Scalable, Balanced Model-based Clustering*

Shi Zhong† and Joydeep Ghosh†

Abstract

This paper presents a general framework for adapting any generative (model-based) clustering algorithm to provide balanced solutions, i.e., clusters of comparable sizes. Partitional, model-based clustering algorithms are viewed as an iterative two-step optimization process—iterative model re-estimation and sample re-assignment. Instead of a maximum-likelihood (ML) assignment, a balance-constrained approach is used for the sample assignment step. An efficient iterative bipartitioning heuristic is developed to reduce the computational complexity of this step and make the balanced sample assignment algorithm scalable to large datasets. We demonstrate the superiority of this approach to regular ML clustering on complex data such as arbitrary-shape 2-D spatial data, high-dimensional text documents, and EEG time series.

Keywords: model-based clustering, scalable algorithms, balanced clustering, constrained clustering

1 Introduction

Clustering or segmentation of data is a fundamental data analysis step that has been widely studied across multiple disciplines for over 40 years [15]. Current clustering methods can be divided into generative (model-based) approaches [29, 6, 27] and discriminative (similarity-based) approaches [36, 28, 14]. Parametric, model-based approaches attempt to learn generative models from the data, with each model corresponding to one particular cluster. In similarity-based approaches, one determines a distance or similarity function between pairs of data samples, and then groups similar samples together into clusters. Unfortunately, calculating the similarities between all pairs of data samples is computationally inefficient, requiring $O(n^2)$ time. So sampling techniques have been typically employed to scale such methods to large datasets [13, 4]. In contrast, several model-based partitional approaches have a complexity of $O(kn)$, where k is the number of clusters, and are thus more scalable.

Several real life data mining applications demand comparably sized segments of the data, irrespective of whether the natural clusters in the data are of comparable sizes or not [12]. For example, a direct marketing campaign often starts with segmenting customers into groups of roughly equal size or equal estimated revenue generation, (based on market basket analysis, or purchasing behavior at a web site), so that the same number of sales teams, marketing dollars etc., can be allocated to each segment. In large retail chains, one often desires product categories/groupings of comparable importance, since subsequent decisions such as shelf/floor space allocation and product placement are influenced by the objective of allocating resources proportional to revenue or gross margins associated with the product groups [33]. Similarly, in clustering of a large corpus of documents to generate topic hierarchies, balancing greatly facilitates navigation by avoiding the generation of hierarchies that are highly skewed, with uneven depth in different parts of the "tree" hierarchy or having widely varying number of documents at the leaf nodes.

In addition to application requirements, balanced clustering is sometimes also helpful because it tends to decrease sensitivity to initialization and to avoid outlier clusters (highly under-utilized representatives) from forming, and thus has a beneficial regularizing effect. In fact, balance is also an important constraint for spectral graph partitioning algorithms [8, 17, 24], which could give completely useless results if the objective function is just the minimum cut instead of a modified minimum cut that favors balanced clusters.

Unfortunately, k-means type algorithms (including the soft EM variant [6], and BIRCH [37]) are increasingly prone to yielding imbalanced solutions as the input dimensionality increases. This problem is exacerbated when a large (tens or more) number of clusters are needed, and it is well known that both hard and soft k-means invariably result in some near-empty clusters in such scenarios [4, 7].

While not an explicit goal in most clustering formulations, certain approaches such as top-down bisecting k-means [30] tend to give more balanced solutions than others such as single-link agglomerative clustering. The most notable type of clustering algorithms in terms of balanced solutions is graph partitioning [20], but it

*Supported in part by an IBM Faculty Partnership Award from IBM/Tivoli and IBM ACAS.

†Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX 78712

needs $O(n^2)$ computation just to compute the similarity matrix. Certain online approaches such as frequency sensitive competitive learning [1] can also be employed for improving balancing. A generative model based on a mixture of von Mises-Fisher distributions has been developed to characterize such approaches for normalized data [3].

The problem of clustering large scale data under constraints such as balancing has recently received attention in the data mining literature [31, 7, 35, 4]. Since balancing is a global property, it is difficult to obtain near-linear time techniques to achieve this goal while retaining high cluster quality. Banerjee and Ghosh [4] proposed a three-step framework for balanced clustering: sampling, balanced clustering of the sampled data, and populating the clusters with the remaining data points in a balanced fashion. This algorithm has relatively low complexity of $O(n \log(n))$ but relies on the assumption that the data itself is very balanced (for the sampling step to work). Bradley, Bennett and Demiriz [7] developed a constrained version of the k-means algorithm. They constrained each cluster to be assigned at least a minimum number of data samples at each iteration. The cluster assignment subproblem was formulated as a minimum cost flow problem, which has a high ($O(n^3)$) complexity.

In this paper, we take a balance-constrained approach built upon the framework of probabilistic, model-based clustering [40]. Model based clustering is very general, and can be used to cluster a wide variety of data types, from vector data to variable length sequences of symbols or numbers [29]. First, a unifying bipartite graph view is presented for model-based clustering. Then a two-step iterative maximum-likelihood (ML) optimization process is presented and analyzed for hard, model-based clustering. The two steps are model re-estimation step and sample re-assignment step, respectively. We formulate a completely balanced sample assignment subproblem, solved using a greedy heuristic at each iteration of the process. The heuristic obtains an approximate solution to the subproblem in $O(kn(k + \log n))$ time. It can be easily generalized to handle partially balanced assignments and specific percentage assignment problems. Finally, we do a post-processing step of ML clustering in situations where strict balancing is not required.

There are several motivations behind our approach. First, probabilistic model-based clustering provides a principled and general approach to clustering [5]. For example, the number of clusters may be estimated using Bayesian model selection, though this is not done in this paper. Second, the two-step view of partitional clustering is natural and has been discussed by Kalton, Wagstaff, and Yoo [16]. Finally, the post-processing refinement is motivated by the observation in [7] that if the balancing constraint is too strict, then often distant data points are forceably grouped together leading to substantial degradation in cluster quality. We observed than often a slight decrease in the amount of balance can buy substantial improvements in cluster quality, a favorable trade-off for many applications.

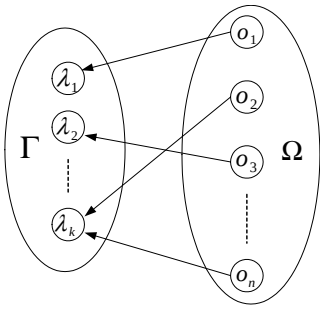

Figure 1: A bipartite graph view of model-based clustering.

The organization of this paper is as follows. Section 2 presents a general model-based clustering framework, and in particular, analyzes partitional model-based clustering algorithms. Section 3 presents our balanced clustering algorithm. Section 4 experimentally investigates the performance of the proposed balanced approach on synthetic, real text and time-series data. Finally, section 5 concludes this paper.

2 Model-based clustering

2.1 A unifying bipartite graph view. In this section, we introduce a unified framework for model-based clustering algorithms, based on a bipartite graph view (Fig. 1) of a data set, O, containing n data samples, $\{o_1, o_2, ..., o_n\}$, and k clusters, represented by k models $\lambda_1, \lambda_2, ..., \lambda_k$, respectively. In this paper, we also refer to λ_j as the set of parameters associated with the j-th model and Λ as the set of parameters for all models (including the model mixture weights introduced in the next section).

