Contents

List of Tables

List of Figures

0-8493-0052-5/00/$0.00+$.50

Part I

Part A

Chapter 1

Clustering with Balancing Constraints

Arindam Banerjee

Department of Computer Science and Engineering
University of Minnesota, Twin Cities

Joydeep Ghosh

Department of Electrical and Computer Engineering
University of Texas at Austin

Abstract In many applications of clustering, solutions that are balanced, i.e, where the clusters obtained are of comparable sizes, are preferred. This chapter describes several approaches to obtaining balanced clustering results that also scale well to large data sets. First, we describe a general scalable framework for obtaining balanced clustering which first clusters only a small subset of the data and then efficiently allocates the rest of the data to these initial clusters while simultaneously refining the clustering. Next, we discuss how frequency sensitive competitive learning can be used for balanced clustering in both batch and on-line scenarios, and illustrate the mechanism with a case study of clustering directional data such as text documents. Finally, we briefly outline balanced clustering based on other methods such as graph partitioning and mixture modeling.

1.1 Introduction

Several chapters in this book describe how to incorporate constraints stemming from additional information about the data, into the clustering process. For example, prior information about known labels or about the knowledge that certain pairs of points have or do not have the same label, are translated into "must-link" and "cannot-link" constraints that impact the clustering algorithms in Chapter XXX. In contrast, in this chapter we deal with a constraint that typically comes from the desiderata or needs of the end-

0-8493-0052-5/00/$0.00+$.50

application. This constraint is that the clusters be all of comparable sizes, i.e., the solution be *balanced.* Here "size" of a cluster typically refers to the number of data points in that cluster. In cases where each point may have a weight/value attached to it, for example the points represent customers and a point's weight is the monetary value of that customer, size can alternatively refer to the net weight of all points in a cluster.

In general, the natural clusters in the data may be of widely varying sizes. Moreover, this variation may not be known beforehand and balanced solutions may not be important. However, for several real life applications, having a balancing requirement helps in making the clusters actually useful and actionable. Thus this imposition comes from the associated application/business needs rather than from the inherent properties of the data. Indeed, it may be specified simply based on the end-goals even without examining the actual data.

Let us now look at some specific applications where a balanced solution is desired:

- Direct Marketing [48, 52]: A direct marketing campaign often starts with segmenting customers into groups of roughly equal size or equal estimated revenue generation, (based on, say, market basket analysis, demographics, or purchasing behavior at a web site), so that the same number of sales teams, marketing dollars, etc., can be allocated to each segment.

- Category Management [39]: Category management is a process that involves managing product categories as business units and customizing them on a store-by-store basis to satisfy customer needs. A core operation in category management, with important applications for large retailers, is to group products into categories of specified sizes such that they match units of shelf space or floor space. Another operation key to large consumer product companies such as Procter & Gamble, is to group related stock keeping units (SKUs) in bundles of comparable revenues or profits. In both operations the clusters need to be refined on an on-going basis because of seasonal difference in sales of different products, consumer trends etc. [26].

- Clustering of Documents [32, 3]: In clustering of a large corpus of documents to generate topic hierarchies, balancing greatly facilitates *browsing/navigation* by avoiding the generation of hierarchies that are highly skewed, with uneven depth in different parts of the hierarchy "tree" or having widely varying number of documents at the leaf nodes. Similar principles apply when grouping articles in a website [32], portal design, and creation of domain specific ontologies by hierarchically grouping concepts.

- Balanced Clustering in Energy Aware Sensor Networks [20, 25]: In distributed sensor networks, sensors are clustered into groups, each represented by a sensor “head”, based on attributes such as spatial location, protocol characteristics, etc. An additional desirable property, often imposed as an external soft constraint on clustering, is that each group consume comparable amounts of power, as this makes the overall network more reliable and scalable.

In all the examples above the balancing constraint is soft. It is not important to have clusters of exactly the same size but rather to avoid very small or very large clusters. A clusters that is too small may not be useful, e.g., a group of otherwise very similar customers that is too small to provide customized solutions for. Similarly very large cluster may not be differentiated enough to be readily actionable. Thus balancing may be sought even though the “natural” clusters in the data are quite imbalanced. Additionally, even the desired range of the number of clusters sought, say 5 to 10 in a direct marketing application, may come from high level “business” requirements rather than from data properties. For these reasons, the most appropriate number and nature of clusters determined from a purely data-driven perspective may not match the number obtained from a need-driven one. In such cases, constrained clustering can yield solutions that are of poorer quality when measured by a data-centric criterion such as “average dispersion from cluster representative” (kmeans objective function), even though these same solutions are more preferable from the application viewpoint. Nevertheless, a positive, seemingly surprising result, empirically illustrated later in this chapter, is that even for fairly imbalanced data, a versatile approach to balanced clustering provides comparable, and sometimes better results as judged by the unconstrained clustering objective function. Thus such balanced solutions are then clearly superior if the benefit of meeting the constraints is also factored in. The advantage is largely because balancing provides a form of regularization that seems to avoid low-quality local minima stemming from poor initialization.

In this chapter, we shall focus on partitional clustering approaches that provide balanced results while scaling well to large datasets. In Section 1.2, we present a general framework for scaling up balanced clustering algorithms capable of working with any representative based clustering algorithm, such as `KMeans`. The framework can provably guarantee a pre-specified minimum number of points per cluster, which can be valuable in several application domains, especially when the number of clusters is large and it is desirable to avoid very small or empty clusters. The framework first clusters a representative sample of the data, and then allocates the rest of the points to these initial clusters by solving a generalization of the stable marriage problem, followed by refinements which satisfy the balancing constraints.

Several practical applications can benefit from a soft balancing approach which produces approximately balanced clusters, but does not necessarily have provable balancing guarantees. Further, since several applications accumulate data over time, it is important to be able to generate online balanced clustering based on a sequence of data points. Section 1.3 describes a family of such methods based on frequency-sensitive competitive learning, applicable to both batch and online settings, along with a case study of balanced text clustering. In Section 1.4, we briefly outline other methods for balanced clustering, with particular emphasis on graph partitioning and mixture modeling based methods. Sections 1.2 and 1.3 are largely adapted from [6] and [5] respectively, and these works can be referenced by the interested reader for details. We conclude in Section 1.5 with a brief discussion on future directions.

1.2 A Scalable Framework for Balanced Clustering

In this section we describe a general framework for partitional clustering with a user specified degree of balancing, which can be provably guaranteed. The proposed method can be broken down into three steps: (1) sampling, (2) clustering of the sampled set and (3) populating and refining the clusters while satisfying the balancing constraints. For N points and k clusters, the overall complexity of the method is $O(kN \log N)$.

1.2.1 Formulation and Analysis

Let $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}, \mathbf{x}_i \in \mathbb{R}^d, \forall i$, be a set of N data-points to be clustered. Let $d : \mathbb{R}^d \times \mathbb{R}^d \mapsto \mathbb{R}_+$ be a given distance function between any two points in $\mathbb{R}^d$. The goal is find a disjoint k-partitioning $\{S_h\}_{h=1}^k$ of $\mathcal{X}$ and a corresponding set of k cluster representatives $M = \{\boldsymbol{\mu}_h\}_{h=1}^k$ in $\mathbb{R}^d$ for a given k such that the clustering objective function:[1]

$$L(\{\boldsymbol{\mu}_h, S_h\}_{h=1}^k) = \sum_{h=1}^{k} \sum_{\mathbf{x} \in S_h} d(\mathbf{x}, \boldsymbol{\mu}_h) \tag{1.1}$$

is minimized under the constraint that $|S_h| \geq m, \forall h$, for a given m with $mk \leq N$. The sampling-based method assumes that for the (unknown) *optimal* partitioning $\{S_h^*\}_{h=1}^k$,

$$\min_h \frac{|S_h^*|}{N} \geq \frac{1}{l} \tag{1.2}$$

for some integer $l \geq k$. In other words, if samples are drawn uniformly at random from the set $\mathcal{X}$, the probability of picking a sample from any particular optimal partition is at least $\frac{1}{l}$. Note that $l = k$ if and only if $|S_h^*| = N/k, \forall h$,

[1]Weighted objects can be simply catered to by incorporating these weights in both the cost and the size constraint expressions below. Weights have not been shown to keep the exposition simple.

so that the optimal partitions are all of the same size. Further, the distance function d is assumed to be well-behaved in the following sense: Given a set of points $\mathbf{x}_i, \cdots, \mathbf{x}_n$, there is an efficient way of finding the representative $\boldsymbol{\mu} = \text{argmin}_c \sum_{i=1}^{n} d(\mathbf{x}_i, c)$. For a very large class of "distance" measures called Bregman divergences, which includes the squared Euclidean distance as well as KL-divergence as special cases, the optimal representative is simply the mean of the cluster and hence can be readily computed [7]. For cosine distance, the representative is again the mean, but scaled to unit length [4]. Therefore the assumption regarding the distance function is not very restrictive.

