Document Clustering using Linear Partitioning Hyperplanes and Reallocation

Canasai Kruengkrai, Virach Sornlertlamvanich, Hitoshi Isahara
Thai Computational Linguistics Laboratory
National Institute of Information and Communications Technology
112 Paholyothin Road, Klong 1, Klong Luang, Pathumthani 12120, Thailand
{canasai,virach}@tcllab.org, isahara@nict.go.jp

ABSTRACT

This paper presents a novel algorithm for document clustering based on a combinatorial framework of the Principal Direction Divisive Partitioning (PDDP) algorithm [1] and a simplified version of the EM algorithm called the spherical Gaussian EM (sGEM) algorithm. The idea of the PDDP algorithm is to recursively split data samples into two subclusters using the hyperplane normal to the principal direction derived from the covariance matrix. However, the PDDP algorithm can yield poor results, especially when clusters are not well-separated from one another. To improve the quality of the clustering results, we deal with this problem by re-allocating new cluster membership using the sGEM algorithm with different settings. Furthermore, based on the theoretical background of the sGEM algorithm, we can naturally extend the framework to cover the problem of estimating the number of clusters using the Bayesian Information Criterion. Experimental results on two different corpora are given to show the effectiveness of our algorithm.

1. INTRODUCTION

Unsupervised clustering has been applied to various tasks in the field of Information Retrieval (IR). One of the challenging problems is document clustering that attempts to discover meaningful groups of documents where those within each group are more closely related to one another than documents assigned to different groups. The resulting document clusters can provide a structure for organizing large bodies of text for efficient browsing and searching [15].

A wide variety of unsupervised clustering algorithms has been intensively studied in the document clustering problem. Among these algorithms, the iterative optimization clustering algorithms have demonstrated reasonable performance for document clustering, e.g. the Expectation-Maximization (EM) algorithm and its variants, and the well-known k-means algorithm. Actually, the k-means algorithm can be considered as a spacial case of the EM algorithm [3] by assuming that each cluster is modeled by a spherical Gaussian, each sample is assigned to a single cluster, and all mixing parameters (or prior probabilities) are equal. The competitive advantage of the EM algorithm is that it is fast, scalable, and easy to implement. However, one major drawback is that it often gets stuck in local optima depending on the initial random partitioning. Several techniques have been proposed for finding good starting clusters (see [3][7]).

Recently, Boley [1] has developed a hierarchal clustering algorithm called the Principal Direction Divisive Partitioning (PDDP) algorithm that performs by recursively splitting data samples into two sub-clusters. The PDDP algorithm has several interesting properties. It applies the concept of the Principal Component Analysis (PCA) but only requiring the principal eigenvector, which is not computationally expensive. It can also generate a hierarchal binary tree that inherently produces a simple taxonomic ontology. Clustering results produced by the PDDP algorithm compare favorably to other document clustering approaches, such as the agglomerative hierarchal algorithm and associative rule hypergraph clustering. However, the PDDP algorithm can yield poor results, especially when clusters are not well-separated from one another. This problem will be described in depth later.

In this paper, we propose a novel algorithm for document clustering based on a combinatorial framework of the PDDP algorithm and a variant of the EM algorithm. As discussed above, each algorithm has its own strengths and weaknesses. We are interested in the idea of the PDDP algorithm that uses the PCA for analyzing the data. More specifically, it splits the data samples into two sub-clusters based on the hyperplane normal to the principal direction derived from the covariance matrix of the data. When the principal direction is not representative, the corresponding hyperplane tends to produce individual clusters with wrongly partitioned contents. One practical way to deal with this problem is to run the EM algorithm on the partitioning results. We present a simplified version of the EM algorithm called the spherical Gaussian EM algorithm for performing such task. Furthermore, based on the theoretical background of the spherical Gaussian EM algorithm, we can naturally extend the framework to cover the problem of estimating the number of clusters using the Bayesian Information Criterion (BIC) [9].

The rest of this paper is organized as follows. Section 2 briefly reviews some important backgrounds of the PDDP algorithm, and addresses the problem causing the incorrect partitioning. Section 3 presents the spherical Gaussian EM algorithm, and describes how to combine it with the PDDP algorithm. Section 4 discusses the idea of applying the BIC to our algorithm. Section 5 explains the data sets and the evaluation method, and shows experimental results. Finally, we conclude in Section 6 with some directions of future work.

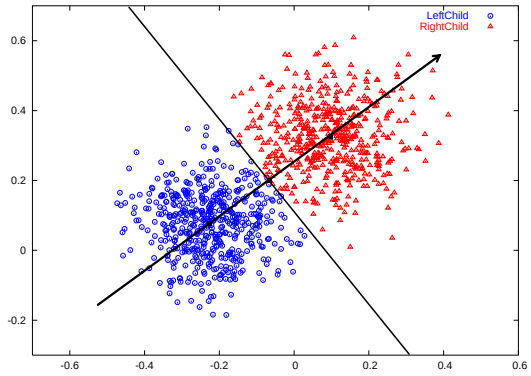

Figure 1: The principal direction and the linear partitioning hyperplane on the 2d2k data set.