In this view, we have data sample vertices in the sample space Ω, probabilistic generative model vertices in the model space Γ, and connections between the two spaces. Each cluster is represented by a model in Γ, which usually contains models from a specific family. The model λ_j can be viewed as a generalized center of cluster j. A connection between sample o_i and model λ_j indicates that the sample o_i is being associated with cluster j. Note that a data sample is not necessarily assigned to only one cluster even though it is so for hard clustering. The connection weight w_{ij} is $\log P(o_i|\lambda_j)$,

which measures the closeness of sample o_i to model λ_j (or the "center" of cluster j). We can define the distance between two models λ_j and $\lambda_{j'}$ (i.e. "centers" of cluster j and j') as the (empirical) KL divergence

$$D(\lambda_j, \lambda_{j'}) = \frac{1}{|O_j|} \sum_{o \in O_j} \left(\log P(o|\lambda_j) - \log P(o|\lambda_{j'}) \right) ,$$

where O_j is the set of samples partitioned into cluster j. One can make the distance measure symmetric by defining $D_s(\lambda_j, \lambda_{j'}) = \frac{1}{2} \left(D(\lambda_j, \lambda_{j'}) + D(\lambda_{j'}, \lambda_j) \right)$. Note that this definition of inter-cluster distance is just one example of many possible designs and can be modified for specific applications.

Having defined a closeness measure between data samples and models, and a distance measure between pair of models, generic model-based clustering algorithms can be readily applied. The closeness measure is used for assigning samples to clusters in partitional clustering and the distance measure for finding the closest pair of clusters to merge in hierarchical agglomerative clustering. Readers are referred to [40] for a detailed description. The maximum-likelihood method is often used for training a model λ given a set of data samples O, to maximize $P(O|\lambda)$, or equivalently, $\log P(O|\lambda)$. A model-based clustering algorithm usually iteratively partitions the data samples and trains the models based on the partitioning, to maximize $P(O|\Lambda)$. For this reason, we often use maximum-likelihood clustering, or ML clustering, to refer to model-based clustering in this paper.

2.2 Model-based partitional clustering. The model-based k-means algorithm (Fig. 2) is a generalized version of the standard k-means. Basically, the algorithm iterates between a model re-estimation step 2a and a sample re-assignment step 2b. The ML assignment is used for the latter step. It can be further extended to soft EM clustering, where each sample o gets fractionally assigned to a cluster j according to the posterior probability $P(j|o, \Lambda)$, and each model is trained using the posterior probability weighted samples. An information-theoretic analysis of these two assignment strategies has been given in [21].

Let us analyze these two algorithms from the perspective of objective function and explain why they usually perform very similarly in practice. The general log-likelihood objective function to be maximized for model-based partitional clustering can be written as

$$\log P(O|\Lambda) = \sum_i \log \left(\sum_j \alpha_{ij} P(o_i|\lambda_j) \right), \tag{2.1}$$

where $\Lambda = \{\lambda_j, \alpha_{ij}\}_{i=1,...,n, j=1,...,k}$ and α_{ij}'s are the model mixture weights with constraints $\sum_j \alpha_{ij} = 1, \ \forall i$. Applying EM algorithm to maximize (2.1), one can derive the general re-estimation formula for Λ as follows:

Algorithm: mk-means

Input: Data samples $O = \{o_1, ..., o_n\}$, and model structure $\Lambda = \{\lambda_1, ..., \lambda_k\}$.

Output: Trained models Λ and a partition of the data samples given by the cluster identity vector $Y = \{y_1, ...y_n\}, \ y_i \in \{1, ..., k\}$.

Steps:

1. Initialization: initialize the model parameters Λ and cluster identity vector Y;

2a. Model re-estimation: for each cluster j, let $O_j = \{o_i | y_i = j\}$, the parameters of each model λ_j is re-estimated as $\lambda_j = \max_{\lambda} \sum_{o \in O_j} \log P(o|\lambda)$;

2b. Sample re-assignment: for each data sample i, set $y_i = \arg\max_j \log P(o_i|\lambda_j)$;

3. Stop if Y does not change or $\sum_i \log P(o_i|\lambda_{y_i})$ converges, otherwise go back to Step 2a.

Figure 2: Model-based k-means algorithm.

$$\lambda_j^{(new)} = \arg\max_{\lambda} \sum_i P(j|o_i, \Lambda) \log P(o_i|\lambda) , \tag{2.2}$$

$$\alpha_{ij}^{(new)} = P(j|o_i, \Lambda) , \tag{2.3}$$

$$P(j|o_i, \Lambda) = \frac{\alpha_{ij} P(o_i|\lambda_j)}{\sum_{j'} \alpha_{ij'} P(o_i|\lambda_{j'})} . \tag{2.4}$$

Special choices of α_{ij}'s lead to several commonly used algorithms. For example, setting $\alpha_{ij} = I(y_i = j)$[1], where the cluster identity $y_i = \arg\max_{j'} \log P(o_i|\lambda_{j'})$, leads to the mk-means algorithm. Constraining α_{ij}'s to be independent of individual data samples, i.e $\alpha_{ij} = \alpha_j, \forall i$, results in the EM clustering algorithm. In this case, the re-estimation of α's (2.3) needs to be modified to $\alpha_j^{(new)} = \frac{1}{n} \sum_i P(j|o_i, \Lambda)$. Further enforcing that $\alpha_j = 1/k, \forall j$ leads to a more constrained version of soft clustering [40]. Note that the general formula (2.2) for re-estimating models need not to be modified no matter how we set the α's.

In practice, we often have the condition $P(o_i|\lambda_{y_i}) \gg P(o_i|\lambda_j), \forall j \neq y_i$ (especially for complex models such as HMMs), which means that $P(j|o_i, \Lambda)$ will be dominated by the likelihood values and be very close to 1 for $j = y_i$, and 0 otherwise, independent of most choices

[1] $I(.)$ is an indicator function that takes value 1 when the predicate argument is true and 0 otherwise.

of α's. This suggests that the difference between hard and soft versions is small, i.e. their clustering results will be fairly similar. As the mk-means has better time complexity than EM ($O(kn)$ vs. $O(k^2n)$), we focus on the analysis of mk-means algorithm from this point onwards.

The objective function of the mk-means algorithm can be simplified as

$$\log P(O|\Lambda) = \sum_i \log P(o_i|\lambda_{y_i}). \tag{2.5}$$

Let z_{ij} be a binary assignment variable with value 1 indicating assignment of sample i to model j. The mk-means clustering can be re-formulated as the following optimization problem

$$\begin{array}{ll} \max\limits_{\lambda,z} & \sum\limits_{i,j} z_{ij} \log P(o_i|\lambda_j) \\ \text{s.t.} & \sum_j z_{ij} = 1, \forall i;\ z_{ij} \in \{0,1\}, \forall i,j. \end{array} \tag{2.6}$$

Note that the step 2a in Fig. 2 corresponds to training λ_j's to solve (2.6) with z_{ij}'s fixed, and the step 2b finding z_{ij}'s to maximize (2.6) with λ_j's fixed. This observation leads to a decomposition of (2.6) into two subproblems: a model estimation subproblem (2.7) and a sample assignment subproblem (2.8).

$$\max_{\lambda} \sum_{i,j} z_{ij} \log P(o_i|\lambda_j). \tag{2.7}$$

$$\begin{array}{ll} \max\limits_{z} & \sum\limits_{i,j} z_{ij} \log P(o_i|\lambda_j) \\ \text{s.t.} & \sum_j z_{ij} = 1, \forall i;\ z_{ij} \in \{0,1\}, \forall i,j. \end{array} \tag{2.8}$$

As we will see in the next section, this decomposition greatly facilitates the development and analysis of a balanced mk-means algorithm. Also note the formulation is generic in that we can plug in any probabilistic models into the subproblem (2.7) to get an instantiated clustering algorithm. The convergence of the generic mk-means algorithm is given by the following theorem.

