Time-Sensitive Dirichlet Process Mixture Models

Xiaojin Zhu Zoubin Ghahramani John Lafferty

May 2005
CMU-CALD-05-104

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We introduce Time-Sensitive Dirichlet Process Mixture models for clustering. The models allow infinite mixture components just like standard Dirichlet process mixture models. However they also have the ability to model time correlations between instances.

1 Introduction

Traditional clustering algorithms make two assumptions that are often false in practice: 1. that the number of clusters is known; 2. that the data points are independent. We propose a model that allows infinite number of clusters, and cluster members may have certain dependency in time.

Consider emails received by a user over a period of time. Suppose we want to cluster the emails by topic thread. There are several ways to do this:

- We can sort emails by the 'subject' line. However it is unreliable and we want a more flexible probabilistic model based on email content.
- We can model each thread with a multinomial distribution over the vocabulary, and treat each email as a 'bag of words'. The whole email collection can be modeled as a mixture of multinomial. The problem is that we do not know the number of threads (mixing components). Fixing the number, which is a common practice, seems arbitrary.
- We can model the collection as a Dirichlet process mixture model (DPM) [1]. DPMs allow potentially infinite number of components. Nonetheless DPMs are exchangeable. When applied to emails, this means that old threads never die down. This is undesirable because we want the emails from years ago to have less influence than those from this morning in predicting the next email.

We therefore would like to introduce the concept of time into DPMs, while keeping the ability to model unlimited number of clusters. This is achieved with the proposed Time-Sensitive Dirichlet Process Mixture (tDPM) models.

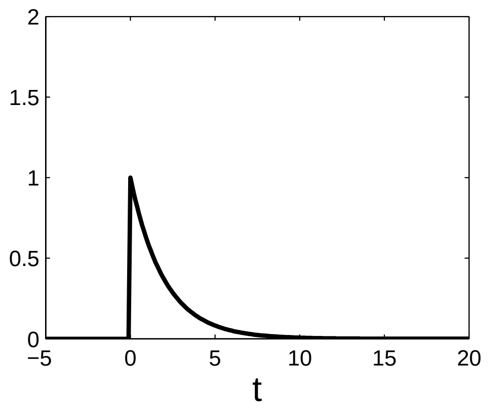
2 The tDPM Framework

Consider a sequence of input d with time stamp t: $(d_1, t_1), \ldots, (d_n, t_n)$, where the time monotonically increases. For concreteness let us assume the d's are email documents, each represented as a bag-of-word vector. Let $s_i \in \{1, 2, \ldots\}$ be the true cluster membership (email thread) of d_i. Notice we do not set the number of clusters a priori. There could potentially be an unlimited number of clusters as the number of documents n grows.

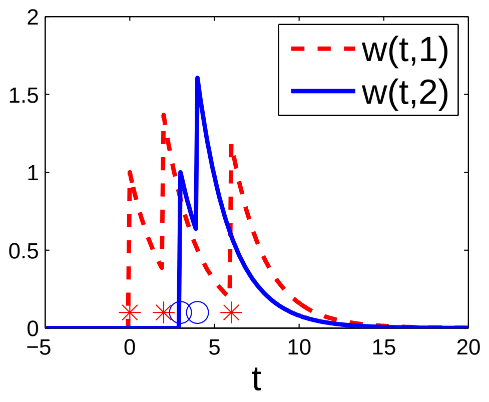
Without loss of generality we assume that each cluster j is represented by a multinomial distribution θ_j over the vocabulary. The probability for cluster j to generate document d_i is then

$$P(d_i|\theta_j) = \prod_{v \in \text{vocabulary}} \theta_j(v)^{d_i(v)} \tag{1}$$

Since past email threads can influence the current email, we want s_i to depend on the history $s_1, \ldots, s_{i-1}$. We also want such dependency to vary with time: older emails should have less influence. We introduce a *weight function* $w(t, j)$ which summarizes



(a)



(b)

Figure 1: (a) The time kernel with $\lambda = 0.5$. (b) The weight functions with data from two clusters, marked as star or circle respectively.

the history at time t. It gives the weight (or 'influence') of cluster j at time t, given the history so far $s_1, \ldots, s_i : t_i < t$,

$$w(t, j) = \sum_{\{i | t_i < t, s_i = j\}} k(t - t_i) \tag{2}$$

Note the weight function is the sum of some time kernel k. In the email example we can use a kernel like $k(t) = \exp(-\lambda t)$ if $t \geq 0$, and $k(t) = 0$ if $t < 0$. This kernel stipulates that an email will boost the probability of the same thread in later emails, but the boost decreases exponentially as specified by the parameter λ. Figure 1(a) shows an example time kernel with $\lambda = 0.5$, while (b) shows two weight functions built upon the kernel. In the example there are documents from cluster 1 at time 0,2,6, and from cluster 2 at time 3,4. Other forms of the time kernel are possible too.