The above balanced clustering formulation can be efficiently solved in three steps, as outlined below:

Step 1: Sampling of the given data: The given data is first sampled in order to get a small representative subset of the data. The idea is to exploit (1.2) and compute the number of samples one must must draw from the original data in order to get a good representation from each of the optimal partitions in the sampled set with high probability. By extending the analysis of the so-called Coupon Collector's problem [37] one can show the following [6]: If X is the random variable for the number of samples be drawn from $\mathcal{X}$ to get at least s points from each partition, $E[X] \leq sl \ln k + O(sl)$, where $E[X]$ is the expectation of X. Further, if $n = csl \ln k \approx cE[X]$ samples are drawn from $\mathcal{X}$ (where c is an appropriately chosen constant), then at least s samples are obtained from each optimal partition with probability at least $(1 - \frac{1}{k^d})$. To better understand the result, consider the case where $l = k = 10$, and say we want at least $s = 50$ points from each of the partitions. Table 1.1 shows the total number of points that need to be sampled for different levels of confidence. Note that if the k optimal partitions are of equal size and $csk \ln k$

d	1	2	3	4	5
Confidence, $100(1 - \frac{1}{k^d})$%	90.000	99.000	99.900	99.990	99.999
Number of Samples, n	1160	1200	1239	1277	1315

FIGURE 1.0: Number of samples required to achieve a given confidence level for k=10 and s=50.

points are sampled uniformly at random, the expected number of points from each partition is $cs \ln k$. Thus the underlying structure is expected to be preserved in this smaller sampled set and the chances of wide variations from this behavior is very small. For example, for the 99.99% confidence level in Table 1.1, the average number of samples per partition is 127, which is only about 2.5 times the minimum sample size that is desired, irrespective of the total number of points in $\mathcal{X}$, which could be in the millions for example.

Step 2: Clustering of the sampled data: The second step involves clustering the set of n sampled points, $\mathcal{X}_s$. The only requirement from this stage is to obtain a k-partitioning of $\mathcal{X}_s$ and have a representative $\boldsymbol{\mu}_h, h = 1, \cdots, k$ corresponding to each partition. There are several clustering formulations that satisfy this requirement, e.g., clustering using Bregman divergences [7] for which the optimal representatives are given by the centroids, clustering using cosine-type similarities [17, 4] for which the optimal representatives are given by the ℓ_2-normalized centroids, convex clustering [36] for which the optimal representatives are given by generalized centroids, etc. Since the size of the sampled set is much less than that of the original data, one can use slightly involved algorithms as well, without much blow-up in overall complexity.

Step 3: Populating and Refining the clusters: After clustering the n point sample from the original data $\mathcal{X}$, the remaining $(N-n)$ points need to be assigned to the clusters, satisfying the balancing constraint. This can be achieved in two phases: *Populate*, where the points that were not sampled, and hence do not currently belong to any cluster, are assigned to the existing clusters in a manner that satisfies the balancing constraints while ensuring good quality clusters; and *Refine*, where iterative refinements are done to improve on the clustering objective function while satisfying the balancing constraints all along. Both phases can be applied irrespective of what clustering algorithm was used in the second step, as long as there is a way to represent the clusters. In fact, the third step is the most critical step and the first two steps can be considered a good way to initialize the populate-refine step.

Let n_h be the number of points in cluster h, so that $\sum_{h=1}^{k} n_h = n$. Let $\mathcal{X}_u$ be the set of $(N-n)$ non-sampled points. The final clustering needs to have at least m points per cluster to be feasible. Let $b = \frac{mk}{N}$, where $0 \leq b \leq 1$ since $m \leq N/k$, be the *balancing fraction.* For any assignment of the members of $\mathcal{X}$ to the clusters, let $\ell_i \in \{1, \ldots, k\}$ denote the cluster assignment of $\mathbf{x}_i$. Further, let $S_h = \{\mathbf{x}_i \in \mathcal{X} | \ell_i = h\}$.

In *Populate*, we just want a reasonably good feasible solution so that $|S_h| \geq m, \forall h$. Hence, since there are already n_h points in S_h, we need to assign $[m - n_h]_+$ more points to S_h, where $[x]_+ = \max(x, 0)$. Ideally, each point in $\mathcal{X}_u$ should be assigned to the nearest cluster so that $\forall \mathbf{x}_i$, $d(\mathbf{x}_i, \mu_{\ell_i}) \leq d(\mathbf{x}_i, \boldsymbol{\mu}_h), \forall h$. Such assignments will be called *greedy* assignments. However, this need not satisfy the balancing constraint. So, we do the assignment of all the points as follows: (i) Exactly $[m - n_h]_+$ points are assigned to cluster $h, \forall h$, such that for each $\mathbf{x}_i$ that has been assigned to a cluster, *either* it has been assigned to its nearest cluster, *or* all clusters h' whose representatives $\mu_{h'}$ are nearer to $\mathbf{x}_i$ than its own representative μ_{ℓ_i} already have the required number of points $[m - n_{h'}]_+$, all of which are nearer to $\mu_{\mathbf{h}'}$ than $\mathbf{x}_i$; and (ii) The remaining points are greedily assigned to their nearest clusters. The assignment condition in (i) above is motivated by the stable marriage problem that tries to get a stable match of n men and n women, each with his/her preference list for marriage over the other set [27]. The populate step can be viewed as a generalization of the standard stable marriage setting in that there are k clusters that want to "get married", and cluster h wants to "marry" at least $[m - n_h]_+$ points. Hence, an assignment of points to clusters that satisfies condition in (i) is called a *stable* assignment and the resulting clustering is called *stable*.

In *Refine*, feasible iterative refinements are done starting from the clustering obtained from the first part until convergence. Note that at this stage, each point $\mathbf{x}_i \in \mathcal{X}$ is in one of the clusters and the balancing constraints are satisfied, i.e., $|S_h| \geq m, \forall h$. There are two ways in which a refinement can be done, and we iterate between these two steps and the updating of the cluster representative: (i) Point which can be moved to a cluster whose representative is nearer than their current representative, without violating the balancing constraint, are all safely re-assigned; and (ii) Groups of points in different clusters, which can only be simultaneously re-assigned in conjunction with other re-assignments to reduce the cost without violating the constraints, are obtained based on strongly connected components of the possible assignment graph, and all such group re-assignments are done.

After all the individual and group reassignments are made, the cluster representatives are re-estimated. Using the re-estimated means, a new set of re-assignments are possible and the above two steps are performed again. The process is repeated until no further updates can be done, and the refinement algorithm terminates.

1.2.2 Experimental Results

We now present results on two high-dimensional text datasets to judge both clustering quality and balancing. The Newsgroup dataset (news20) is a widely used compilation of documents from 20 usenet newsgroups, having naturally balanced clusters with approximately 1000 documents per newsgroup. We tested our algorithms on not only the original dataset, but on a variety of subsets with differing characteristics to explore and understand the behavior of the balanced clustering algorithms. The Yahoo dataset (yahoo) is a collection of 2340 Yahoo news articles belonging one of 20 different Yahoo categories, with cluster sizes ranging from 9 to 494. The dataset helps in studying the effect of forcing balanced clustering in naturally unbalanced data.

Six algorithms are compared:

- Standard `KMeans` applied to the L_2 normalized version of the data, which makes a fairer comparison for text (applying `KMeans` on the original sparse high-dimensional data gives very poor results [17].)

- `SPKMeans`, the spherical kmeans algorithm [17] (see Section 1.3) which uses cosine similarity between data points and cluster representatives and has been shown to give good results on several benchmark text datasets [17, 4].

- SPKpr, a balanced clustering method that uses SPKMeans as the base clustering algorithm and uses both the populate (p) and refine (r) steps.

For lesion studies, three variants are considered in which one or more components of the proposed framework are missing.

- SPKpnr uses SPKMeans as the base clustering algorithm. It uses the populate (p) step, but does not refine (nr) the resulting clusters. The algorithm satisfies any given balancing constraints but need not give good results since the feasible solution is not refined.
- SPKgpnr also uses SPKMeans for clustering. It uses a greedy populate (gp) scheme where every point is assigned to the nearest cluster. Further, no refinements (nr) are done after the greedy populate step. Clearly, this algorithm is not guaranteed to satisfy balancing constraints.
- SPKgpr uses SPKmeans as the base clustering algorithm. It uses greedy populate (gp) to put points into clusters, but performs a full refinement (r) after that. The algorithm is not guaranteed to satisfy the balancing constraints since the populate step is greedy and the refinements do not start from a feasible clustering.