THEOREM 2.1. *If $P(o|\lambda)$ is bounded from above, the mk-means algorithm given in Fig. 2 will converge to a local maximum of the objective function in (2.5).*

Proof: It is easy to verify that both step 2a and 2b in Fig. 2 will not decrease the objective function (2.5). The objective function is upper-bounded since $P(o|\lambda)$ is bounded from above. These two conditions complete the convergence proof. □

3 Balanced model-based clustering

3.1 Completely balanced mk-means clustering.

The completely balanced mk-means clustering problem can be defined as:

$$\begin{array}{ll} \max\limits_{\lambda,z} & \sum\limits_{i,j} z_{ij} \log P(o_i|\lambda_j) \\ \text{s.t.} & \sum_j z_{ij} = 1, \forall i;\ \sum_i z_{ij} = n/k, \forall j; \\ & z_{ij} \in \{0,1\}, \forall i,j. \end{array} \tag{3.9}$$

If n/k is not an integer, we round it to the closest integer and make slight changes so that $\sum_{i,j} z_{ij} = n$ holds. This problem is again decomposed into two subproblems: the model estimation subproblem is the same as in (2.7), whereas the balanced sample assignment subproblem becomes

$$\begin{array}{ll} \max\limits_{z} & \sum\limits_{i,j} z_{ij} \log P(o_i|\lambda_j) \\ \text{s.t.} & \sum_j z_{ij} = 1, \forall i;\ \sum_i z_{ij} = n/k, \forall j; \\ & z_{ij} \in \{0,1\}, \forall i,j. \end{array} \tag{3.10}$$

Fig. 3 describes a generic completely balanced mk-means algorithm that solves the problem (3.9). Its convergence is given by the following theorem.

Algorithm: completely balanced mk-means

Input: Data samples $O = \{o_1, ..., o_n\}$ and model structure $\Lambda = \{\lambda_1, ..., \lambda_k\}$.

Output: Trained models Λ and a partition of data given by cluster identity vector $Y = \{y_1, ...y_n\}$, $y_i \in \{1, ..., k\}$.

Steps:

1. Initialization: initialize model parameters Λ, cluster identity vectors Y and $z_{ij} = I(y_i = j)$;
2a. Model re-estimation: let λ_j's be the solution to problem (2.7);
2b. Sample re-assignment: let z be the solution to problem (3.10) and set $y_i = \max_j z_{ij}$;
3. Stop if Y do not change or $\sum_i \log P(o_i|\lambda_{y_i})$ converges, otherwise go back to Step 2a.

Figure 3: Completely balanced mk-means algorithm.

THEOREM 3.1. *If $P(o|\lambda)$ is bounded from above, the algorithm given in Fig. 3 will converge to a local maximum of the objective function in (3.9).*

Proof of this theorem is straightforward and similar to the proof of Theorem 2.1.

Note that the generic algorithm does not specify how to solve the balanced sample assignment subproblem (3.10). It is an integer programming problem, which is NP-hard in general. Fortunately, this integer programming problem is special in that it has the same optimum as its corresponding real relaxation [7], which

Algorithm: iterative greedy bipartitioning

Input: Log-likelihood matrix $w_{ij} = \log P(o_i|\lambda_j), i = 1, ..., n,\ j = 1, ..., k$.

Output: A partition matrix z that satisfies $z_{ij} \in \{0, 1\}$, $\sum_j z_{ij} = 1, \forall i$ and $\sum_i z_{ij} = \frac{n}{k}, \forall j$.

Steps:

1. Initialization: set $j' = 1$, $J = \{1, ..., n\}$ and $z_{ij} = 0, \forall i, j$;
2. Calculating log-likelihood difference vector: let $dv_i = w_{ij'} - \max_{j>j'} w_{ij}, \forall i \in J$ and $dv = \{dv_i\}_{i \in J}$;
3. Bipartitioning using sorted difference vector: sort dv in descending order and assign the top $\frac{n}{k}$ samples to cluster j', i.e. set $z_{ij'} = 1, \forall i \in I$, where I is the set of indices corresponding to the top $\frac{n}{k}$ samples;
4. Stop if $j' = k$, otherwise let $j' = j' + 1$, $J = J - I$ and go back to Step 2.

Figure 4: Iterative greedy bipartitioning algorithm.

is a linear programming problem. The relaxed optimization problem is

$$\begin{array}{ll} \max\limits_{z} & \sum\limits_{i,j} z_{ij} \log P(o_i|\lambda_j) \\ \text{s.t.} & \sum_j z_{ij} = 1, \forall i;\ \sum_i z_{ij} = n/k, \forall j; \\ & z_{ij} \geq 0, \forall i, j. \end{array} \tag{3.11}$$

The best known exact algorithm to solve this problem is an improved interior point method that has a complexity of $O(n^3k^3/\log(nk))$, according to [2]. The famous simplex algorithm has exponential worst case time complexity but often exhibits polynomial expected time complexity [10, pp.96], which is around $O(n^3k)$ in our case.

To make our clustering algorithm scalable, we seek approximate solutions to (3.10) that can be obtained in time better than $O(n^2)$. There are a number of heuristics that can be used, such as the one-to-many stable matching algorithm, used by Banerjee and Ghosh [4] to populate balanced clusters (learned from a small sampled set of data points).An alternative is an iterative greedy bipartitioning algorithm (Fig. 4) that assigns n/k data samples to one of the k clusters in a locally optimal fashion at each iteration. Both heuristics work well in our preliminary experiments, but we choose the second one since it fits well within the optimization problem setting in this paper. The completely balanced clustering algorithms using this greedy heuristic always converge in our experiments, though not monotonically.

The motivation behind this heuristic is that it solves (3.10) exactly for $k = 2$. In other words, if there are just two clusters, we simply sort the difference vector $dv_i = \log P(o_i|\lambda_1) - \log P(o_i|\lambda_2), i = 1, ..., n$ in descending order and assign the first $n/2$ samples to cluster 1 and the second half to cluster 2. It is easy to show this gives a $\{\frac{n}{2}, \frac{n}{2}\}$ bipartition that maximizes (2.5).

For $k > 2$, we conduct a greedy bipartition at each iteration that separates the data samples for one cluster from all the others in such a way that the objective (2.5) is locally maximized. It is trivial to show that the j-th iteration of the algorithm in Fig. 4 gives a locally optimal $\{\frac{n}{k}, \frac{(k-j)n}{k}\}$ bipartition that assigns $\frac{n}{k}$ samples to the j-th cluster.

We now look at the time complexity of this algorithm. Let $n_j = \frac{(k+1-j)n}{k}$ be the length of the difference vector computed at the j-th iteration. Calculating the difference vectors takes $\sum_j n_j(k-j) \simeq O(k^2n)$ time and sorting them takes $\sum_j n_j \log n_j \simeq O(kn \log n)$ time. The total time complexity for this algorithm is $O(kn(k + \log n))$. The greedy nature of the algorithm stems from the imposition of an arbitrary ordering of the clusters using j'. So one should investigate the effect of different orderings. In our experiments, the ordering is done at random in each experiment, multiple experiments are run and the variation in results is inspected. The results exhibit no abnormally large variations and suggest that the effect of ordering is small.