We define the prior probability of assigning cluster j to d_i, given the history $s_1, \ldots, s_{i-1}$, to be

$$\begin{align} & P(s_i = j | s_1, \ldots, s_{i-1}) \tag{3} \\ = \; & P(s_i = j | w(t_i, \cdot)) \tag{4} \\ = \; & \begin{cases} \frac{w(t_i, j)}{\sum_{j'} w(t_i, j') + \alpha} & \text{if } j \text{ is in history} \\ \frac{\alpha}{\sum_{j'} w(t_i, j') + \alpha} & \text{if } j \text{ is new} \end{cases} \tag{5} \end{align}$$

where α is a concentration parameter. We call this a time-sensitive Dirichlet process mixture (tDPM) model. Intuitively if there has been many recent emails from cluster j, the new email will have a large probability also from j. In addition, there is always a possibility that the new email is from a new cluster not seen so far.

tDPM is very similar to the standard Dirichlet process mixture (DPM) models. In fact, it can be shown that if the time kernel k is a step function, then we recover

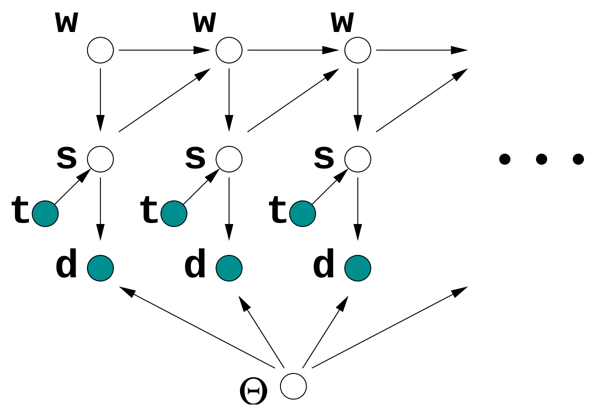


Figure 2: The graphical model for Time-sensitive Dirichlet Process Mixture models. d is the feature (e.g. words of an email), t is the time stamp, s is the cluster label, and w is the sufficient statistic that summarizes the history. Shaded nodes are observed.

the standard DPMs. It is the decaying of k over time that allows us to include time information in to the process. The graphical model representation of tDPM is given in Figure 2.

3 Inference

Given d and t, we would like to infer s. We use a Markov Chain Monte Carlo method. Notice w is a deterministic function of s and t and does not need to be sampled. As shown later if we used conjugate priors, we do not need to actually sample θ but can analytically integrate it out. Therefore we only need to sample s.

In Gibbs sampling, we need to sample s_i from the distribution

$$P(s_i = j | s_{-i}, d_1, \ldots, d_n) \quad \propto \quad P(s_i = j | s_{-i}) P(d_i | d_{-i:s_{-i}=j}) \tag{6}$$

where $d_{-i:s_{-i}=j}$ is the set of documents in cluster $s_i = j$, excluding d_i.

The prior $P(s_i = j|s_{-i})$ in (6) involves all nodes before and after s_i:

$$\begin{aligned} & P(s_i = j | s_{-i}) \\ \propto \;& \left(\prod_{m=1}^{i-1} P(s_m | s_1, \ldots, s_{m-1}) \right) P(s_i = j | s_1, \ldots, s_{i-1}) \left(\prod_{m=i+1}^{n} P(s_m | s_1, \ldots, s_{m-1}) \right) \\ \propto \;& P(s_i = j | s_1, \ldots, s_{i-1}) \left(\prod_{m=i+1}^{n} P(s_m | s_1, \ldots, s_{m-1}) \right) \end{aligned} \tag{7}$$

Substituting in the definition (5), it is easy to show that the denominators are the same for different values of j, and the only difference is in the numerator.