2. DOCUMENT CLUSTERING VIA LINEAR PARTITIONING HYPERPLANES

Suppose we have a 1-dimensional data set, e.g. real numbers on a line. The question is how to split this data set into two groups. One simple solution may be the following procedures. We first find the mean value of the data set, and then compare each point with the mean value. If the point value is less the mean value, it is assigned to the first group. Otherwise, it is assigned to the second group. The problem arises when we have a d-dimensional data set. Based on the idea of the PDDP algorithm, we can deal with this problem by projecting all the data points onto the principal direction (the principal eigenvector of the covariance matrix of the data set), and then the splitting process can be performed based on this principal direction. In geometric terms, the data points are partitioned into two sub-clusters using the hyperplane normal to the principal direction passing through the mean vector [1]. We refer to this hyperplane as the linear partitioning hyperplane. Figure 1 illustrates the principal direction and the linear partitioning hyperplane on the `2d2k` data set, containing 1000 points distributed in 2 Gaussians[1].

Given a matrix $\mathbf{M}$, we obtain the covariance matrix $\mathbf{C} = (\mathbf{M} - \mathbf{m}\boldsymbol{e}^T)(\mathbf{M} - \mathbf{m}\boldsymbol{e}^T)^T = \mathbf{A}\mathbf{A}^T$, where $\mathbf{A} = \mathbf{M} - \mathbf{m}\boldsymbol{e}^T$, $\mathbf{m}$ is the mean vector of $\mathbf{M}$, and $\boldsymbol{e}$ is the vector of ones, $(1, \ldots, 1)^T$. To obtain the principal eigenvector of $\mathbf{C}$, the notion of the Singular Value Decomposition (SVD) is used. The SVD of $\mathbf{A}$ is the factorization $\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$, where $\mathbf{U}^T\mathbf{U} = \mathbf{V}^T\mathbf{V} = \mathbf{I}$, and $\boldsymbol{\Sigma}$ is a diagonal matrix, whose non-negative values are sorted in decreasing order. It is known that the left and right singular vectors of $\mathbf{A}$ are equivalent to the eigenvectors of $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$, respectively. If we write $\mathbf{A}$ in terms of the SVD, we get

$$\mathbf{A}\mathbf{A}^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T\mathbf{V}\boldsymbol{\Sigma}^T\mathbf{U}^T = \mathbf{U}\boldsymbol{\Sigma}\mathbf{I}\boldsymbol{\Sigma}^T\mathbf{U}^T = \mathbf{U}\boldsymbol{\Sigma}^2\mathbf{U}^T \ .$$

Thus, the columns of $\mathbf{U}$, known as the left singular vectors, are the eigenvectors of $\mathbf{A}\mathbf{A}^T$. In our work, we just require the principal eigenvector, which is the first left singular vector denoted by $\mathbf{u}_1$. To efficiently compute a partial SVD of a matrix, the Lanczos algorithm is applied (see [6] for more details).

[1]The `x2d2k` data set is available at `http://www.lans.ece.utexas.edu/~strehl/data.html`.

Let $\mathbf{d}_i$ be a document vector, and $\mathcal{C}$ be a cluster, where $\mathcal{C} = \{\mathbf{d}_1, \ldots, \mathbf{d}_{|\mathcal{C}|}\}$. The principal direction of $\mathcal{C}$ is denoted by $\mathbf{u}_\mathcal{C}$. The mean (or centroid) vector of the cluster can be calculated as follows:

$$\mathbf{m}_\mathcal{C} = \frac{1}{|\mathcal{C}|}\sum_{i=1}^{|\mathcal{C}|}\mathbf{d}_i \ . \tag{1}$$

The linear partitioning hyperplane for splitting $\mathcal{C}$ into the left child $\mathcal{L}$ and right child $\mathcal{R}$ corresponds to the following discriminant function [1]:

$$f_\mathcal{C}(\mathbf{d}_i) = \mathbf{u}_\mathcal{C}^T(\mathbf{d}_i - \mathbf{m}_\mathcal{C}) \ , \tag{2}$$

where

$$\mathbf{d}_i \in \begin{cases} \mathcal{L}, & \text{if } f_\mathcal{C}(\mathbf{d}_i) \leq 0, \\ \mathcal{R}, & \text{if } f_\mathcal{C}(\mathbf{d}_i) > 0. \end{cases} \tag{3}$$

The PDDP algorithm begins with all the document vectors in a large single cluster, and proceeds by recursively splitting the cluster into two sub-clusters using the linear partitioning hyperplane according to the discriminant function in Equations 2 and 3. The algorithm stops splitting based on some heuristic, e.g. a predefined number of clusters. It finally yields a binary tree, whose leaf nodes form the resulting clusters. To keep the binary tree balanced, it selects an un-split cluster to split by using the scatter value, measuring the average distance from the data points in the cluster to their centroid. The basic PDDP algorithm is given in Algorithm 1.

As mentioned earlier, the problem of the PDDP algorithm is that it cannot achieve good results when clusters are not well-separated from one another. Let us describe with an empirical example. Figure 2 shows two partitions produced by performing the first iteration of the PDDP algorithm on a 2-dimensional data set[2]. The data set consists of 334 points. The actual class labels are not given, but one can observe that it is composed of five compact clusters [8]. Based on the principal direction and the corresponding linear partitioning hyperplane, we can see that the PDDP algorithm starts with significantly wrong partitioning on the middle left-hand cluster. Figure 3 shows three partitions after the second iteration. If we further perform the partitioning without making some adjustments, the resulting clusters become worse. This indicates that the basic PDDP algorithm can produce poor solutions in some distributions of the data, which we do not know in advance. Also, we may require some information to suggest whether to split a particular cluster further or keep it as it is.