The algorithm can be easily generalized to solve the sample assignment problem with specific balance constraints. For example, if we have some prior knowledge about the partition percentages (e.g. a $\{20\%, 20\%, 30\%, 30\%\}$ partition), we can easily build the numbers into the algorithm and assign a corresponding number of data samples to each cluster. Or if we just want each cluster to have at least $m(< \frac{n}{k})$ samples, i.e. similar to [7], we can assign just the top m samples at each iteration and use ML assignment for the remaining data. This variation may be useful in situations where "near" balanced clustering is desired, but is not investigated in this paper.

3.2 Refinement step. This step is really an optional step, depending on what kind of results are desired. If approximate rather than exact balanced solutions are acceptable, then a minor refinement can be used to improve cluster quality. This is achieved by letting the results from completely balanced mk-means serve as an initialization for the regular mk-means. Since the regular mk-means has relative low complexity of $O(kn)$, this extra overhead is low. The experiments reported in this paper reflect a "full" refinement in the sense that we run the regular mk-means in the refinement step until convergence. Alternatively, partial refinement

such as one round of ML re-assignment can be used and is expected to give an intermediate result between the completely balanced one and the "fully" refined one. In the experimental results, intermediate results are not shown but they will be bounded from both sides by the completely balanced and the "fully" refined results.

3.3 Clustering models used in our experiments.

In this section, we briefly introduce the four generative models used in our experiments.

K-means

For spherical Gaussian models, we have $\lambda = \{\mu, \sigma\}$ and $P(o|\lambda) = \frac{1}{Z(\sigma)} \exp - \frac{\|o-\mu\|^2}{2\sigma^2}$, where $Z(\sigma)$ is a normalization term. For k-means clustering, σ is the same for all clusters, and the normalized log-likelihood (NLL) objective is simply:

$$NLL_{k-means} = -\frac{1}{n}\sum_{i=1}^{n} \|o_i - \mu_{y_i}\|^2. \tag{3.12}$$

K-vMFs

Euclidean distance is not appropriate for clustering high dimensional normalized data such as text [34]. A better metric used for text clustering is the cosine similarity, which can be derived from directional statistics — the von Mises-Fisher distribution [3]. The general vMF distribution can be written as

$$P(o|\lambda) = \frac{1}{Z(\kappa)} \exp\left(\kappa \frac{o^T \mu}{\|\mu\|}\right) , \tag{3.13}$$

where o here is a unit-length document vector (in L_2 norm) and the Bessel function $Z(\kappa)$ is a normalization term. Similar to the k-means algorithm, we assume the directional variance (dispersion) κ is the same for all clusters. With some manipulation, we get $\log P(o|\lambda) \propto \frac{o^T \mu}{\|\mu\|} + C$, where C is a constant related to κ. The generalized k-means using this simplified vMF model has been named spherical k-means [9] and successfully used for document clustering. The model estimation for k-vMFs amounts to $\mu_j = \frac{1}{|O_j|}\sum_{o_i \in O_j} o_i, j = 1, ..., k$. The objective function evaluated for this model in our experiments is

$$NLL_{k-vMFs} = \frac{1}{n}\sum_{i} \frac{o_i^T \mu_{y_i}}{\|\mu_{y_i}\|} , \tag{3.14}$$

which is in fact the average cosine similarity of all data samples and their cluster means.

K-multinomials

Multinomial models have been popular for text classification (with naïve Bayes assumption) and can be adapted for clustering.A multinomial model for cluster j represents a document d_i by a multinomial distribution of the words in the document

$$P(d_i|\lambda_j) = P(c_j) \prod_{l} P(w_l|c_j)^{n_{l,i}} , \tag{3.15}$$

where $P(c_j)$ is the prior probability for cluster j, $P(w_l|c_j)$ the probability of word w_l occurring in the cluster j, and $n_{l,i}$ the number of times word w_l occurs in document d_i. The normalized log-likelihood (NLL) evaluated for this model in our experiments is

$$NLL_{k-multinomials} = \frac{1}{n}\sum_{i}\left(\log P(c_{y_i}) + \sum_{l} n_{l,i} \log P(w_l|c_{y_i})\right). \tag{3.16}$$

K-HMMs

Hidden Markov models have long been used for modeling sequences [26] and the k-HMMs algorithm has been used by several authors to cluster time sequences [29, 22]. The NLL objective used for the k-HMMs algorithm is

$$NLL_{k-HMMs} = \frac{1}{n}\sum_{i} \frac{1}{T_i} \log P(o_i|\lambda_{y_i}) , \tag{3.17}$$

where T_i is the length of sequence o_i. Space prohibits a detailed introduction to HMMs, which can be found in [26].

For each of the above models, we compare three versions of the clustering algorithm in our experiments: a regular version, a balanced version and a balanced+refinement version. For simplicity, the four balanced algorithms are named bk-means, bk-vMFs, bk-multinomials and bk-HMMs, respectively, and the refined ones are called refined bk-means, refined bk-vMFs, ..., etc. As a reminder, the balanced version generates completely balanced clustering and the refined version attaches a post-processing phase to the balanced version. Finally, we emphasize that our balanced approach is built on the general model-based clustering framework and applies to any application for which appropriate models exist, for the balance constraint is completely independent of the model re-estimation part.

4 Clustering results and discussions

4.1 Clustering evaluation.

To evaluate the performance of our balanced clustering algorithms, we compare them with the regular ML clustering in terms of balance, objective value and mutual information between cluster labels and class labels, if they exist. We measure the balance of a clustering by normalized entropy that is defined as

$$N_{entro} = -\frac{1}{\log k}\sum_{j=1}^{k} \frac{n_j}{n} \log\left(\frac{n_j}{n}\right) , \tag{4.18}$$

where n_j is the number of data samples in cluster j. A normalized entropy of 1 means perfectly balanced clustering and 0 extremely unbalanced clustering. We use the normalized objectives in (3.12), (3.14), (3.16), and (3.17) for k-means, k-vMFs, k-multinomials and k-HMMs, respectively.

For datasets that come with original class labels, we also evaluate the quality of clustering using the normalized mutual information [34], which is defined as

$$NMI = \frac{\sum_{h,l} n_{h,l} \log\left(\frac{n \cdot n_{h,l}}{n_h n_l}\right)}{\sqrt{\left(\sum_h n_h \log\frac{n_h}{n}\right)\left(\sum_l n_l \log\frac{n_l}{n}\right)}}, \tag{4.19}$$

where n_h is the number of data samples in class h, n_l the number of samples in cluster l and $n_{h,l}$ the number of samples in class h as well as in cluster l. The NMI value is 1 for a perfect clustering and close to 0 for a random partitioning. This is a better measure than purity or entropy which are both biased towards high k solutions. For a more detailed discussion on the NMI measure, see [34, 32].

4.2 Results on synthetic data. We first tested the balanced k-means algorithm on a synthetic but difficult dataset—the t4 dataset (Fig. 5(c)) included in the CLUTO toolkit [18]. There are no ground truth labels for this dataset but there are six natural clusters plus a lot of noise according to human judgment. The best algorithm that can identify all the six natural clusters uses a hybrid partitional-hierarchical approach [19, 18]. It partitions the data into a large number of clusters and then merges them back to a proper granularity level.