The likelihood term $p(d_i|d_{-i:s_{-i}=j})$ in (6) is domain-specific. For the email task, a Dirichlet-multinomial [2] is the natural choice:

$$p(d_i|d_{-i:s_{-i}=j}) \quad = \quad \int p(d_i|\theta)p(\theta|d_{-i:s_{-i}=j})d\theta \tag{8}$$

where $p(\theta|d_{-i:s_{-i}=j})$ is a posterior Dirichlet distribution. The posterior is derived from a prior (base) Dirichlet distribution G_0, and the observed data $d_{-i:s_{-i}=j}$. Let the Dirichlet prior G_0 be parameterized by βm, where m is a vector over the vocabulary and m sums to 1, and β is the strength of the prior:

$$p(\theta|\beta m) = \frac{\Gamma(\beta)}{\prod_v \Gamma(\beta m_v)} \prod_v \theta_v^{\beta m_v - 1} \tag{9}$$

Treating the document collection $d_{-i:s_{-i}=j}$ as a single, large document, the Dirichlet posterior after observing counts f_v for word v in $d_{-i:s_{-i}=j}$ is

$$p(\theta|d_{-i:s_{-i}=j}) = p(\theta|f, \beta m) = \frac{\Gamma(\sum_v f_v + \beta)}{\prod_v \Gamma(f_v + \beta m_v)} \prod_v \theta_v^{f_v + \beta m_v - 1} \tag{10}$$

And the Dirichlet-multinomial is

$$P(d_i|d_{-i:s_{-i}=j}) \quad = \quad \int p(d_i|\theta)p(\theta|d_{-i:s_{-i}=j})d\theta \tag{11}$$

$$\quad = \quad \frac{\Gamma(\sum_v f_v + \beta)}{\prod_v \Gamma(f_v + \beta m_v)} \frac{\prod_v \Gamma(d_i(v) + f_v + \beta m_v)}{\Gamma(\sum_v d_i(v) + \sum_v f_v + \beta)} \tag{12}$$

Putting everything together for (6), we can fix all other s and sample for s_i. A single Gibbs sampling iteration consists of looping through $i = 1 \dots n$ and sample s_i in turn. The algorithm is given in Figure 3. The time complexity is $O(n^2)$ for each iteration of the Gibbs sampler. If k has limited support, the complexity reduces $O(n)$ but we lose the ability to model long range correlations. Finally we run the Gibbs sampler for many iterations to get the marginals on s.

Some readers may be disturbed by the apparent 'double counting' in Figure 3 when we assign $u(c) = \alpha$ to not only the brand new state c_{new}, but also to states not in $\{s_{<i}\}$ but in $\{s_{>i}\}$. We assure the readers that it is merely an artifact of numbering. If we were to renumber the states at each iteration, we can recover (5) exactly.

4 Parameter Learning

The parameters of the model include the base Dirichlet distribution G_0, the concentration parameter α, and the time kernel parameter λ. We fix the base Dirichlet G_0. For the time being let us assume that all clusters share the same kernel parameter λ. The free parameters are $\Theta = \{\alpha, \lambda\}$.

We learn the parameters by evidence maximization. Since our model is conditioned on time, the evidence is defined as

$$P(D|T, \Theta) = \sum_S P(D|S)P(S|T, \Theta) \tag{13}$$

```
for position $i = 1$ to $n$
    /* $\mathcal{C}$ is the candidate states for $s_i$, */
    /* where $\{s_{-i}\}$ is the set of current states at positions other than $i$, */
    /* and $c_{\text{new}} \notin \{s_{-i}\}$ is a new state, represented by an arbitrary new number. */
    $\mathcal{C} = \{s_{-i}\} \cup \{c_{\text{new}}\}$

    /* Compute the unnormalized probability $p(s_i = c|s_{-i})$ for all candidate $c$ */
    for $c \in \mathcal{C}$
        /* evaluate candidate $s_i = c$ */
        $s_i \leftarrow c$
        /* Prior: the history part. $\{s_{<i}\}$ is the set of states before position $i$ */
        if $c \in \{s_{<i}\}$ then $u(c) = w_c(t_i)$
        else $u(c) = \alpha$
        /* Prior: the future part. */
        for $j = i + 1$ to $n$
            if $s_j \in \{s_{<j}\}$ then $u(c) = u(c) * w_{s_j}(t_j)$
            else $u(c) = u(c) * \alpha$
        end
        /* Likelihood. */
        $u(c) = u(c) * P(d_i|d_{-i:s_{-i}=c})$
    end

    /* pick the state $s_i$ with probability proportional to $u()$ */
    $s_i \sim u(\mathcal{C})$
end
```