In a tabular form, the four algorithms can be presented as follows:

	No Refine	Refine	Balancing
Greedy Populate	SPKgpnr	SPKgpr	No Guarantee
Populate	SPKpnr	SPKpr	Guaranteed

Performance of the algorithms are evaluated using one measure for the quality of clustering and two measures for balance of cluster sizes. Since class labels are available for both datasets, a suitable indicator quality of clustering (without regard for balancing) is *Normalized Mutual Information* (NMI) [47], which measures the agreement of the assigned cluster labels and the true class labels from the confusion matrix of the assignments. A value of 1 for NMI indicates perfect agreement. Quality of balancing is evaluated using two measures: (i) the *Standard Deviation in Cluster Sizes* (SDCS) and (ii) the *Ratio between the Minimum to Expected* (RME) cluster sizes. The second measure highlights situations where some very small or empty clusters are obtained. All reported results have been averaged over 10 runs. All algorithms were started with the same random initialization for a given run to ensure fairness of comparison. Moreover, each run was started with a *different* random initialization.

Figure 1.1 shows the results on news20. Recollect that this is a typical high-dimensional text clustering problem where the true clusters are balanced. As shown in Figure 1.1(a), the balanced algorithms SPKpr and SPKpnr perform as good as SPKMeans, whereas the unconstrained algorithms SPKgpr and SPKgpnr do not perform as well. Clearly, the balancing constraints resulted in better results. `KMeans` does not perform as well as the other algorithms. Under a stricter balancing requirement in Figure 1.1(b), as before, `SPKgpr` performs marginally better than `SPKpr`, but the latter satisfies the balancing constraints. The same behavior is observed for `SPKgpnr` and its corresponding `SPKpnr`. Note that among the two balancing algorithms, `SPKpr` performs much better than `SPKpnr`, thereby showing the value of the refinement step. The same is observed for the unbalanced algorithms as well. Figure 1.1(c) shows the variation in NMI across balancing constraints for the right number of clusters. We note that the refined algorithms perform much better, although the constraints do decrease the performance by a little amount. Interestingly, both `KMeans` and `SPKmeans` achieve very low minimum balancing fraction. Figure 1.1(d) shows the standard deviation in cluster sizes. The balancing algorithms achieve the lowest standard deviations, as expected. Figures 1.1(e) and 1.1(f) show the minimum-to-average ratio of cluster sizes. Clearly, the balancing algorithms respect the constraints whereas the ratio gets really low for the other algorithms. For a large number of clusters, almost all the unconstrained algorithms start giving zero-sized clusters.

Figure 1.2 shows the results on `yahoo`. This is a very different dataset from the previous datasets since the natural clusters are highly unbalanced with cluster sizes ranging from 9 to 494. The comparison on most measures of performance look similar to that of other datasets. The major difference is in the minimum-to-average ratio shown in figures 1.2(e) and (f). As expected, the balanced algorithms `SPKpr` and `SPKpnr` respect the constraints. The other algorithms (except `KMeans`) start getting zero-sized clusters for quite low values of clusters. Also, as the balancing requirement becomes more strict (as in figure 1.2(f)), the disparity between the balanced and other algorithms become more pronounced. Surprisingly, even for such an unbalanced data, the balanced algorithms, particularly `SPKpr`, perform almost as good as the unconstrained algorithms (Figure 1.2(c)).

Overall, the results show that the sampling based balanced clustering method is able to guarantee balancing properties with little or no compromise in matching cluster to class labels.

1.3 Frequency Sensitive Approaches for Balanced Clustering

An alternative approach to obtain balanced clustering is via frequency sensitive competitive learning (FSCL) methods for clustering [1], where clusters of larger sizes are penalized so that points are less likely to get assigned to them. Such an approach can be applied both in the batch as well as in the online settings [5]. Although frequency sensitive assignments can give fairly balanced clusters in practice, there is no obvious way to guarantee that every cluster will have at least a pre-specified number of points. In this section, we first outline the basic idea in FSCL, and then discuss a case study in balanced clustering of directional data, with applications in text clustering.

1.3.1 Frequency Sensitive Competitive Learning

Competitive learning techniques employ winner-take-all mechanisms to determine the most responsive cell to a given input [22, 44, 23]. If this cell or exemplar then adjusts its afferent weights to respond even more strongly to the given input, the resultant system can be shown to perform unsupervised clustering. For example, the non-normalized competitive learning version of Rumelhart and Zipser [44], essentially yields an on-line analogue of the popular `KMeans` clustering algorithm. In its simplest form, one initializes k cluster prototypes or representatives, then visits the data points in arbitrary sequence. Each point is assigned to the nearest prototype (the "winner") which in turn moves a bit in the direction of the new point to get even closer to its most recent member. The kinship to `KMeans` is obvious. There are also soft competitive learning methods with multiple winners per input [53], that can be viewed as on-line analogues of soft batch-iterative clustering algorithms such as fuzzy c-means [10] as well as the expectation-maximization (EM)-based approach to clustering data modeled as a mixture of Gaussians [11].

To address the problem of obtaining clusters of widely varying sizes, a "conscience" mechanism was proposed for competitive learning [14], that made frequently winning representatives less likely to win in the future because of their heavier conscience. This work was followed by the notable frequency sensitive competitive learning (FSCL) method [1, 18]. FSCL was originally formulated to remedy the problem of under-utilization of parts of a codebook in vector quantization. In FSCL, the competitive computing units are penalized in proportion to the frequency of their winning, so that eventually all units participate in the quantization of the data space. Specifically, [1] proposed that each newly examined point $\mathbf{x}$ be assigned to the cluster h^* where $h^* = \text{argmin}_h \ \left\{n_h \|\mathbf{x} - \boldsymbol{\mu}_h\|^2\right\}$, and n_h is the number of points currently in the h^{th} cluster with representative $\boldsymbol{\mu}_h$. Thus highly winning clusters, which have higher values of n_h, are discouraged from attracting new inputs. This also has the benefit of making the algorithm less susceptible to poor initialization. Convergence properties of the FSCL algorithm to a local minima have been studied by approximating the final phase of the FSCL by a diffusion process described by a Fokker-Plank equation [19].

Alternatively, suppose we initially start with a data model that is a mixture of identity co-variance Gaussians. Points are now sequentially examined. Each considered point is assigned to the most likely Gaussian (hard assignments), and simultaneouly *shrink* the covariance of this Gaussian in proportion to the number of points that have been assigned to it so far. In this case one can show that the assignment rule will be [5]:

$$h^* = \text{argmin}_h \ \left\{ n_h \|\mathbf{x} - \boldsymbol{\mu}_h\|^2 + d \ln n_h \right\}. \tag{1.3}$$

Note that the empirically proposed FSCL method [1] only considers $n_h\|x - \boldsymbol{\mu}_h\|^2$ while a more formal treatment of the idea results in an extra second term, namely $d \ln n_h$.

1.3.2 Case Study: Balanced Clustering of Directional Data

In this section we show how the FSCL principle can applied to problems for which the domain knowledge indicates that data is directional [33]. For example, existing literature on document clustering often normalize the document vectors to unit length after all other preprocessing have been carried out. The cosine of the angle between two such normalized vectors then serves as the similarity measure between the two documents that they represent. Normalization prevents larger documents from dominating the clustering results. Normalization of high-dimensional vectors before clustering is also fruitful for market basket data analysis if one is interested in, say, grouping customers based on the similarities between the percentages of their money spent on the various products.

A suitable generative model for directional datasets is a mixture of von Mises-Fisher (vMF) distributions [33]. The vMF is an analogue of the Gaussian distribution on a hypersphere [33, 4] in that it is the maximum entropy distribution on the hypersphere when the first moment is fixed [29] under the constraint that the points are on a unit hypersphere. The density of a d-dimensional vMF distribution is given by

$$f(\mathbf{x}; \boldsymbol{\mu}, \kappa) = \frac{1}{Z_d(\kappa)} \exp\left(\kappa \mathbf{x}^T \boldsymbol{\mu}\right), \tag{1.4}$$

where $\boldsymbol{\mu}$ represents the mean direction vector of unit L_2 norm and κ is the concentration around the mean, analogous to the mean and covariance for the multivariate Gaussian distribution. The normalizing coefficient is

$$Z_d(\kappa) = (2\pi)^{d/2} I_{d/2-1}(\kappa)/\kappa^{d/2-1}, \tag{1.5}$$

where $I_r(y)$ is the modified Bessel function of the first kind and order r [35].