3. REFINING PARTITIONING WITH RE-ALLOCATION

In practice, it is possible to refine the partitioning results by re-allocating new cluster membership. The basic idea of the reallocation method [12] is to start from some initial partitioning of the data set, and then proceed by moving objects from one cluster to another cluster to obtain an improved partitioning. Thus, any iterative optimization clustering algorithm can be applied to do such operation. We formulate our problem as a finite mixture model, and apply a variant

[2]This data set is available at `http://www.jihe.net/datasets.htm`.

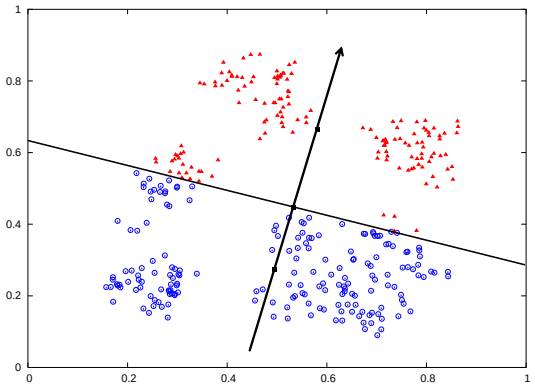

Figure 2: Two partitions after the first iteration.

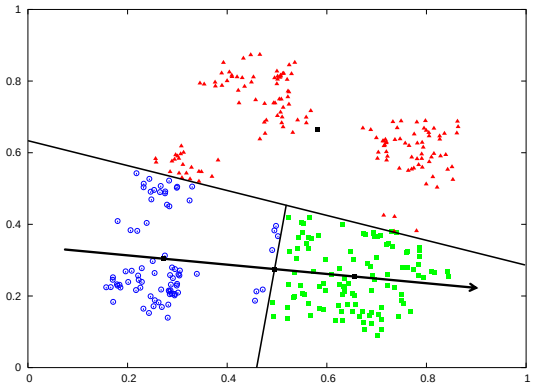

Figure 3: Three partitions after the second iteration.

of the EM algorithm for learning the model. We assume that each document vector $\mathbf{d}_i$ is generated independently from a probability density function:

$$p(\mathbf{d}_i|\mathbf{\Theta}) = \sum_{j=1}^{k} p(\mathbf{d}_i|c_j;\boldsymbol{\theta}_j)P(c_j) \ , \qquad (4)$$

where k is the number of components, the conditional probability density function $p(\mathbf{d}_i|c_j;\boldsymbol{\theta}_j)$ is the component density, and the prior probability $P(c_j)$ is the mixing parameter. Thus, we have the model parameters:

$$\mathbf{\Theta} = \{P(c_1),\ldots,P(c_k);\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_k\} \ .$$

To estimate the model parameters, we work with the maximum likelihood (ML) estimation. Given a set of document vectors $\mathcal{D} = \{\mathbf{d}_1,\ldots,\mathbf{d}_n\}$, the incomplete-data log likelihood can be expressed as:

$$\begin{aligned} \log L(\mathbf{\Theta}) &= \log p(\mathcal{D}|\mathbf{\Theta}) \\ &= \log \prod_{i=1}^{n} p(\mathbf{d}_i|\mathbf{\Theta}) \\ &= \sum_{i=1}^{n} \log \sum_{j=1}^{k} p(\mathbf{d}_i|c_j;\boldsymbol{\theta}_j)P(c_j) \ . \qquad (5) \end{aligned}$$

Our objective is to maximize $\log L$. However, Equation 5 is difficult to optimize because it contains the log of sums.

Algorithm 1: The basic PDDP algorithm.

input : A data set representing by a matrix $\mathbf{M} = (\mathbf{d}_1,\mathbf{d}_2,\ldots,\mathbf{d}_n)$, and a desired number of clusters $k_{\max}$.

output : A binary tree $\mathcal{T}$ with leaf nodes forming a partitioning of the data set.

begin

Initialize a binary tree $\mathcal{T}$ with a single root node, and $k \leftarrow 0$.

while $k < k_{max}$ **do**

Select the leaf node $\mathcal{C}$ with the largest scatter value.

Compute the mean vector $\mathbf{m}_{\mathcal{C}}$ and the principal direction $\mathbf{u}_{\mathcal{C}}$.

For each document $\mathbf{d}_i \in \mathcal{C}$, assign $\mathbf{d}_i$ to $\mathcal{L}$ or $\mathcal{R}$ according to Equations 2 and 3.

Set $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{L},\mathcal{R}\}$, and $k \leftarrow k+1$.

end

end

Thus, we introduce the variable of the missing data

$$\mathcal{Z} = \{\mathbf{z}_1,\mathbf{z}_2,\ldots,\mathbf{z}_n\},$$

where $\mathbf{z}_i = (z_{i1},\ldots,z_{ik})^T$ is a vector of binary indicator variables, and $(\mathbf{z}_i)_j = z_{ij} = 1$ if the vector $\mathbf{d}_i$ was generated from the component c_j; otherwise $z_{ij} = 0$. Now we can write the complete-data log likelihood as:

$$\log L_c(\mathbf{\Theta}) = \sum_{i=1}^{n}\sum_{j=1}^{k} z_{ij} \log(p(\mathbf{d}_i|c_j;\boldsymbol{\theta}_j)P(c_j)) \ . \qquad (6)$$

Since $\mathcal{Z}$ is unknown, we instead work with its expectation. A local maximum of $\log L_c$ can be found by applying the EM algorithm that performs by iterating two steps:

- **E-step:** $\hat{\mathcal{Z}}^{\langle t+1\rangle} = E[\mathcal{Z};\hat{\mathbf{\Theta}}^{\langle t\rangle}]$,
- **M-step:** $\hat{\mathbf{\Theta}}^{\langle t+1\rangle} = \text{argmax}_{\mathbf{\Theta}}\Big(\log L_c(\mathbf{\Theta};\hat{\mathcal{Z}}^{\langle t+1\rangle})\Big)$.