Figure 3: A single Gibbs sampling iteration for tDPM

where D is the set of all documents, T is the corresponding set of time stamps, and S is the set of cluster assignments. We want to find the best parameters Θ^* that maximize the evidence:

$$\begin{aligned} \Theta^* &= \arg\max_{\Theta} P(D|T,\Theta) && (14) \\ &= \arg\max_{\Theta} \sum_{S} P(D|S)P(S|T,\Theta) && (15) \end{aligned}$$

We find the parameters with a stochastic EM algorithm. The cluster labels S are hidden variables. Let Θ_0 be the current parameters. We can sample $S^{(1)} \dots S^{(M)}$ from the posterior distribution $P(S|D,T,\Theta_0)$, as detailed in section 3. In generalized EM algorithm, we seek a new parameter Θ which increases the expected log likelihood of the complete data

$$\begin{aligned} Q(\Theta_0,\Theta) &= E_{P(S|D,T,\Theta_0)}\left[\log P(S,D|T,\Theta)\right] && (16) \\ &= E_{P(S|D,T,\Theta_0)}\left[\log P(D|S) + \log P(S|T,\Theta)\right] && (17) \end{aligned}$$

Notice $\log P(D|S)$ does not depend on α, λ. We approximate the expectation by sample average

$$\begin{aligned} Q(\Theta_0,\Theta) &= Const(\Theta) + E_{P(S|D,T,\Theta_0)}\left[\log P(S|T,\Theta)\right] && (18) \\ &\approx Const(\Theta) + \frac{1}{M}\sum_{m=1}^{M} \log P(S^{(m)}|T,\Theta) && (19) \end{aligned}$$

And we find the gradients w.r.t. Θ for parameter update

$$\begin{aligned} \frac{\partial Q}{\partial \Theta} &\approx \frac{\partial}{\partial \Theta}\frac{1}{M}\sum_{m=1}^{M} \log P(S^{(m)}|T,\Theta) && (20) \\ &= \frac{1}{M}\sum_{m=1}^{M}\sum_{i=1}^{N} \frac{\partial}{\partial \Theta} \log P(s_i^{(m)}|s_1^{(m)} \dots s_{i-1}^{(m)}, T, \Theta) && (21) \end{aligned}$$

where $P(s_i^{(m)}|s_1^{(m)} \dots s_{i-1}^{(m)}, T, \Theta)$ is defined in (5). The gradients are:

$$\frac{\partial}{\partial \alpha} \log P(s_i|s_1 \dots s_{i-1}, T, \Theta) = \begin{cases} -\frac{1}{\sum_c w(t_i,c)+\alpha} & \text{if } s_i \text{ in history} \\ \frac{1}{\alpha} - \frac{1}{\sum_c w(t_i,c)+\alpha} & \text{if } s_i \text{ new} \end{cases} \qquad (22)$$

$$\begin{aligned} &\frac{\partial}{\partial \lambda} \log P(s_i|s_1 \dots s_{i-1}, T, \Theta) \\ &= \begin{cases} \frac{\frac{\partial}{\partial \lambda} w(t_i,s_i)}{w(t_i,s_i)} - \frac{\sum_c \frac{\partial}{\partial \lambda} w(t_i,c)}{\sum_c w(t_i,c)+\alpha} & \text{if } s_i \text{ in history} \\ -\frac{\sum_c \frac{\partial}{\partial \lambda} w(t_i,c)}{\sum_c w(t_i,c)+\alpha} & \text{if } s_i \text{ new} \end{cases} \end{aligned} \qquad (23)$$

where

$$w(t,c) = \sum_{i:t_i<t,s_i=c} k(t-t_i) = \sum e^{-\lambda(t-t_i)} \tag{24}$$

$$\frac{\partial}{\partial\lambda} w(t,c) = \sum_{i:t_i<t,s_i=c} -(t-t_i)e^{-\lambda(t-t_i)} \tag{25}$$

We then take a gradient step in the M-step of the generalized EM algorithm to improve the log likelihood.