Suppose one applies expectation maximization (EM) to maximize the likelihood of fitting a mixture of vMF distributions to the data, assuming the same κ for each mixture component. Consider a special case where the E-step assigns each data point to the most likely vMF distribution to have generated it. This results in the simple assignment rule: $h^* = \text{argmax}_h \ \mathbf{x}^T \boldsymbol{\mu}_h$. The M-step, which involves computing the $\boldsymbol{\mu}_h, h = 1, \cdots, k$, using the current assignments of the data, results in updating the cluster means according to:

$$\boldsymbol{\mu}_h = \frac{\sum_{\mathbf{x} \in S_h} \mathbf{x}}{\| \sum_{\mathbf{x} \in S_h} \mathbf{x} \|}. \tag{1.6}$$

Hence the EM iterations become identical to the spherical kmeans (`SPKMeans`) algorithm, introduced by Dhillon et al. [17], and shown to be far superior to regular `KMeans` for document clustering. One can further show that the objective function to be minimized by this algorithm can be expressed as:

$$L(\{\boldsymbol{\mu}_h, S_h\}_{h=1}^k) = \frac{1}{n} \sum_{h=1}^{k} \sum_{\mathbf{x} \in S_h} \mathbf{x}^T \boldsymbol{\mu}_h. \tag{1.7}$$

L can be interpreted as the average *cosine similarity* (cosine of the angle) between any vector $\mathbf{x}$ and its cluster representative $\boldsymbol{\mu}_h$. It serves as an intrinsic measure of cluster quality and will be called the `SPKMeans` objective function.

A frequency sensitive version of the above method can be constructed by making κ inversely proportional to the number of points assigned to the corresponding distribution rather than keeping it constant. Thus, if n_h is the number of points assigned to S_h, then we set $\kappa_h \propto 1/n_h$. Thus, if a point $\mathbf{x}$ is such that $\mathbf{x}^T\boldsymbol{\mu}_1 = \mathbf{x}^T\boldsymbol{\mu}_2$ but $n_{h_1} < n_{h_2}$, then $\mathbf{x}$ has a higher likelihood of having been generated from S_{h_1} than S_{h_2} in the frequency sensitive setting. Hence, the likelihood of points going to clusters having less number of points increases and this implicitly discourages poor local solutions having empty clusters or clusters having very small number of points. In [5] a hard assignment variant of EM was applied to such a frequency sensitive version of mixture of mVF distributions, to obtain three algorithms

- `fs-SPKMeans`, which is a direct extension of `SPKMeans` using frequency sensitive assignments;
- `pifs-SPKMeans`, a partially incremental version of `fs-SPKMeans` where the effective number of points per cluster are updated incrementally after processing every point and the mean of every cluster is updated in batch once in every iteration (after processing all the points); and
- `fifs-SPKMeans`, which is a fully incremental version of `fs-SPKMeans` where both the effective number of points per cluster and the cluster means are updated after processing every point.

All these algorithms need to know the number of points to be processed up-front and hence are applicable to static batch data. But suppose we are faced with streaming data and have limited storage available. Such situations are typical of non-stationary environments requiring continuous on-line adaptation [41]. The need for clustering streaming, normalized data is encountered, for example, for real-time incremental grouping of news stories or message alerts that are received on-line. For constructing a balanced, online variant of `SPKMeans`, we first note that a non-normalized mean $\boldsymbol{\mu}^{(t+1)}$ of $(t+1)$ data points can be written as a recursion in terms of $\boldsymbol{\mu}^{(t)}$ [50] as follows:

$$\boldsymbol{\mu}^{(t+1)} = \boldsymbol{\mu}^{(t)} + \frac{1}{t+1}(\mathbf{x}_{t+1} - \boldsymbol{\mu}^{(t)}) \ . \tag{1.8}$$

If the data is obtained from a stationary process, i.e., the parameters of the underlying generative model do not change with time, then $\boldsymbol{\mu}^{(t)}$, as computed by the above recursion will converge, and do not need updating after sufficiently large t. However, typical streaming data is non-stationary. There are two popular approaches taken in such cases: (i) if the data characteristics change abruptly, then such breakpoints can be detected, and a model is fitted for each segment (regime) between two successive breakpoints, assuming stationarity within such segments. Piecewise autoregressive modeling is an example of such an approach. (ii) If the data characteristics vary slowly over time, the problem may be addressed by discounting the past. In particular, a recursion can be used that keeps an exponentially decaying window over the history of observations and maintains the effective count c_{t+1} of the history rather than the exact $(t+1)$. More precisely, the approximate recursion for the mean [50] is given by:

$$\tilde{\boldsymbol{\mu}}^{(t+1)} = \tilde{\boldsymbol{\mu}}^{(t)} + \frac{1}{c_{t+1}}(\mathbf{x}_{t+1} - \tilde{\boldsymbol{\mu}}^{(t)}),$$

where $c_{t+1} = (1 - 1/L)c_t + 1$ and L is a large number [50, 42, 38]. Note that this exponential decay factor of $(1 - 1/L)$ ensures that c_{t+1} converges from below to L. Thus, after the "cold start" period is over, the history maintained in the computation has an effective length L. The choice of L depends on the degree of non-stationarity, and a fundamental tradeoff between resolution and memory depth is encountered [40]. One can take a similar approach for approximating the normalized mean. Now, to make the frequency sensitive version of `SPKMeans` applicable to streaming data, as before, we want to make $\kappa_h \propto 1/n_h$. However, the number of points to be processed, n_h is unknown and may be unbounded. Using an exponential decay recursion for n_h so that $\tilde{n}_h^{(t+1)} = (1 - 1/L)\tilde{n}_h^{(t)} + 1$ and $\tilde{n}_h^{(0)} = 0$, one obtains [5]:

$$\tilde{\boldsymbol{\mu}}_h^{(t+1)} = \frac{\tilde{\boldsymbol{\mu}}_h^{(t)} + \frac{1}{\tilde{n}_h^{(t+1)}}(\mathbf{x}_{t+1} - \tilde{\boldsymbol{\mu}}_h^{(t)})}{\|\tilde{\boldsymbol{\mu}}_h^{(t)} + \frac{1}{\tilde{n}_h^{(t+1)}}(\mathbf{x}_{t+1} - \tilde{\boldsymbol{\mu}}_h^{(t)})\|}, \tag{1.9}$$

Also, the most likely distribution to have generated a particular point $\mathbf{x}_i$ is given by

$$h^* = \underset{h}{\operatorname{argmax}} \frac{1}{\tilde{n}_h^{(t)}} \left\{ \mathbf{x}^T \tilde{\boldsymbol{\mu}}_h^{(t)} + 1 - \frac{\tilde{n}_h^{(t)}}{Ld} \log \tilde{n}_h^{(t)} \right\} . \tag{1.10}$$

Using these results, algorithm `sfs-SPKMeans`, a variant of frequency sensitive `SPKMeans` applicable to streaming data, can be constructed.

1.3.3 Experimental Results

Let us now look at how the balanced variants of `SPKMeans` compare with the original version, using the same two datasets as in Section 1.2.2. In addition to the metrics used to gauge cluster quality and balancing in Section 1.2.2, we use an additional intrinsic measure of cluster quality, namely the SPKMeans *objective function* (SOF) value (1.7). Note that using this measure favors SPKMeans which optimizes this measure, while all the proposed methods attempt to optimize modified versions of this objective that also weave in balancing constraints. As before, all the results presented are averaged over 10 runs. The initial k means of the SPKMeans were generated by computing the mean of the entire data and making k small random perturbations to this mean [16]. For stability and repeatability, the frequency sensitive algorithms were initialized at points of local minima of the SPKMeans objective function.

1.3.3.1 Experiments with Batch Algorithms

For news20, fs-SPKMeans and fifs-SPKMeans show very similar behavior (Figure 1.3). pifs-SPKMeans has a high bias towards balancing, whereas SPKMeans has no explicit mechanism for balancing. All the algorithms achieve their individual highest values of the NMI at $k = 20$, which is the correct number of clusters (Figure 1.3(a)). At $k = 20$, fifs-SPKMeans and fs-SPKMeans perform better than the other two in terms of the NMI, and also show good balancing. For lower values of k, pifs-SPKMeans performs worse than the other three which have quite similar behavior. For much higher values of k, SPKMeans has significantly higher values of NMI compared to the three proposed approaches, since it starts generating zero-sized clusters (Figure 1.3(d)) thereby maintaining the objective as well as NMI at a reasonable value. On the other hand, since none of the proposed algorithms generate zero sized clusters, their performance in terms of NMI suffers. As seen from Figures 1.3(c),(d), pifs-SPKMeans has the most bias towards balancing thereby achieving the lowest SDCS and the highest RME values for the entire range of k over which experiments were performed. It is interesting to note that fs-SPKMeans and fifs-SPKMeans seems to follow a middle ground in terms of its cluster balancing and quality biases. Surprisingly, the SOF values for the proposed algorithms are equal or greater than those achieved by SPKMeans.