So far, the problem is how to estimate the model parameters. Here we assume that the data samples are drawn from the multivariate normal density in $\Re^d$. We also assume that features are statistically independent, and a component c_j generates its members from the spherical Gaussian with the same covariance matrix [5]. Thus, the parameter for each component c_j becomes $\boldsymbol{\theta}_j = (\mathbf{m}_j,\sigma^2\mathbf{I}_d)$, and the component density is:

$$p(\mathbf{d}_i|c_j;\boldsymbol{\theta}_j) = \frac{1}{(2\pi)^{d/2}\sigma^d} \exp\Big(-\frac{\| \mathbf{d}_i - \mathbf{m}_j \|^2}{2\sigma^2}\Big) \ . \qquad (7)$$

Based on Bayes' theorem, the probability that $\mathbf{d}_i$ falls into c_j can be defined as the product of the component density and the mixing parameter:

$$P(c_j|\mathbf{d}_i;\boldsymbol{\theta}_j) = p(\mathbf{d}_i|c_j;\boldsymbol{\theta}_j)P(c_j) \ . \qquad (8)$$

Note that we estimate this probability with $\log(P(c_j|\mathbf{d}_i;\boldsymbol{\theta}_j))$ that corresponds to the log term in Equation 6. Algorithm 2 gives an outline of a simplified version of the EM algorithm. From the assumptions of the model, we refer to this algorithm as the spherical Gaussian EM algorithm (or the sGEM

Algorithm 2: The spherical Gaussian EM algorithm.

begin

Initialization: Set $(\mathbf{z}_i)_j^{\langle 0\rangle}$ from a partitioning of the data, and $t \leftarrow 0$.

repeat

E-step: For each $\mathbf{d}_i, 1 \le i \le n$ and $c_j, 1 \le j \le k$, find its new component index as:

$$(\mathbf{z}_i)_j^{\langle t+1\rangle} = \begin{cases} 1, & \text{if } j^* = \text{argmax}_j \log(P^{\langle t\rangle}(c_j|\mathbf{d}_i;\boldsymbol{\theta}_j)), \\ 0, & \text{otherwise.} \end{cases}$$

M-step: Re-estimate the model parameters:

$$P(c_j)^{\langle t+1\rangle} = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{z}_i)_j^{\langle t+1\rangle}$$

$$\mathbf{m}_j^{\langle t+1\rangle} = \sum_{i=1}^{n}\mathbf{d}_i(\mathbf{z}_i)_j^{\langle t+1\rangle} / \sum_{i=1}^{n}(\mathbf{z}_i)_j^{\langle t+1\rangle}$$

$$\sigma^{2\langle t+1\rangle} = \frac{1}{n\cdot d}\sum_{i=1}^{n}\sum_{j=1}^{k} \parallel \mathbf{d}_i - \mathbf{m}_j \parallel^2 (\mathbf{z}_i)_j^{\langle t+1\rangle} .$$

until $\Delta \log L_c(\boldsymbol{\Theta}) < \delta$;

end

algorithm for short). The algorithm tries to maximize $\log L_c$ at very step, and iterates until convergence. For example, the algorithm terminates when $\Delta \log L_c < \delta$, where δ is a predefined threshold. In this paper, we set $\delta = 0.001$.

The sGEM algorithm can be combined with the PDDP algorithm in several ways. We can run the sGEM algorithm on the local region after the splitting process by fixing $k = 2$. This idea is closely related to the concept of the bisecting k-means algorithm [14]. However, our initial partitions are based on the PDDP algorithm rather than using the random initialization. We can also run the sGEM algorithm at the end of the PDDP algorithm, where k is equal to the number of leaf nodes of the PDDP's binary tree. In other words, we can think that the sGEM algorithm can get trapped in a local optimum due to a bad initialization, so performing the PDDP algorithm as the first round of the sGEM algorithm may help to avoid this problem. The computational time is reasonable. Since the initial partitions generated by the PDDP algorithm are better than random ones, the combined algorithm performs a moderate number of the sGEM iterations.

4. ESTIMATING NUMBER OF DOCUMENT CLUSTERS

When we need to apply the clustering algorithm to a new data set having little knowledge about its contents, fixing a predefined number of clusters is too strict and inefficient to discover the latent cluster structures. Based on the finite mixture model described in the previous section, we can naturally extend the combination of the PDDP algorithm and the sGEM algorithm to cover the problem of estimating the number of clusters in the data set. Here we apply a model selection technique called the Bayesian Information Criterion (BIC) [9]. Generally, the problem of model selection is to choose the best one among a set of candidate models.

Algorithm 3: The combination of the PDDP and sGEM algorithms using the BIC as the stopping criterion.

input : A data set representing by a matrix $\mathbf{M} = (\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_n)$.

output : A set of disjoint clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_k$.

begin

Initialize a binary tree $\mathcal{T}$ with a single root node, and $k \leftarrow 0$.