5 Experiments

We create synthetic datasets which have explicit time dependency between instances, and use them to illustrate the time sensitivity of tDPM models.

All synthetic datasets have $n = 100$ instances. We first create the time stamps of each instances by sampling from a Poisson process. In particular, the interval between two consecutive time stamps has an exponential distribution with mean $1/\gamma = 1$:

$$p(t_{i+1} - t_i) = \gamma e^{-\gamma(t_{i+1}-t_i)} \tag{26}$$

For the instance d_i at time t_i, its state s_i is sampled from the conditional distribution (5). We use an exponential function as the kernel k,

$$k(t) = e^{-0.5t}, t \geq 0 \tag{27}$$

and the concentration parameter α is set to 0.2. This emulates the situation where new clusters are created from time to time, and a cluster stays alive better if many preceding instances are from the cluster.

If a new cluster c is created, we sample its multinomial distribution θ_c from the base distribution G_0. The base distribution G_0 is a flat Dirichlet on a vocabulary of size three: $G_0 \sim Dir(1,1,1)$, so that all multinomials are equally likely. Finally documents are sampled from their corresponding multinomial θ, where all documents have the same length $|d|$. We create two datasets with document length $|d|$ equals 20 and 50 respectively, with everything else being the same. Given that the vocabulary size is 3, they correspond to somewhat hard (less words) and easy (more words) datasets respectively. Figure 4 shows time vs. cluster plots of the two datasets. Notice documents from the same cluster tend to group together in time, which fits our intuition on real world problems like emails.

During evaluation, the input to various algorithms are the documents d_i and their time stamps t_i, and the goal is to infer the clustering s_i. Notice the true number of clusters is not given to the algorithms.

For the tDPM model, we assume we know the true base distribution $G_0 \sim Dir(1,1,1)$, concentration parameter $\alpha = 0.2$, and the kernel $k(t) = e^{-0.5t}$. We run the Gibbs sampler with initial states $s_1 = \ldots = s_n = 1$. Each MCMC iteration updates $s_1, \ldots, s_n$ once, and thus consists of n Gibbs steps. We ignore the burn-in period of the first 100 MCMC iterations, and then take out a sample of $s_1, \ldots, s_n$ every 11 iterations. In this experiment we take out 109 samples altogether. We evaluate the performance of tDPM by three measures:

1. Number of clusters discovered. Notice each sample $s_1, \ldots, s_n$ is a clustering of the data, and different samples may have different number of clusters. In fact Figure 5(a,b) shows the distribution of number of clusters in the 109 samples, on the hard ($|d| = 20$) and easy ($|d| = 50$) synthetic datasets respectively. The modes are at 12 and 15, very close to the true values of 12 and 14 respectively.

2. Confusion matrix. One way to combine the samples with possibly different number of clusters is to compute the $n \times n$ confusion matrix M, where M_{ij} is the probability that i, j are in same cluster. This can be easily estimated from the 109 samples by the frequency of i, j in the same cluster. Ideally M should be similar to the 'true confusion matrix' M^*, defined by $M^*_{ij} = 1$ if the true cluster has label $s_i = s_j$, and 0 otherwise. In Figure 5(c,d) we plot the true confusion matrices M^*. Notice we sort the instances by their true cluster for better visualization. In Figure 5(e,f) we plot the tDPM confusion matrices, using the same order. They are reasonably similar.

3. Variation of Information. We compute the variation of information measure [3] between the true clustering and each sample clustering. We list the mean and standard deviation for the two synthetic datasets: (hard) 0.9272 ± 0.1718, (easy) 0.1245 ± 0.0911.

We compare tDPM to a standard DPM model, by using a step function as the kernel k. Again we assume we know the true base distribution $G_0 \sim Dir(1, 1, 1)$, and concentration parameter $\alpha = 0.2$. The Gibbs sampling is done exactly the same as in tDPM. We find that

1. Number of clusters discovered. Figure 6(a,b) shows the distribution of number of clusters with DPM. DPM discovers fewer clusters than tDPM. The modes are at 6(or 7) and 9 respectively, and the true values are 12 and 14.