Fig. 5(a) shows the balance results and Fig. 5(b) the NLL results, for different number of clusters. All the results are averaged over ten experiments with the bars at each data point indicating ± 1 standard deviation in ten experiments. The balance performance of regular k-means deteriorates significantly as the number of clusters increases. The bk-means algorithm always delivers perfectly balanced clusterings but pays in terms of NLL. In contrast, the refined bk-means approach achieves very balanced clusterings as well as dramatically better objectives than the regular k-means. In this case, the refinement step seems to be able to keep the balance of the clusterings from bk-means and improve the NLL objective simultaneously.

Fig. 5(c)&(d) show a typical clustering result for $k = 30$. It can be seen that the regular k-means produces many empty clusters and ones that mix data points from different natural clusters, whereas the refined bk-means gives much cleaner and more balanced results. Fig. 5(e)&(f) show the histogram distribution of cluster sizes for the results in Fig. 5(c)&(d). The regular k-means produces many empty clusters and large variations in cluster sizes, whereas the refined bk-means gives much more balanced clusters. In addition, the refined bk-means achieves a much better local solution (with an NLL of -620.9 vs. -1237.1 for the regular k-means).

4.3 Results on real text data. We used the 20-newsgroups (NG20)[2], mini 20-newsgroups (mini20), and the CLASSIC datasets for empirical performance analysis on text data. The NG20 dataset is a collection of 20,000 messages, collected from 20 different usenet newsgroups, 1,000 messages from each. We preprocessed the raw dataset using the Bow toolkit [23], including chopping off headers and removing stop words as well as words that occur in less than three documents. In the resulting dataset, each document is represented by a 43,586-dimensional sparse vector and there are a total of 19,949 documents (with empty documents removed, still around 1,000 per category). The mini20 dataset is a random sample from NG20, with 100 messages from each category. After the same preprocessing step, the resulting dataset consists of 1,998 documents in 10,633 dimensional vector space. This dataset has been used by Nigam [25] for text classification and is included in this paper to evaluate the performance of different models on small document collections. Note that both NG20 and mini20 datasets contain 20 completely balanced clusters.

The CLASSIC dataset is contained in the datasets for the CLUTO software package [18] and has been used in [38]. It was obtained by combining the CACM, CISI, CRANFIELD, and MEDLINE abstracts that were used in the past to evaluate various information retrieval systems[3]. CACM consists of 3,203 abstracts from computer systems papers, CISI consists of 1,460 abstracts from information retrieval papers, MEDLINE consists of 1,033 abstracts from medical journals, and CRANFIELD consists of 1,398 abstracts from aeronautical systems papers. The CLASSIC dataset is already preprocessed in CLUTO package and represents each document as a 41,681-dimensional sparse vector. The clusters in this dataset are slightly unbalanced, with a normalized entropy of 0.92.

Two types of models, vMFs and multinomials, have been used for clustering experiments on these three datasets. For vMF models, we use log(IDF) weighted (and normalized) document vectors since the weighting seems to substantially improve clustering results.

[2] Available from http://kdd.ics.uci.edu/databases/ 20newsgroups/20newsgroups.html.

[3] Available from ftp://ftp.cs.cornell.edu/pub/smart.

Fig. 6, 7 and 8 show the results on the NG20, mini20 and CLASSIC datasets, respectively, with results for multinomial models on the left column and those for vMF models on the right. The first row shows balance results (normalized entropy), the second row NLL objectives and the last row NMI values. All results are shown as *average* $\pm$ 1 *standard deviation* over 20 experiments.

In all cases, the balanced clustering algorithms, with refinement, perform either comparably or significantly better than regular ML clustering in terms of both NLL and NMI, and provide significantly more balanced clusterings than the regular methods.

Comparing multinomial models with vMF ones, we see that the vMF-based algorithms produce much more balanced clusterings for all datasets, no matter whether the original dataset is highly balanced or not. In terms of NMI, k-multinomials and k-vMFs work comparably for large collections (NG20 and CLASSIC, 1,000+ documents per cluster) and k-multinomials tends to be better for large number of clusters. For small collections (mini20, 100 documents per cluster), the vMF-based algorithms perform significantly better than multinomial ones. We suspect the reason is that, the parameter estimation for multinomial models is more sensitive to the data size than that for vMF models. More experiments are needed to validate this argument.

4.4 Results on EEG time-series. The EEG data from the UCI KDD archive (http://kdd.ics.uci.edu) arose from a large study to examine EEG correlates of genetic predisposition to alcoholism. There are two groups of subjects in the study: alcoholic and control. The data contains measurements, sampled at 256 Hz for 1 second, from 64 electrodes placed on the scalp. We extracted from the archive a subset called EEG-2 [39], that contains 20 measurements for two subjects—one alcoholic and one control, for each type of three stimuli types, from 2 electrodes (F4, P8). As a result, the EEG-2 dataset contains 120 data samples and each sample is a pair of sequences of length 256.

When modeling EEG time series, we set the dimension of observation vector to be the number of EEG channels, and use an HMM with bivariate Gaussian observations and five hidden states. We evaluate the performance of the regular k-HMMs, bk-HMMs and refined bk-HMMs on the EEG-2 dataset with 6 clusters. The results are shown in Table 4.4. Similar to what we have seen from previous experiments, bk-HMMs generates lower quality clusters than the regular k-HMMs. But the refined bk-HMMs significantly outperforms the regular k-HMMs in terms of both balance and NLL measures, based on t-tests at $p = 0.05$ level. In addition, it leads to slightly better NMI values.

Table 1: Clustering results on EEG-2 with 6 clusters

	N_{entro}	NLL	NMI
regular k-HMMs	0.90 ± 0.05	-113.4 ± 1.0	0.32 ± 0.03
bk-HMMs	1.0 ± 0	-114.2 ± 1.6	0.31 ± 0.04
refined bk-HMMs	0.97 ± 0.02	-112.4 ± 0.6	0.33 ± 0.03

5 Concluding remarks

We have presented a general framework for scalable, balanced model-based clustering that comes from an analysis of the two-step optimization process embedded in any model-based partitional clustering algorithm. The balanced approach is applicable to any domain for which good probabilistic models exist. We have used an efficient greedy heuristic to solve the balanced sample assignment subproblem in $O(kn(k + \log n))$ time, and employed a refinement phase to improve the quality of clusters generated by the completely balanced mk-means algorithm. Finally, we have demonstrated the superiority of our two-phase clustering algorithms to regular ML clustering in several diverse applications.

We have focused on hard clustering algorithms in this paper. It would be interesting to investigate the extensions to soft clustering. Also, there are several methods for for estimating the number of clusters in a model-based clustering framework [11]. The interaction of balanced clustering and model selection strategies can be investigated.

We also plan to explore more applications in which our balanced clustering approach is useful. In hybrid partitional-hierarchical clustering, an initial balanced flat partition is often desired as the starting point for subsequent hierarchical clustering. The experimental results on the synthetic dataset suggest that our balanced clustering algorithm might serve well for this purpose. Spectral clustering has been a hot area of clustering research and often reduces to using regular k-means to search for appropriate clusters in the eigenspace of either the original vector data [17] or the similarity graph constructed from the original data [24]. Our balanced k-means may be used to replace the regular k-means and get more balanced and hopefully improved spectral clustering results.