$BIC_{global} \leftarrow BIC$(root node)

repeat

Select the leaf node $\mathcal{C}$ with the largest scatter value.

Compute the mean vector $\mathbf{m}_\mathcal{C}$ and the principal direction $\mathbf{u}_\mathcal{C}$.

For each document $\mathbf{d}_i \in \mathcal{C}$, assign $\mathbf{d}_i$ to $\mathcal{L}$ or $\mathcal{R}$ according to Equations 2 and 3.

Run sGEM($\{\mathcal{L},\mathcal{R}\}$) for the local refinement.

if $BIC(\{\mathcal{L},\mathcal{R}\}) > BIC(\mathcal{C})$ **then**

$\mathcal{T}_{tmp} \leftarrow$ all the current leaf nodes except node $\mathcal{C}$.

$BIC_{tmp} \leftarrow BIC(\mathcal{T}_{tmp} \cup \{\mathcal{L},\mathcal{R}\})$

if $BIC_{tmp} > BIC_{global}$ **then**

$BIC_{global} \leftarrow BIC_{tmp}$

Set $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathcal{L},\mathcal{R}\}$, and $k \leftarrow k+1$.

until *the leaf nodes cannot be partitioned*;

Initialize $\mathcal{C}_1^{\langle 0\rangle}, \mathcal{C}_2^{\langle 0\rangle}, \ldots, \mathcal{C}_k^{\langle 0\rangle}$ with leaf nodes of $\mathcal{T}$, and $t \leftarrow 0$.

Run sGEM($\{\mathcal{C}_j^{\langle t\rangle}\}_{j=1}^k$) until convergence.

end

Given a data set $\mathcal{D}$, the BIC of a model $\mathcal{M}_i$ is defined as:

$$BIC(\mathcal{M}_i) = \log L(\mathcal{D}|\mathcal{M}_i) - \frac{p_i}{2}\cdot \log|\mathcal{D}| \ , \qquad (9)$$

where $\log L(\mathcal{D}|\mathcal{M}_i)$ is the log likelihood of the data according to the model $\mathcal{M}_i$, and p_i is the number of independent parameters to be estimated in the model $\mathcal{M}_i$. The BIC contains two components, where the first term measures how well the parameterized model predicts the data, and the second term penalizes the complexity of the model [4]. Thus, we select the model having the largest value of the BIC, $\mathcal{M}^* = \text{argmax}_i BIC(\mathcal{M}_i)$.

In the context of document clustering, the log likelihood of the data can be derived from Equation 6. As a result, we directly obtain the value of the first term of the BIC from running the sGEM algorithm. However, we can also compute it from the data according to the partitioning. The number of parameters is the sum of $k-1$ component probabilities, $k \cdot d$ centroid coordinates, and 1 variance.

Boley's subsequent work [2] also suggests a dynamic threshold called the centroid scatter value (CSV) for estimating the number of clusters. This criterion is based on the distribution of the data. Since the PDDP algorithm is a kind of the divisive hierarchical clustering algorithm, it gradually produces a new cluster by splitting the existing clusters. As the PDDP algorithm proceeds, the clusters get smaller. Thus, the maximum scatter value in any individual cluster

also gets smaller. The idea of the CSV is to compute the overall scatter value of the data by treating the collection of centroids as individual data vectors. This stopping test terminates the algorithm when the CSV exceeds the maximum cluster scatter value at any particular point.

We can think of the CSV as a value that captures the overall improvement, whereas the BIC can be used to measure the improvement in both the local and global structure. As mentioned earlier, in the splitting process, we need some information to make the decision whether to split a cluster into two sub-clusters or keep its current structure. We first calculate the BIC locally when the algorithm performs the splitting test in the cluster. The BIC is calculated globally to measure the overall structure improvement. If both the local and global BIC scores improve, we then split the cluster into two children clusters. Algorithm 3 describes the unified algorithm based on the PDDP and sGEM algorithms using the BIC as the stopping criterion.

5. EXPERIMENTS

5.1 Data Sets and Experimental Setup

The `Yahoo News` (`K1a`) data set[3] consists of 2340 news articles obtained from the Web in October 1997. Each document corresponds to a web page listed in the subject hierarchy of YAHOO! This data set is divided into 20 categories: `Health`, `Entertainment` (including sub-categories: `art`, `cable`, `culture`, `film`, `industry`, `media`, `multimedia`, `music`, `online`, `people`, `review`, `stage`, `television`, `variety`), `Sports`, `Politics`, `Technology`, and `Business`. Note that the number of documents in each category varies considerably, ranging from 9 to 494.

The `20 Newsgroups` data set consists of 20000 articles evenly divided among 20 different discussion groups [10]. This data set is collected from UseNet postings over a period of several months in 1993. Many categories fall into confusable clusters. For example, five of them are computer discussion groups, and three of them discuss religion. We used the `Bow` toolkit [11] to construct the term-document matrix (sparse format). We removed the UseNet headers, and also eliminated the stopwords and low-frequency words (occurring less than 2 times). We finally obtained the 59965×19950 term-document matrix for this data set.