2. Confusion matrix. In Figure 6(c,d) we plot the DPM confusion matrices. Notice they are much less similar to the true matrices.

3. Variation of Information. With DPM we have (hard) 1.8627 ± 0.1753, (easy) 0.6630 ± 0.1144. This means the sample clusterings are significantly farther away from the true clustering, compared to tDPM.

To summarize, tDPM is better than the standard DPM model, when the instances have a time dependency.

6 Discussions

The tDPM model is a way to take time into consideration. Notice it is different than simply adding time as a new feature for cluster.

The tDPM is not time reversible nor exchangeable in general. This is different from the standard DPM. It is both a blessing and curse. It allows for the modeling of time, but at the expense of computation.

There are many ways one can extend the tDPM model proposed here:

- The time kernel k can have different forms. For example, different clusters can have different decay rate λ. More interestingly, k can even be periodic to model repetitive emails like weekly meeting announcements.
- Currently the models for each cluster are stationary and do not evolve over time. This can potentially be relaxed.
- One can have a generative model on time dependencies. For example one can assume a Poisson process on cluster, and then a non-homogeneous Poisson process on the documents within the cluster.

References

[1] Radford M. Neal. Markov chain sampling methods for dirichlet process mixture models. Technical Report Technical Report No. 9815, Dept. of Statistics, University of Toronto, 1998.

[2] D. J. C. MacKay and L. Peto. A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):1–19, 1994.

[3] Marina Meila. Comparing clusterings. In *COLT*, 2003.