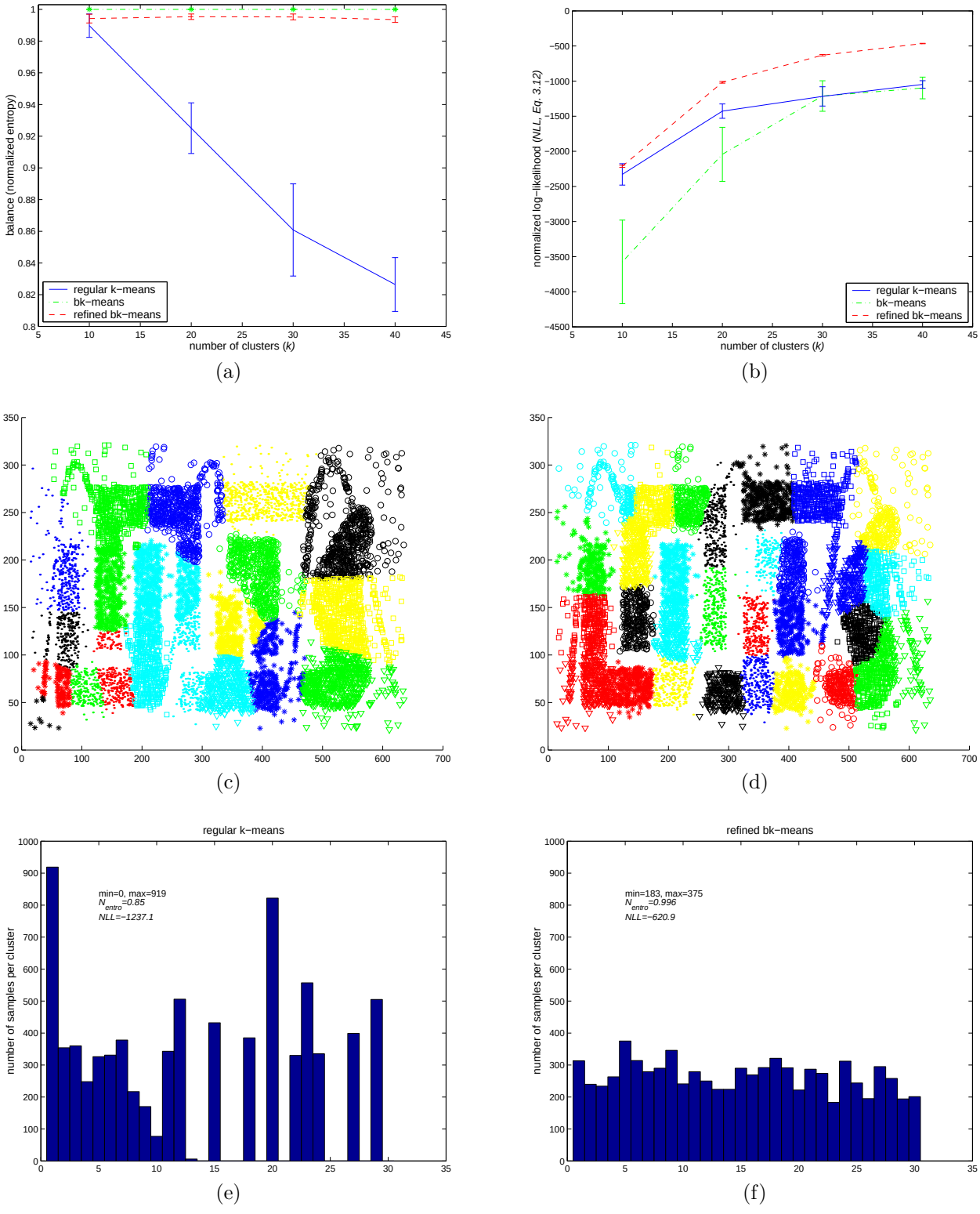

Figure 5: Results on the t4 dataset: comparison of (a) balance results and (b) NLL results; a typical 30-cluster partition from (c) regular k-means and (d) refined bk-means; histogram distribution of cluster sizes for (e) regular k-means and (f) refined bk-means.

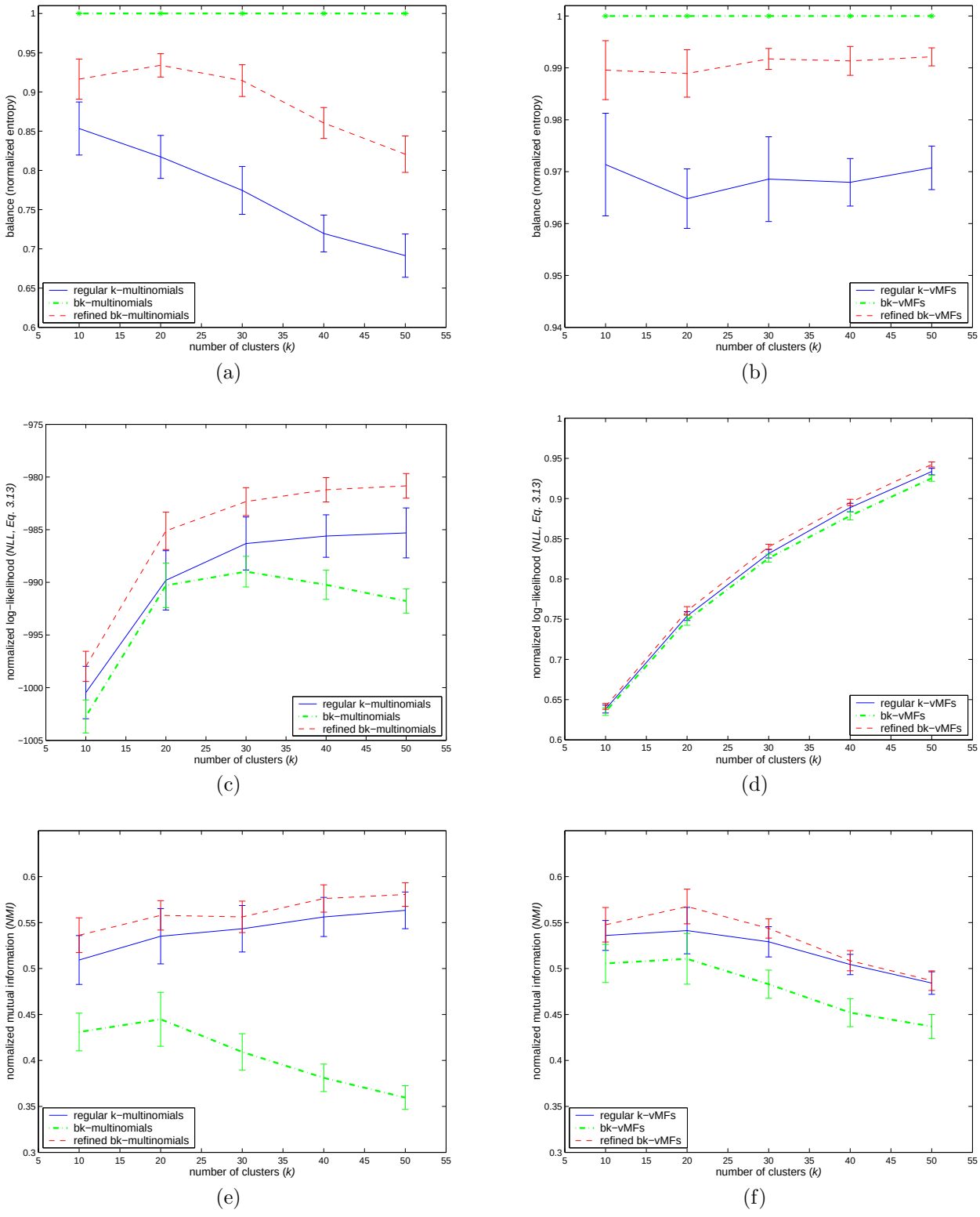

Figure 6: Results on the NG20 dataset: balance results for (a) multinomial models and (b) vMF models; log-likelihood results for (c) multinomial models and (d) vMF models; mutual information results for (e) multinomial models and (f) vMF models.