We also applied the well-known tf·idf term weighting technique. Let $\mathbf{d}_i = (w_{i1}, w_{i2}, \ldots, w_{im})^T$, where m is the total number of the unique terms. The tf·idf score of each w_{ik} can be computed by the following formula [13]:

$$w_{ik} = tf_{ik} \cdot \log\left(\frac{n}{df_k}\right) \quad , \tag{10}$$

where tf_{ik} is the term frequency of w_k in $\mathbf{d}_i$, n is the total number of documents in the corpus, and df_k is the number of documents that w_k occurs. Finally, we normalized each document vector using the L_2 norm.

For the purpose of comparison, we chose the basic PDDP algorithm as the baseline. We first tested three different

[3]The 21839×2340 term-document matrix is publicly available at `http://www-users.cs.umn.edu/~karypis/cluto/download.html`.

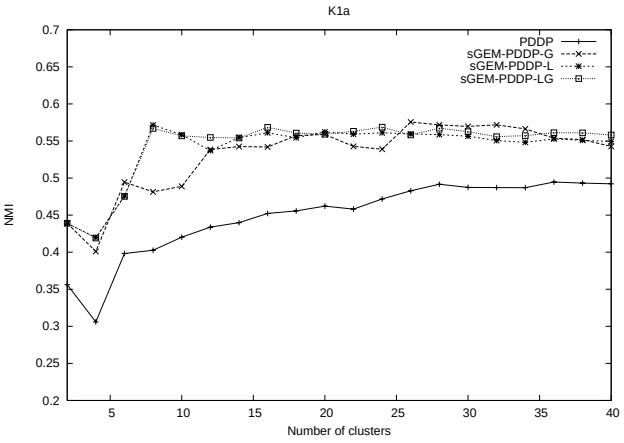

Figure 4: NMI results on the Yahoo News (K1a) data set.

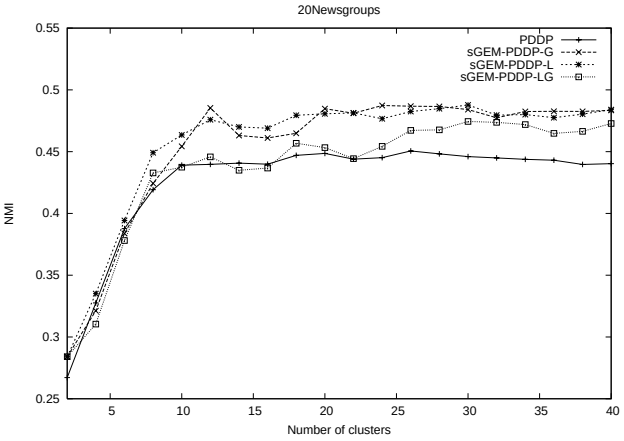

Figure 5: NMI results on the 20 Newsgroups data set.

settings of Algorithm 3, including sGEM-PDDP-L, sGEM-PDDP-G, and sGEM-PDDP-LG. sGEM-PDDP-L refers to running the sGEM algorithm locally on each PDDP splitting process, whereas sGEM-PDDP-G refers to running the sGEM algorithm globally after the PDDP algorithm converged. sGEM-PDDP-LG refers to running the sGEM algorithm with the PDDP algorithm both locally and globally. The number of clusters k is varied in the range $[2, 2k]$, and no stopping criterion was used. Then we applied both the CSV and the BIC to the above settings in order to test the estimation of the number of clusters.

5.2 Evaluation Method

Since all the documents are already categorized, we can perform evaluation by comparing clustering results with the true class labels. In our experiments, we used the normalized mutual information (NMI) [16]. In the context of document clustering, mutual information can be used as a symmetric measure for quantifying the degree of relatedness between the generated clusters and the actual categories. Particularly, when the number of clusters differs from the actual number of categories, mutual information is very useful without a bias towards smaller clusters. By

Data set	Criterion	Algorithm	k found	NMI	Time (sec.)
Yahoo News (K1a)	CSV	PDDP	15	0.447	2.79
		sGEM-PDDP-G	15	0.538	8.14
		sGEM-PDDP-L	15	0.555	4.77
		sGEM-PDDP-LG	15	0.564	11.98
	BIC	PDDP	5	0.478	2.01
		sGEM-PDDP-G	5	0.496	3.64
		sGEM-PDDP-L	12	0.589	4.78
		sGEM-PDDP-LG	12	0.571	6.34
20 Newsgroups	CSV	PDDP	34	0.443	15.838
		sGEM-PDDP-G	34	0.482	105.39
		sGEM-PDDP-L	26	0.478	50.40
		sGEM-PDDP-LG	26	0.466	219.33
	BIC	PDDP	25	0.426	14.70
		sGEM-PDDP-G	25	0.463	78.45
		sGEM-PDDP-L	28	0.473	50.58
		sGEM-PDDP-LG	28	0.476	164.74

Table 1: Clustering results by varying stoping criteria on the Yahoo News (K1a) and 20 Newsgroups data sets.

normalizing this criterion to take values between 0 and 1, the NMI can be calculated as follows:

$$NMI = \frac{\sum_{h,l} n_{h,l} \log(n \cdot n_{h,l}/n_h n_l)}{\sqrt{(\sum_h n_h \log(n_h/n))(\sum_l n_l \log(n_l/n))}}, \quad (11)$$

where n_h is the number of documents in the category h, n_l is the number of documents in the cluster l, and $n_{h,l}$ is the number of documents in the category h as well as in the cluster l. We can interpret that the NMI value is 1 when clustering results exactly match the true class labels, and close to 0 for a random partitioning [17].

