael Pitt and Neil Shephard. Filtering via simulation: Auxiliary particle filters. *Journal of the American Statistical Association*, 94(446):590–599, 1999.

[23] Carine Hue, Jean-Pierre Le Cadre, and Patrick Pérez. Tracking multiple objects with particle filtering. *IEEE Transactions on Aerospace and Electronic Systems*, 38(3):791–812, 2002.

[24] Jaco Vermaak, Arnaud Doucet, and Partick Pérez. Maintaining multi-modality through mixture tracking. In *Proceedings of the 9th IEEE International Conference on Computer Vision*, 2003.