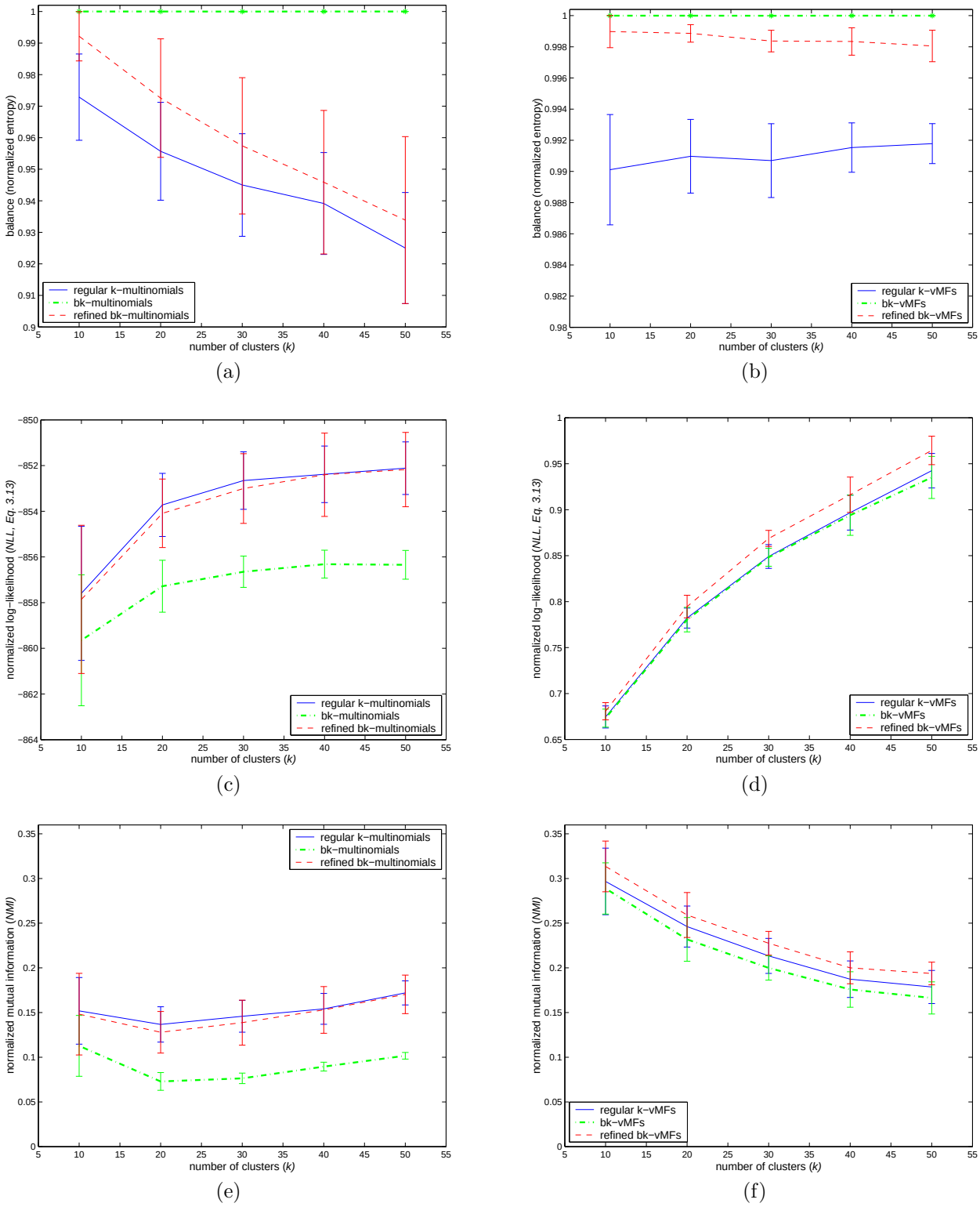

Figure 7: Results on the mini20 dataset: balance results for (a) multinomial models and (b) vMF models; log-likelihood results for (c) multinomial models and (d) vMF models; mutual information results for (e) multinomial models and (f) vMF models.