5.3 Experimental Results

Figure 4 shows the clustering results on the Yahoo News (K1a) data set. The horizontal axis indicates the number of clusters, while the vertical axis indicates the NMI values. In this data set, we can see that our combined algorithms significantly outperform the basic PDDP algorithm over the entire range. Furthermore, we can observe the correlation between PDDP and sGEM-PDDP-G. This indicates that performing the global refinement with the sGEM algorithm can considerably improve the quality of the clustering results in this data set. We can also observe the correlation between sGEM-PDDP-G and sGEM-PDDP-LG. From the curves, performing the global refinement after the local refinement seems to slightly affect the quality of the clustering results. As the number of clusters increases, the results of all the combined algorithms tend to converge to similar values.

Figure 5 shows the clustering results on the 20 Newsgroups data set. In this data set, we can see that our combined algorithms perform relatively better than the basic PDDP algorithm, particularly with sGEM-PDDP-G and sGEM-PDDP-L. However, performing the global refinement after the local refinement as in sGEM-PDDP-LG degrades the quality of the clustering results. The global refinement with the sGEM algorithm leads to more decisions to move each document from its cluster to other candidate clusters. As described earlier, since some categories in this data set have similar or related topics, it has a chance that the document is reassigned to the wrong cluster.

We further discuss the results of estimating the number of underlying clusters. Note that both the data sets have the actual number of categories at 20. Table 1 summarizes clustering results by varying the stopping criteria. For the Yahoo News (K1a) data set, applying the CSV to all the combined algorithms as well as the basic PDDP algorithm gives the same result at 15 clusters. With the BIC, PDDP and sGEM-PDDP-G underestimate the number of clusters at 5, while sGEM-PDDP-L and sGEM-PDDP-LG find 12 clusters. Since the BIC is calculated locally on each splitting test, it can be affected by the local refinement more than the CSV. This case is illustrated in the below examples. For the 20 Newsgroups data set, with the CSV, PDDP and sGEM-PDDP-G find 34 clusters, while sGEM-PDDP-L and sGEM-PDDP-LG find 26 clusters. When we tested with the BIC, the results are slightly different. PDDP and sGEM-PDDP-G find 25 clusters, and sGEM-PDDP-L and sGEM-PDDP-LG find 28 clusters.

Let us focus on the Yahoo News (K1a) data set. Table 2 shows the confusion matrix generated by running sGEM-PDDP-G. Columns of the confusion matrix show the actual category labels, and rows show the predicted clusters generated by the algorithm. For evaluating a single cluster, we also calculated entropy and purity (see [15] for more details). Since the BIC measures how well the partitioning can model a given data set, without the local refinement, sGEM-PDDP-G decides to keep almost all documents of the Entertainment sub-categories in a large single cluster $\mathcal{C}_0$. Table 3 shows the confusion matrix generated by running sGEM-PDDP-LG. Although sGEM-PDDP-LG further splits the data set, a large cluster containing documents of the Entertainment sub-categories is still kept. Note that the other main topics can be partitioned into their own clusters, e.g. Health (H) in $\mathcal{C}_2$, Sports (S) in $\mathcal{C}_1$, Politics (P) in $\mathcal{C}_3$, and Business (B) in $\mathcal{C}_5$. By using the CSV, the confusion matrix is shown in Table 4. Large clusters are now partitioned into small sub-clusters, since the selection method is just based on the largest scatter value. However, a small number of documents of each Entertainment sub-categories remains in the cluster $\mathcal{C}_{13}$. We can see that both the BIC

$\mathcal{C}_j$	*Purity*	*Entropy*	H	Ep	Ec	Em	Ei	Ef	Emu	Et	Ev	Ea	Er	Eo	Emm	Ecu	Es	E	S	P	T	B
1	0.996	0.010	488	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1
4	0.903	0.139	.	.	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	2	3	56
2	0.900	0.137	1	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	54	.	4
3	0.365	0.442	1	.	2	.	2	.	1	1	.	.	.	54	.	1	.	.	.	.	52	34
0	0.176	0.840	4	247	42	21	67	278	124	185	54	24	158	11	14	73	18	9	141	58	5	47

Table 2: Confusion matrix generated by using sGEM-PDDP-G and the BIC.

$\mathcal{C}_j$	*Purity*	*Entropy*	H	Ep	Ec	Em	Ei	Ef	Emu	Et	Ev	Ea	Er	Eo	Emm	Ecu	Es	E	S	P	T	B
9	1.000	0.000	.	.	.	.	.	.	.	.	.	.	.	.	.	25	.	.	.	.	.	.
10	1.000	0.000	.	30	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
2	0.998	0.005	488	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1
1	0.978	0.036	.	.	.	.	3	.	.	.	.	.	.	.	.	.	.	.	132	.	.	.
3	0.900	0.137	1	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	54	.	4
5	0.878	0.166	.	.	.	.	5	.	.	.	.	.	.	.	.	.	.	.	.	2	5	86
7	0.865	0.184	.	4	.	.	.	45	.	1	1	1	.	.	.	.	.	.	.	.	.	.
8	0.719	0.363	.	82	.	4	.	3	12	5	.	1	.	1	.	5	.	1	.	.	.	.
11	0.718	0.308	.	1	1	.	.	.	.	28	1	.	1	7	.	.	.	.	.	.	.	.
6	0.680	0.351	.	3	6	.	8	1	.	85	21	1	.	.	.	.	.	.	.	.	.	.
4	0.425	0.372	1	.	.	.	.	.	1	.	.	.	.	48	.	.	.	.	.	.	44	19
0	0.216	0.837	4	128	37	17	54	229	112	67	31	21	157	9	14	44	18	8	9	58	11	32

Table 3: Confusion matrix generated by using sGEM-PDDP-LG and the BIC.