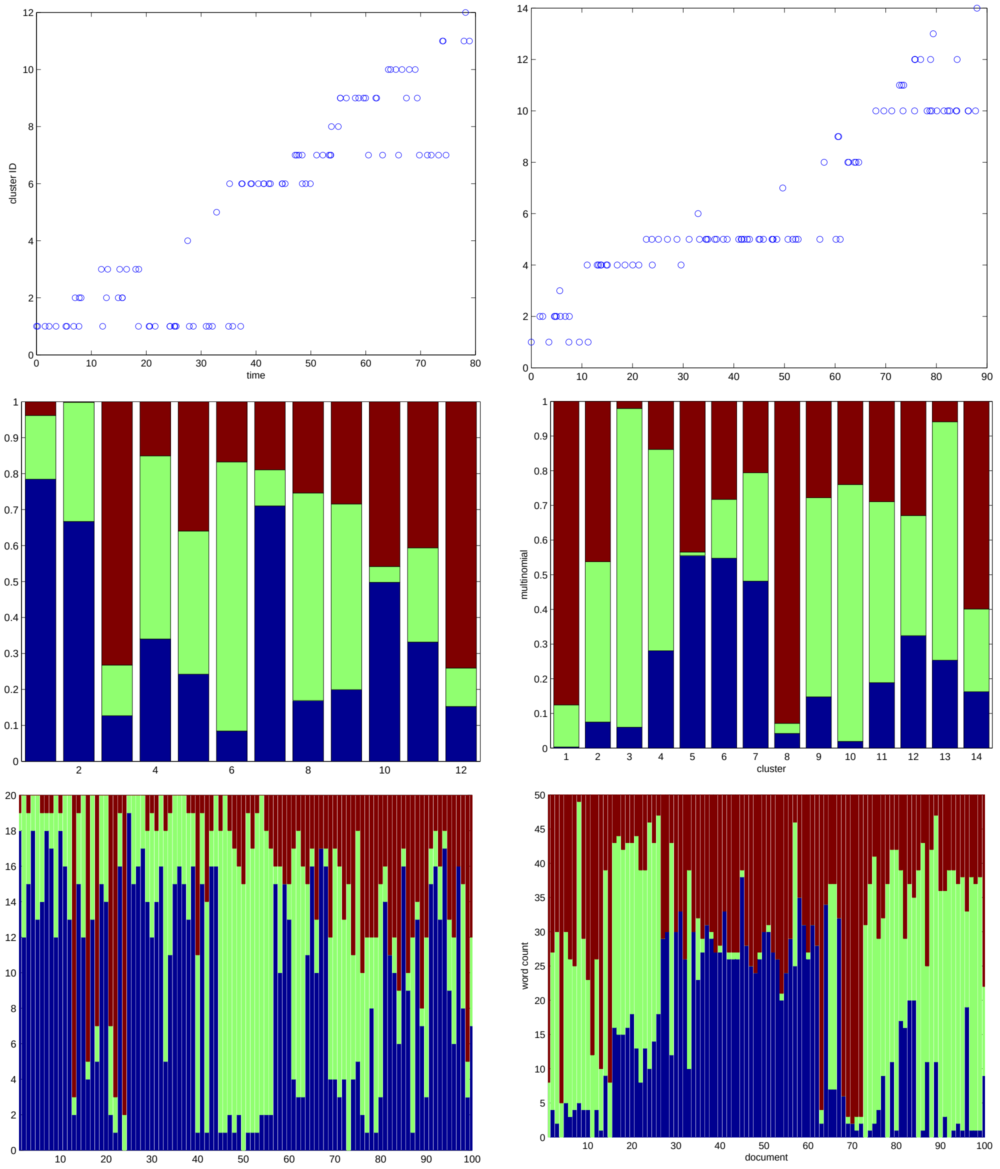


Figure 4: Two synthetic datasets with $|d| = 20$ (left) and $|d| = 50$ (right) respectively. Top row: Time stamps t_i vs. cluster ID s_i; Middle row: the cluster multinomials θ_c; Bottom row: word counts for each document d_i.