As mentioned earlier, yahoo has highly imbalance in the true class sizes. Hence, results on this dataset show how the proposed algorithms handle the data when their balancing bias is not going to help. It is interesting to see that the performance of the algorithms in terms of the NMI is quite similar to what was observed for news20. As before fs-SPKMeans and fifs-SPKMeans perform very similarly and the NMI values they achieve deteriorate for values of k greater than 20, the correct number of clusters (Figure 1.4(a)). pifs-SPKMeans performs poorly in terms on the NMI because of its high bias towards balancing that does not help in this particular dataset. It also performs slightly worse than the other algorithms in terms of the SOF values (Figure 1.4(b)). However, as before, it consistently gives the lowest SDCS (Figure 1.4(c)) and highest RME values (Figure 1.4(d)). SPKMeans maintains a reasonable value of the NMI even for large values of k by generating empty clusters. It is interesting to note that due to fact that the natural clusters are not at all balanced, fs-SPKMeans and fifs-SPKMeans give quite low values of RME, but never actually give a zero-sized cluster in the range of k over which experiments were performed. Again, these two algorithms seem to have a good balance between the biases and can respond quite well to the underlying nature of the dataset.

Summary of Results: Both fs-SPKMeans and fifs-SPKMeans perform admirably when the value of k chosen is in the neighborhood of the number of classes in the data. They are comparable to or superior than SPKMeans in terms of cluster quality, and superior in terms of balancing. This result is particularly remarkable for yahoo, where the underlying classes have widely varying priors. This is indicative of the beneficial effect of the regularization provided by the soft balancing constraint. However, if k is chosen to be much larger than the number of natural clusters, SPKMeans has an advantage since it starts generating zero-sized clusters, while the others are now hampered by their proclivity to balance cluster sizes. On the other hand, if balancing is very critical, then pifs-SPKMeans is the best choice, but it has to compromise to some extent on cluster quality in order to achieve its superior balancing. So the choice of algorithm clearly depends on the nature of the dataset and the clustering goals, but in general, both fs-SPKMeans and fifs-SPKMeans are attractive even when balancing is not an objective.

1.3.3.2 Experiments with the Streaming Algorithm

The experiments with streaming algorithms were done by artificially "streaming" the static datasets. The data points are presented sequentially to the sfs-SPKMeans algorithm, repeating the process as many times as necessary in order to simulate streaming data. We call a sequence of showing every document in the selected dataset once as an *epoch*, and the algorithm is run over multiple epochs until it converges or some preset maxEpoch value is reached. We now present results corresponding to two choices of L—100 and 1000. The corresponding algorithms are referred to as sfs100-SPKMeans and sfs1000-SPKMeans respectively. Note that both these values of L are less than the data set sizes. This means that sfs-SPKMeans has less effective memory than the static algorithms. In fact, such a low effective memory handicaps the streaming algorithm as compared to the static ones which use all the data to update their parameters. As we shall see, the streaming algorithm actually performs reasonably well even with this handicap. Additional results with other values of L are given in [5].

In news20, the streaming algorithms perform significantly better than the static ones in terms of the NMI (Figure 1.5(a)). The reason for this surprising result appears to be that since the natural clusters in the data are perfectly balanced and the streaming algorithms are biased towards balanced clustering, they get the correct structure in the data due to their bias. Further, the streaming approach is possibly avoiding bad local minima that affects the performance of `KMeans` and variants. Among the streaming algorithms, `infs100-SPKMeans` performs marginally better than `infs1000-SPKMeans` though the differences are not always significant. The SOF values for the static algorithms are significantly better than those achieved by the streaming algorithms (Figure 1.5(b)). There is no significant difference in the SDCS for the various algorithms (Figure 1.5(c)). The frequency sensitive algorithms perform better than `SPKMeans` in terms of the RME values, and the streaming algorithms give higher values of RME than `fs-SPKMeans`(Figure 1.5(d)).

In yahoo, the static algorithms seem to achieve higher values of NMI than the streaming ones (Figure 1.6(a)). In the trade-off between balancing and cluster quality, the streaming algorithms seem to give more importance to the balancing aspect whereas the static ones seems to give higher priority to the cluster quality. The streaming algorithms being biased towards the balancing criterion, performs poorly in terms of the NMI in this dataset that has highly unbalanced natural clusters. Due to this bias, they give significantly better RME values as compared to the static algorithms (Figure 1.6(d)). Like news20, the SOF values achieved by the static algorithms are significantly better than those by the streaming ones (Figure 1.6(b)). Also, similar to news20, there is not much difference in the SDCS across all the algorithms (Figure 1.6(c)).

1.4 Other Balanced Clustering Approaches

In this section, we first briefly comment on the balancing properties of commonly used clustering approaches and then discuss alternative approaches for balanced clustering.

We begin by noting that widely used clustering algorithms based on kmeans, expectation maximization (EM) and variants, do not have any explicit way to guarantee that there is at least a certain minimum number of points per cluster, though, in theory, they have an implicit way of preventing highly skewed clusters [31]. For extreme situations when both the input dimensionality and the number of clusters is high, several researchers [9, 24, 17] have observed that kmeans and related variants quite often generate some clusters that are extremely small or even empty. Note that such imbalances arise in the assignment step, since the step of updating means does not directly govern cluster size. The cluster assignment step can be modified by solving a minimum cost flow problem satisfying constraints [9] on the cluster sizes, as was done in [20] to obtained balanced groupings in energy-aware sensor networks. However this approach is $O(N^3)$ and thus has poor scaling properties.

Agglomerative clustering methods also do not provide any guarantees on balancing. Complete link agglomerative clustering as well as Ward's method produce more compact clusters as compared to the single link or nearest neighbor agglomerative methods, but these clusters could be of widely varying sizes. Note that any agglomerative clustering method can be readily adapted so that once a cluster reaches a certain size in the bottom-up agglomeration process, it can be removed from further consideration. However, this may significantly impact cluster quality. Moreover, agglomerative clustering methods have a complexity of $\Omega(N^2)$ and hence do not scale well.

In contrast, certain top-down or divisive clustering methods tend to provide more balanced solutions. Most notable in this category is bisecting kmeans [46] which recursively partitions the current largest cluster into two clusters by solving a 2-means problem. If one ensures that the final split results in two clusters of the same size, then one can show that the largest cluster is no more than twice the size of the second largest one. However, no statement can be made of the smallest cluster size.

A pioneering study of constrained clustering in large databases was presented by Tung et al. [51]. They describe a variety of constraints that may be imposed on a clustering solution, but subsequently focus solely on a balancing constraint on certain key objects called pivot objects. They start with any clustering (involving *all* the objects) that satisfies the given constraints. This solution is then refined so as to reduce the clustering cost, measured as net dispersion from nearest representatives, while maintaining the constraint satisfaction. The refinement proceeds in two steps: pivot movement and deadlock resolution, both of which are shown to be NP-hard. They propose to scale their approach by compressing the objects into several tight "micro-clusters"[12] where possible, in a pre-clustering stage, and subsequently doing clustering at the micro-cluster level. Since this is a coarse grain solution, an option of finer grain resolution needs to be provided by allowing pivot points to be shared among multiple micro-clusters. This last facility helps to improve solution quality, but negates some of the computational savings in the process.

1.4.1 Balanced clustering by graph partitioning

Of the very wide variety of approaches that have been proposed for clustering [28, 21], methods based on graph partitioning form the only general category that provides soft balancing. A clustering problem can be converted into a problem of graph partitioning as follows [30, 48]: A weighted graph is constructed whose vertices are the data-points. An edge connecting two vertices has a weight proportional to the similarity between the corresponding data-points. Thus vertices that represent very similar points are more strongly connected. The choice of the similarity measure quite often depends on the problem domain, e.g., Jaccard coefficient for market-baskets, normalized dot products for text, etc. If a pairwise distance value, d, is available instead of similarity, s, then it can be converted into a similarity value using a suitable inverse, monotonic relationship such as: $s = e^{-d^2}$ or $s = \frac{1}{1+d}$ [49].

Alternatively one can apply multi-dimensional scaling to the data to get an embedding into a low-dimensional vector space from which the distances can be obtained.

The weighted graph is then partitioned into k disjoint subgraphs by removing a set of edges, known as the "cut." The basic objective function is to minimize the size of this cut, which is calculated as the sum of the weights of all edges belonging to the cut. This tends to retain highly similar points in the same partition, which is also the objective of clustering. The simple min-cut objective has no balancing constraint, and may produce cuts that isolate a very small subset of the points from the rest, but are not of high quality from a clustering viewpoint. The balanced clustering objective functions based on graph partitioning are typically a normalized variant of the simple min-cut objective that ensures that the different partitions are comparable in size. Several ways of normalizing the cut using this added penalty are surveyed in [15], which also shows how graph clustering methods are related to kernel k-means. Note that the penalty term incorporated into the min-cut problem in order to obtain useful solutions explicitly provides a soft balancing constraint.

A case study of applying graph partitioning to balanced clustering of market baskets is given in [48]. In this work, the need for balancing came from a domain requirement of obtaining groups of customers so that (i) each group has about the same number of customers, or (ii) each group represents comparable revenue amounts. Both types of constraints were obtained through a suitable formulation of the efficient hierarchical "min-cut" algorithm, METIS [30], and a simple visualization scheme was used to show the balanced nature of the clusterings obtained.