$\mathcal{C}_j$	*Purity*	*Entropy*	H	Ep	Ec	Em	Ei	Ef	Emu	Et	Ev	Ea	Er	Eo	Emm	Ecu	Es	E	S	P	T	B
4	1.000	0.000	122	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
7	0.995	0.010	212	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1
8	0.994	0.013	155	.	.	1	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
3	0.992	0.015	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	132	.	.	.
10	0.898	0.139	1	.	.	.	.	.	.	1	.	.	.	.	.	.	.	.	.	53	.	4
1	0.564	0.458	.	79	.	3	.	2	8	10	.	1	.	1	.	30	5	1	.	.	.	.
0	0.517	0.281	.	30	1	.	.	.	.	26	1	.	.	.	.	.	.	.	.	.	.	.
5	0.517	0.587	1	23	3	1	1	.	104	7	2	4	25	2	4	13	2	3	5	1	.	.
9	0.507	0.377	.	.	.	.	3	.	.	.	.	.	.	43	.	.	.	.	.	2	53	104
12	0.485	0.383	.	8	1	.	1	79	.	1	2	.	66	.	.	3	1	1	.	.	.	.
2	0.480	0.312	.	4	.	.	.	45	.	47	1	1	.	.	.	.	.	.	.	.	.	.
11	0.474	0.536	.	7	14	.	9	100	3	36	35	1	.	1	1	2	2	.	.	.	.	.
6	0.387	0.492	.	36	1	.	3	47	.	8	.	.	65	.	.	2	6	.	.	.	.	.
14	0.309	0.695	.	4	21	7	42	2	4	25	13	.	.	1	8	2	.	2	.	.	2	3
13	0.209	0.796	3	57	3	9	11	3	6	25	.	17	2	17	1	22	2	2	4	58	5	30

Table 4: Confusion matrix generated by using sGEM-PDDP-LG and the CSV.

and the CSV can yield a reasonable number of clusters, but the contents of clusters are generated differently due to their theoretical background concepts.

6. CONCLUSION AND FUTURE WORK

We have presented several strategies for improving the basic PDDP algorithm. When the principal direction is not representative, the corresponding hyperplane tends to produce individual clusters with wrongly partitioned contents. By formulating the problem with the finite mixture model, we describe the sGEM algorithm that can be combined with the PDDP algorithm in several ways for refining the partitioning results. Preliminarily experimental results on two different document sets are very encouraging.

In future work, we intend to investigate other model selection techniques for approximating the number of underlying clusters. Recently, work by [7] has demonstrated that estimating the number of clusters in the k-means algorithm using the Anderson-Darling test yields very promising results, and seems to outperform the BIC. We think that this statistical measure can also be applied to our proposed algorithm.

7. REFERENCES

[1] Boley, D. Principal direction divisive partitioning. *Data Mining and Knowledge Discovery*, 2(4):325–344, 1998.

[2] Boley, D., and Borst, V. Unsupervised clustering: A

fast scalable method for large datasets. *CSE Report TR-99-029, University of Minnesota*, 1999.

[3] Bradley, P. S., and Fayyad, U. M. Refining initial points for k-means clustering. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 91–99, 1998.

[4] Chickering, D., Heckerman, D., and Meek, C. A bayesian approach to learning bayesian networks with local structure. In *Proceedings of the thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 80–89. Morgan Kaufmann, 1997.

[5] Dasgupta, S., and Schulman, L. J. A two-round variant of em for gaussian mixtures. *Sixteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, 2000.

[6] Golub, G., and Loan, C. V. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, 1989.

[7] Hamerly, G., and Elkan, C. Learning the k in k-means. In *Proceedings of the seventeenth annual conference on neural information processing systems (NIPS)*, December 2003.

[8] He, J., Tan, A.-H. , Tan, C.-L., and Sung, S.-Y. *On Quantitative Evaluation of Clustering Systems*. In W. Wu and H. Xiong, editors, Information Retrieval and Clustering. Kluwer Academic Publishers, 2003.

[9] Kass, R. E., and Raftery, A. E. Bayes factors. *Journal of the American Statistical Association*, 90:773–795, 1995.

[10] Lang, K. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[11] McCallum, A. K. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`.

[12] Rasmussen, E. *Clustering algorithms*. In W. Frakes and R. Baeza-Yates, editors,Information retrieval: data structures and algorithms. Prentice-Hall, 1992.

[13] Salton, G., and Buckley, C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management: an International Journal*, 24(5):513–523, 1988.

[14] Steinbach, M., Karypis, G., and Kumar, V. A comparison of document clustering techniques. *KDD Workshop on Text Mining*, 1999.

[15] Strehl, A., Ghosh, J., and Mooney, R. J. Impact of similarity measures on web-page clustering. In *Proceedings of AAAI Workshop on AI for Web Search*, pages 58–64, 2000.

[16] Strehl, A., and Ghosh, J. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Journal on Machine Learning Research*, 3:583–617, 2002.

[17] Zhong, S., and Ghosh, J. A comparative study of generative models for document clustering. *SDM Workshop on Clustering High Dimensional Data and Its Applications*, 2003.