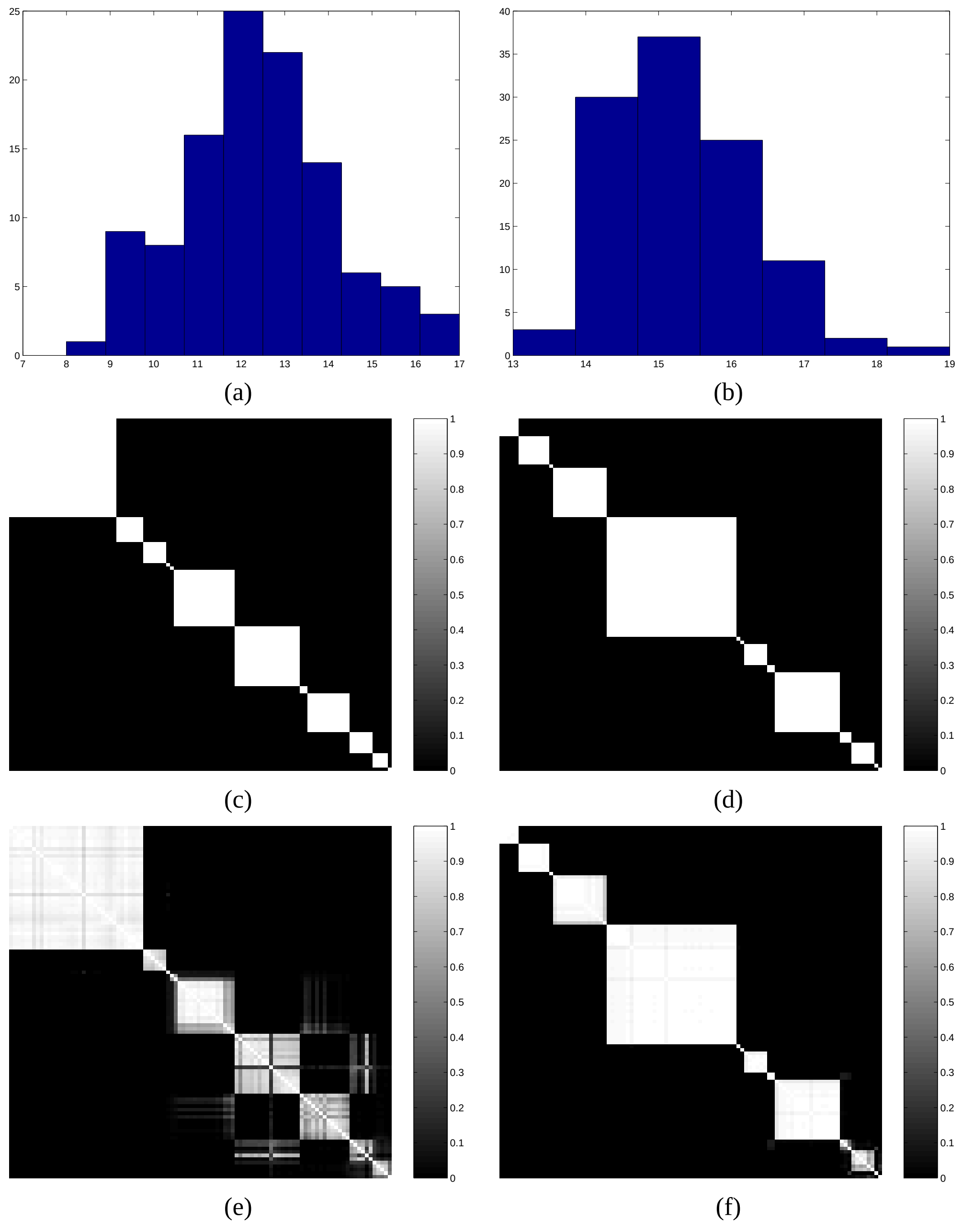


Figure 5: tDPM results on the hard ($|d| = 20$, left) and easy ($|d| = 50$, right) synthetic datasets. (a, b) Number of clusters discovered in MCMC samples; (c, d) Confusion matrix with true cluster labels; (e, f) Confusion matrix from tDPM MCMC samples.

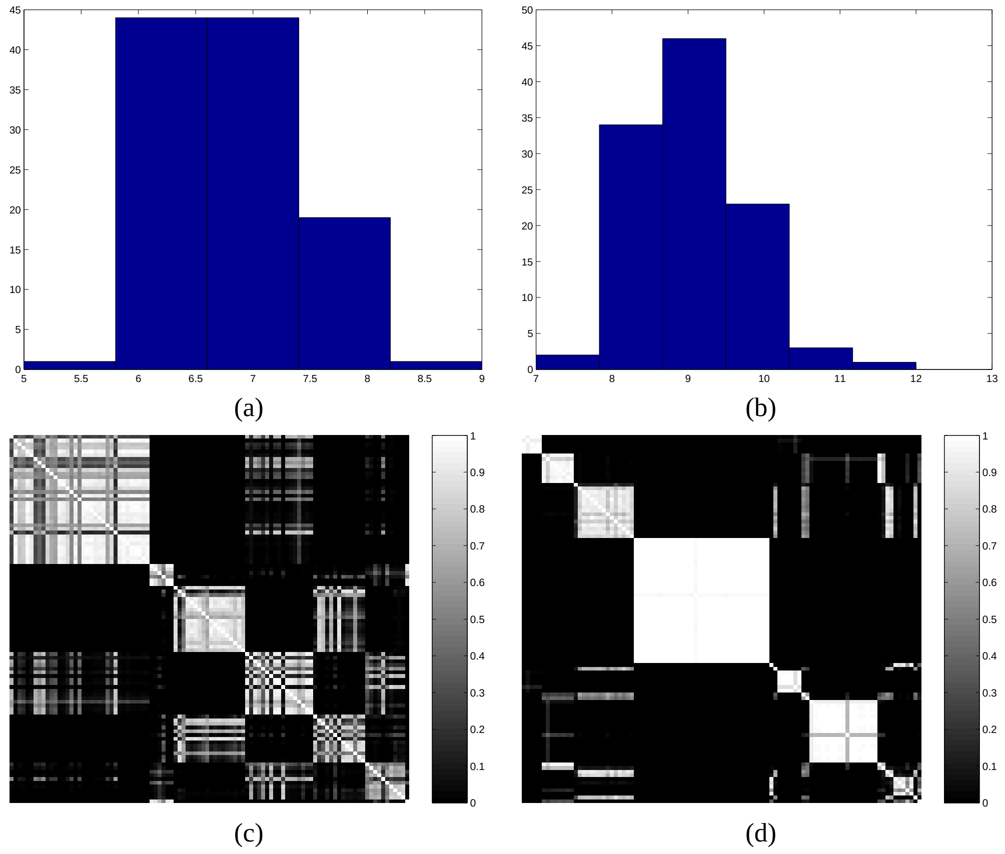


Figure 6: Standard DPM results on the hard ($|d| = 20$, left) and easy ($|d| = 50$, right) synthetic datasets. (a, b) Number of clusters discovered in MCMC samples; (c, d) Confusion matrix from DPM MCMC samples.