Overall, graph partitioning or spectral clustering methods often give very good results, but they involve $\Omega(N^2)$ complexity in both memory requirements and computational complexity, since the size of the similarity matrix itself is N^2. In some situations, a similarity threshold can be used, so entries with similarity value less than the threshold are zeroed out. If the resultant graph is highly sparse, then more efficient storage and computational methods are available.

1.4.2 Model-based Clustering with Soft Balancing

In model-based clustering, one estimates k probabilistic models from the N *objects* to be clustered, with each model representing a cluster. Perhaps the most well known model-based technique is to fit a mixture of k multivariate Gaussians to a set of vectors using the EM algorithm. Here each Gaussian represents a cluster. But model-based clustering is a very general and versatile framework, catering to a wide variety of data-types/datasets so long as reasonable probabilistic models are known for them [54, 55, 34, 8, 13]. For example, certain sets of strings are well-characterized using a mixture of Hidden Markov Models. For this reason, this section uses the term "objects" rather than "data-points" for the entities being clustered.

When assigning object x to cluster y, the goal is to maximize the expected log-likelihood

$$L = \sum_x P(x) \sum_y P(y|x) \log p(x|\lambda_y), \tag{1.11}$$

where λ_y represents (the parameters of) model y [31]. Directly maximizing (1.11) over $P(y|x)$ and λ_y leads to a generic model-based k-means algorithm which iterates between the following two steps:

$$P(y|x) = \begin{cases} 1, \, y = \text{argmax}_{y'} \log p(x|\lambda_{y'}); \\ 0, \text{ otherwise,} \end{cases} \tag{1.12}$$

and

$$\lambda_y = \text{argmax}_\lambda \sum_x P(y|x) \log p(x|\lambda_y) \ . \tag{1.13}$$

To make the data assignment step soft, one adds entropy terms to (1.11) [43], to get a modified objective: $L_1 = L + T \cdot H(Y|X) - T \cdot H(Y) = L - T \cdot I(X;Y)$, where $I(X;Y)$ is the mutual information between the set X of all objects and the set Y of all cluster indices. The parameter T is a Lagrange multiplier used to trade-off between maximizing the average log-likelihood L and minimizing the mutual information between X and Y, and can be interpreted as "temperature" using a analogy with determistic annealing [43]. The net effect of the added term is to modify the assignment update to:

$$P(y|x) = \frac{P(y)p(x|\lambda_y)^{\frac{1}{T}}}{\sum_{y'} P(y)p(x|\lambda_{y'})^{\frac{1}{T}}}, \tag{1.14}$$

which now fractionally assigns each object to each cluster.

In [54], a simple but effective and efficient soft balancing strategy was proposed for the general soft model-based clustering framework described above. The key idea was to add another penalty that constrains the expected number of data objects in each cluster to be equal, rather than constraining the actual number of data objects in each cluster to be equal. In the resulting algorithm, the temperature parameter controls both the softness of clustering as well as that of balancing, and provides a useful knob for users to adjust the level of balancing.

The soft balancing constraints are expressed as:

$$\sum_x P(y|x) = \frac{N}{K}, \ \forall y \ , \tag{1.15}$$

and the modified Lagrangian is

$$L_2 = L_1 + \sum_x \xi_x (\sum_y P(y|x) - 1) + \sum_y \eta_y \left(\sum_x P(y|x) - M \right) \ , \tag{1.16}$$

where ξ_x and η_y are Lagrange multipliers. The resulting assignment step is now:

$$P(y|x) = \frac{P(y)\,[e^{\eta_y} p(x|\lambda_y)]^{\frac{1}{T}}}{\sum_{y'} P(y')\,[e^{\eta_{y'}} p(x|\lambda_{y'})]^{\frac{1}{T}}}\ . \tag{1.17}$$

For balanced clustering, it makes sense to set $P(y)$ to be $1/K$, which eliminates $P(y)$ from (1.17). Substituting (1.17) into (1.15) and some algebra results in an iterative formula for $\beta_y = e^{\eta_y}$ [54]:

$$\log \beta_y^{(t+1)} = T \cdot \log\left(\frac{N}{K}\right) - T \cdot \log\left(\sum_x \frac{e^{\frac{1}{T}\log p(x|\lambda_y)}}{\sum_{y'} e^{\frac{1}{T}\left(\log \beta_{y'}^{(t)} + \log p(x|\lambda_{y'})\right)}}\right). \tag{1.18}$$

where t is the iteration number, and log() is taken to avoid possible problems with very small likelihood values. For speedier computation, an annealing approach can be taken for computing $\log \beta_y$. That is, one starts from a high temperature (e.g., $T = 0.1$) and quickly lowers the temperature toward $T = 0.01$. At every temperature a small number of iterations are run after initializing $\log \beta_y$'s using the values computed from the previous temperature.

Compared to hard balancing, soft balancing for model-based clustering can be solved exactly and efficiently using the iterative strategy described above. If we fix the maximum number of iterations, the time complexity for computing $\log \beta$'s is $O(KN)$. Detailed derivations and experimental results that show the impact of temperature on balancing as well as on cluster quality can be found in [54].

1.5 Concluding Remarks

Obtaining a balanced solution is an explicit goal in certain clustering applications, irrespective of the underlying structure of the data. In other cases, obtaining clusters of comparable sizes is not a stated objective, but some amount of balancing helps in countering poor initializations in iterative clustering algorithms that converge only to a local optimum. In this chapter we covered a variety of methods for achieving scalable, balanced clustering. An important issue that was not covered however is how to determine the appropriate number of clusters. This model selection issue in clustering been extensively studied. Techniques range from information theoretic criteria[2, 45] for model comparison to purely Bayesian approaches such as reversible jump MCMC. But there is no universally accepted solution [28]. Moreover, not much work is available on model selection within a balanced clustering framework. One promising approach is to adapt competitive learning variants that add new clusters if need be as more data is encountered (see [53] and references cited therein). Alternatively, one can first obtain solutions for different values of k and then select a suitable one based on an appropriate model selection criterion that is modified to include a balancing criterion.

Acknowledgements

This research was supported in part by the Digital Technology Center Data Mining Consortium (DDMC) at the University of Minnesota, Twin Cities, and NSF grants IIS 0307792 and III-0713142.

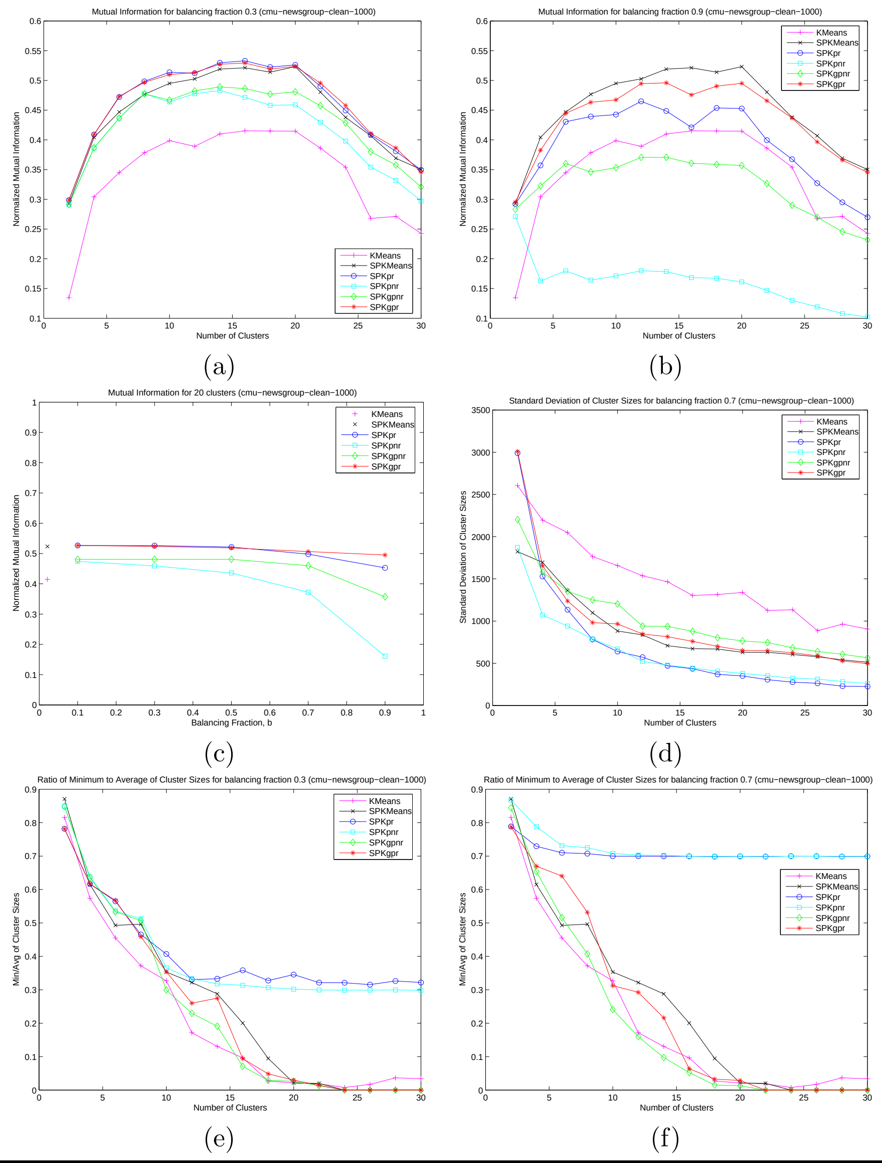

FIGURE 1.1: Results of applying the sampling based scalable balanced clustering framework on the news20 dataset.