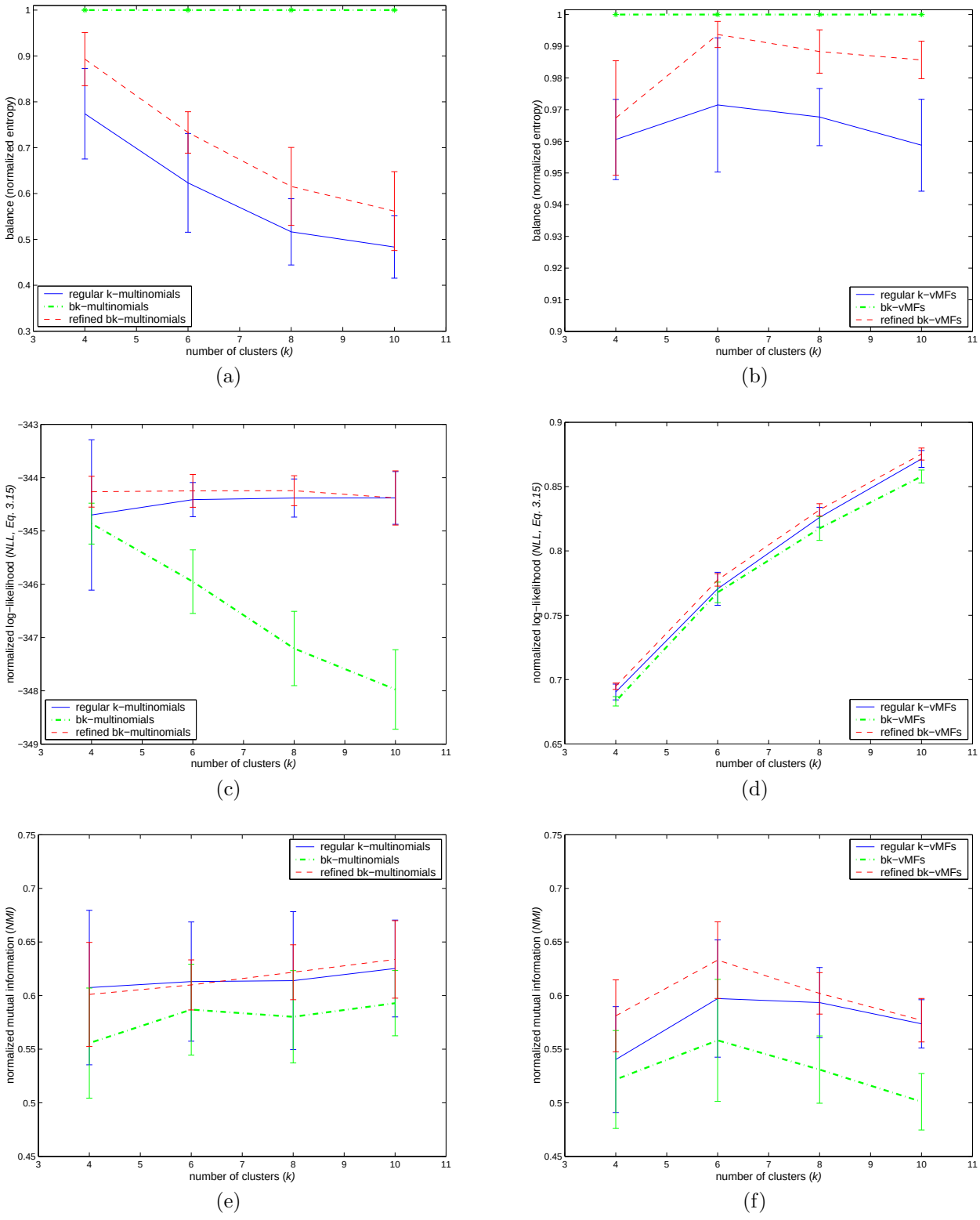

Figure 8: Results on the CLASSIC dataset: balance results for (a) multinomial models and (b) vMF models; log-likelihood results for (c) multinomial models and (d) vMF models; mutual information results for (e) multinomial models and (f) vMF models.

References

[1] S. C. Ahalt, A. K. Krishnamurthy, P. Chen, and D. E. Melton. Competitive learning algorithms for vector quantization. *Neural Networks*, 3(3):277–290, 1990.

[2] K. M. Anstreicher. Linear programming in $o([n3/\log n]l)$ operations. *SIAM Journal on Optimization*, 9(4):803–812, 1999.

[3] A. Banerjee and J. Ghosh. Frequency sensitive competitive learning for clustering on high-dimensional hyperspheres. In *Proc. IEEE Int. Joint Conf. on Neural Networks*, pages 1590–1595, Honolulu, Hawaii, May 2002.

[4] A. Banerjee and J. Ghosh. On scaling up balanced clustering algorithms. In *Proc. 2nd SIAM Int. Conf. on Data Mining*, pages 333–349, Arlington, VA, April 2002.

[5] J. D. Banfield and A. E. Raftery. Model-based Gaussian and non-Gaussian clustering. *Biometrics*, 49:803–821, 1993.

[6] J. A. Blimes. A gentle tutorial of the EM algorithm and its application to parameter estimation for gaussian mixture and hidden markov models. Technical report, U. C. Berkeley, April 1998.

[7] P. S. Bradley, K. P. Bennett, and A. Demiriz. Constrained k-means clustering. Technical Report MSR-TR-2000-65, Microsoft Research, 2000.

[8] I. S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 269–274, 2001.

[9] I. S. Dhillon and D. S. Modha. Concept decompositions for large sparse text data using clustering. *Machine Learning*, 42(1):143–175, 2001.

[10] S.-C. Fang and S. Puthenpura. *Linear Optimization and Extensions: Theory and Algorithms*. Prentice-Hall, 1993.

[11] C. Fraley and A. E. Raftery. How many clusters? Which clustering method? Answers via model-based analysis. *The Computer Journal*, 41(8), 1998.

[12] J. Ghosh. Scalable clustering methods for data mining. In N. Ye, editor, *Handbook of Data Mining*, 2002.

[13] S. Guha, R. Rastogi, and K. Shim. Cure: An efficient clustering algorithm for large databases. In *Proc. SIGMOD*, pages 73–84, New York, 1998.

[14] P. Indyk. A sublinear-time approximation scheme for clustering in metric spaces. In *Proceedings of the 40th Symposium on Foundations of Computer Science*, pages 154–159, 1999.

[15] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, New Jersey, 1988.

[16] A. Kalton, K. Wagstaff, and J. Yoo. Generalized clustering, supervised learning, and data assignment. In *Proc. 7th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pages 299–304, San Francisco, CA USA, 2001.

[17] R. Kannan, S. Vempala, and A. Vetta. On clusterings — good, bad and spectral. In *41st Annual IEEE Symp. on Foundations of Computer Science (FOCS'00)*, Redondo Beach, 2000.

[18] G. Karypis. *CLUTO - A Clutering Toolkit*. Dept. of Computer Science, University of Minnesota, May 2002.

[19] G. Karypis, E.-H. Han, and V. Kumar. Chameleon: Hierarchical clustering using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[20] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[21] M. Kearns, Y. Mansour, and A. Y. Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Proc. 13th Uncertainty in Artificial Intelligence*, pages 282–293, 1997.

[22] C. Li and G. Biswas. Applying the hidden Markov model methodology for unsupervised learning of temporal data. *International Journal of Knowledge-based Intelligent Engineering Systems*, 6(3):152–160, July 2002.

[23] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. http://www.cs.cmu.edu/ mccallum/bow, 1996.

[24] A. Y. Ng, M. I. Jordan, and Y. Weiss. On spectral clustering: analysis and an algorithm. In *Advances in Neural Information Processing Systems*, 2001.

[25] K. P. Nigam. *Using Unlabeled Data to Improve Text Classification*. PhD thesis, School of Computer Science, Carnegie Mellon University, May 2001.

[26] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of IEEE*, 77(2):257–286, 1989.

[27] K. Rose. Deterministic annealing for clustering, compression, classification, regression, and related optimization problems. *Proc. IEEE*, 86(11):2210–39, 1998.

[28] B. Scholkopf and A. Smola. *Learning with Kernels*. MIT Press, 2001.

[29] P. Smyth. Clustering sequences with hidden Markov models. In M. C. Mozer, M. I. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 648–654. MIT Press, 1997.

[30] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[31] A. Strehl and J. Ghosh. A scalable approach to balanced, high-dimensional clustering of market-baskets. In *Proc. HiPC 2000*, volume 1970 of *LNCS*, pages 525–536, Bangalore, India, December 2000. Springer.

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

[33] A. Strehl and J. Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 2002.

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

Austin, TX, USA, July 2000. AAAI/MIT Press.

[35] A. K. H. Tung, R. T. Ng, L. V. D. Lakshmanan, and J. Han. Constrained clustering on large database. In *Proc. 8th Int. Conf. on Database Theory (ICDT'01)*, pages 405–419, London, UK, 2001.

[36] V. Vapnik. *Statistical Learning Theory.* John Wiley, New York, 1998.

[37] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An efficient data clustering method for very large databases. In *SIGMOD Rec. 25*, volume 2, pages 103–114, 1996.

[38] Y. Zhao and G. Karypis. Criterion functions for document clustering: experiments and analysis. Technical Report #01-40, Department of Computer Science, University of Minnesota, November 2001.

[39] S. Zhong and J. Ghosh. HMMs and coupled HMMs for multi-channel EEG classification. In *Proc. IEEE Int. Joint Conf. on Neural Networks*, pages 1154–1159, Honolulu, Hawaii, May 2002.

[40] S. Zhong and J. Ghosh. A unified framework for model-based clustering and its applications to clustering time sequences. Technical report, ECE Dept., The University of Texas at Austin, May 2002.