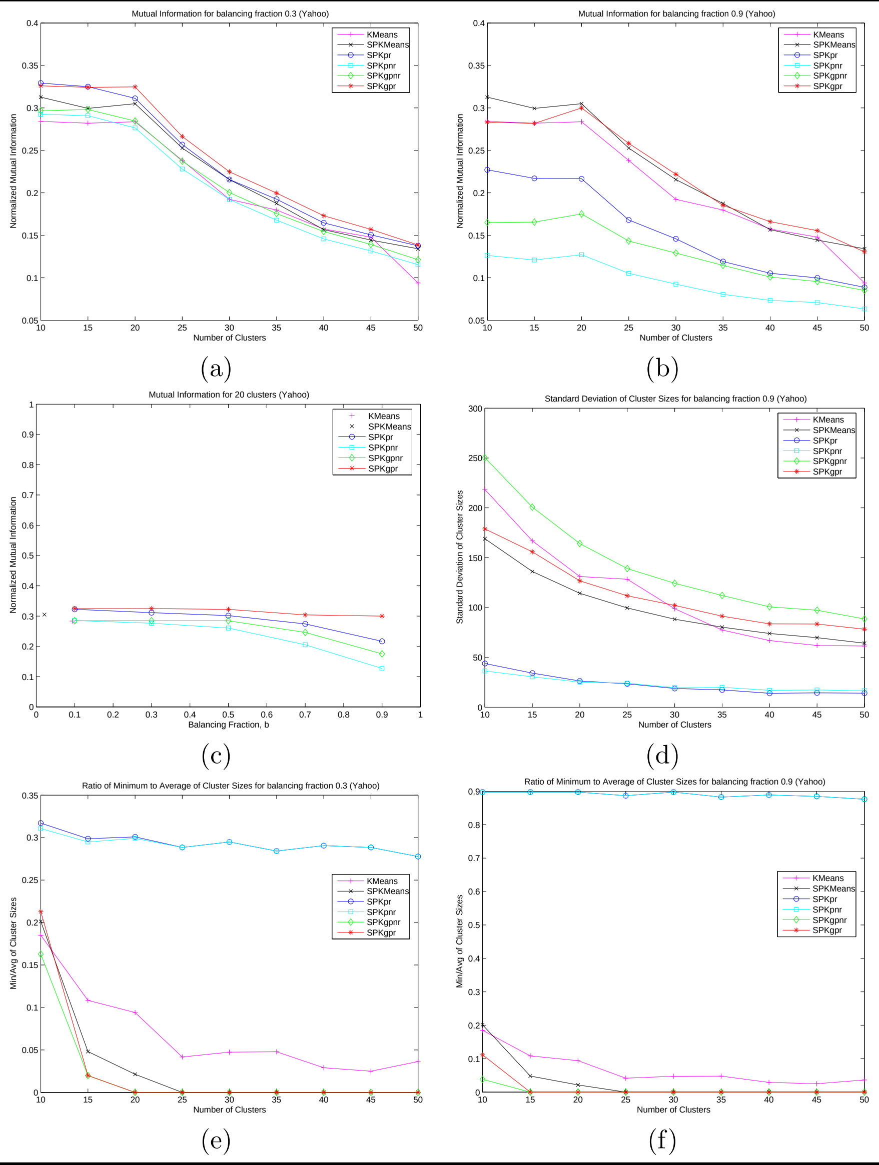

FIGURE 1.2: Results of applying the sampling based scalable balanced clustering framework on the yahoo dataset.

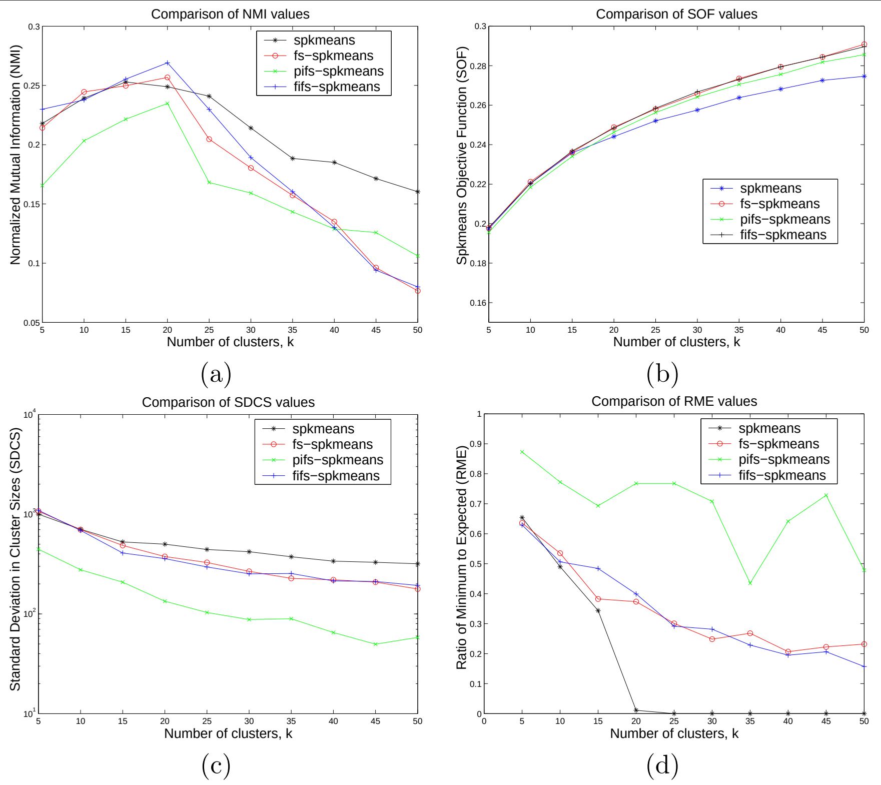

FIGURE 1.3: Comparison between the static frequency sensitive versions of spherical k-means on the News20 data: (a) the normalized mutual information values, (b) the `SPKMeans` objective function values, (c) the standard deviation in cluster sizes, and (d) the ratio of the minimum to expected cluster size values.

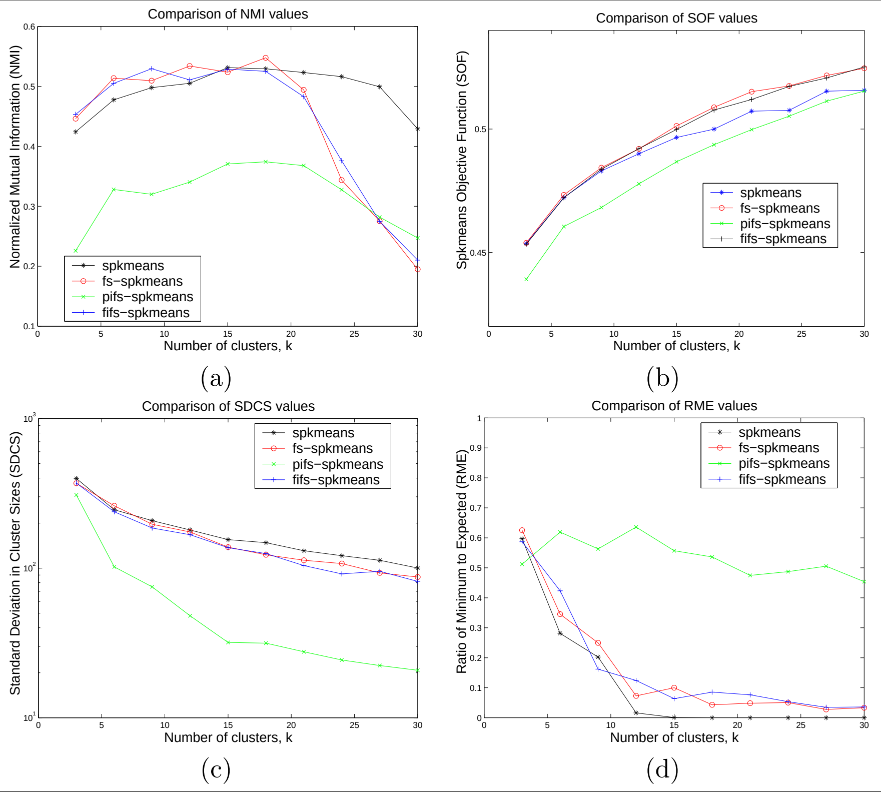

FIGURE 1.4: Comparison between the static frequency sensitive versions of spherical k-means on the Yahoo20 data: (a) the normalized mutual information values, (b) the `SPKMeans` objective function values, (c) standard deviation in cluster sizes, and (d) the ratio of the minimum to expected cluster size values.

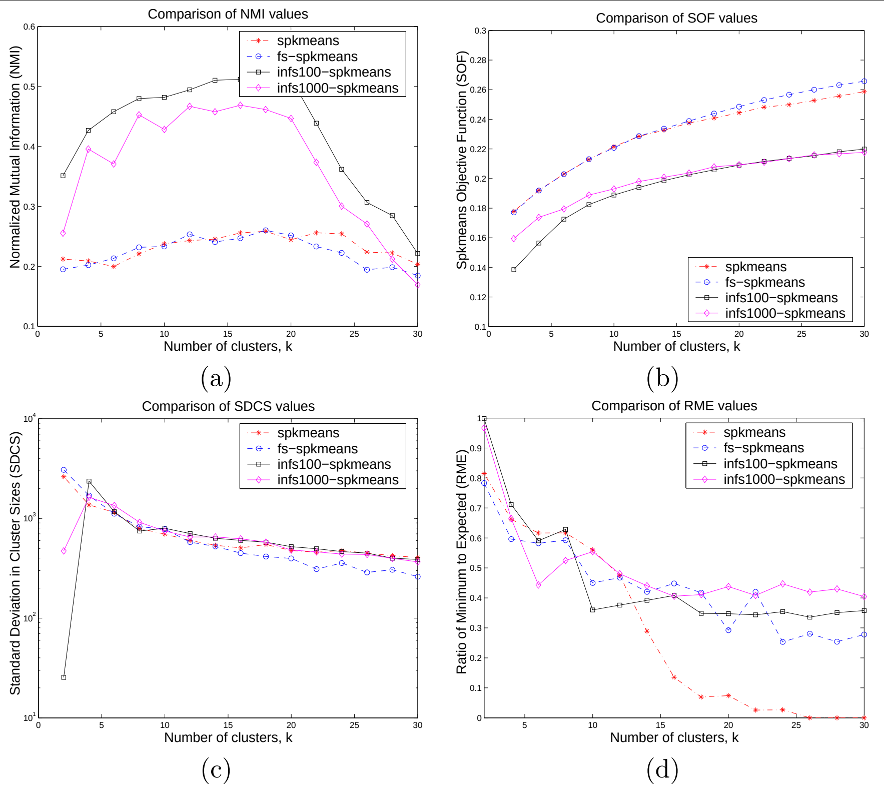

FIGURE 1.5: Comparison between streaming and static algorithms on the News20 data: (a) the normalized mutual information values, (b) the SPKMeans objective function values, (c) the standard deviation in cluster sizes, and (d) the ratio of the minimum to the expected cluster size values.

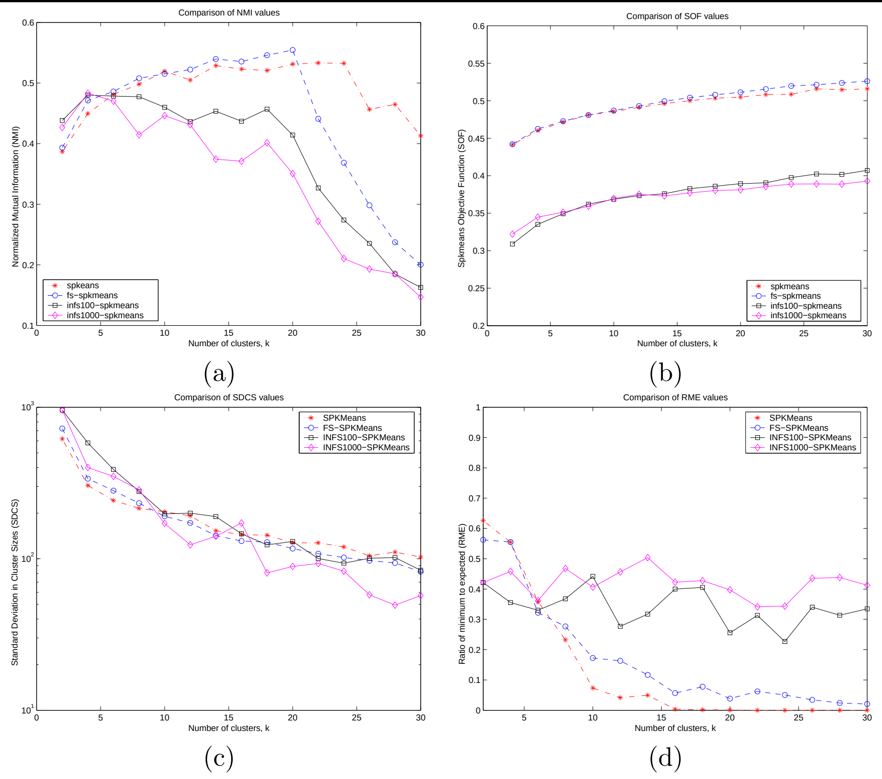

FIGURE 1.6: Comparison between streaming and static algorithms on the Yahoo20 data: (a) the normalized mutual information values, (b) the `SPKMeans` objective function values, (c) the standard deviation in cluster sizes, and (d) the ratio of minimum to the expected cluster size values.

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] H. Akaike. A new look at statistical model identification. *IEEE Transactions on Automatic Control*, AU-19:716–722, 1974.

[3] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval.* Addison Wesley, New York, 1999.

[4] A. Banerjee, I. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von Mises-Fisher distributions. *Journal of Machine Learning Research*, 6:1345–1382, 2005.

[5] A. Banerjee and J. Ghosh. Frequency sensitive competitive learning for balanced clustering on high-dimensional hyperspheres. *IEEE Transactions on Neural Networks*, 15(3):702–719, May 2004.

[6] A. Banerjee and J. Ghosh. Sclabale clustering with balancing constraints. *Data Mining and Knowledge Discovery*, 2006.

[7] A. Banerjee, S. Merugu, I. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

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

[9] K. Bennet and E. Bredensteiner. Duality and geometry in svm classifiers. In *Proceedings of the 17th International Conference on Machine Learning*, 2000.

[10] J. C. Bezdek and S.K. Pal. *Fuzzy Models for Pattern Recognition.* IEEE Press, Piscataway, NJ, 1992.

[11] 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.

[12] P. S. Bradley, U. M. Fayyad, and C. Reina. Scaling clustering algorithms to large databases. In *Proceedings of the 4th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 9–15, 1998.

[13] I. V. Cadez, S. Gaffney, and P. Smyth. A general probabilistic framework for clustering individuals and objects. pages 140–149, Aug 2000.

[14] D. deSieno. Adding conscience to competitive learning. In *IEEE Annual International Conference on Neural Networks*, pages 1117–1124, 1988.

[15] I. Dhillon, Y. Guan, and B. Kulis. A unified view of kernel k-means, spectral clustering and graph clustering. In *UTCS Technical Report TR-04-05*, 2005.

[16] I. S. Dhillon, J. Fan, and Y. Guan. Efficient clustering of very large document collections. In V. Kumar R. Grossman, C. Kamath and R. Namburu, editors, *Data Mining for Scientific and Engineering Applications.* Kluwer Academic Publishers, 2001.

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

[18] A. S. Galanopoulos and S. C. Ahalt. Codeword distribution for frequency sensitive competitive learning with one-dimensional input data. *IEEE Trans. Neural Networks*, 7(3):752–756, 1996.

[19] A. S. Galanopoulos, R. L. Moses, and S. C. Ahalt. Diffusion approximation of frequency sensitive competitive learning. *IEEE Trans. Neural Networks*, 8(5):1026–1030, Sept 1997.

[20] Soheil Ghiasi, Ankur Srivastava, Xiaojian Yang, and Majid Sarrafzadeh. Optimal energy aware clustering in sensor networks. *Sensors*, 2:258–269, 2002.

[21] J. Ghosh. Scalable clustering methods for data mining. In Nong Ye, editor, *Handbook of Data Mining*, pages 247–277. Lawrence Erlbaum, 2003.

[22] S. Grossberg. Adaptive pattern classification and universal recoding: 1. Parallel development and coding of neural feature detectors. *Biological Cybernetics*, 23:121–134, 1976.

[23] S. Grossberg. Competitive learning: From interactive action to adaptive resonance. *Cognitive Science*, 11:23–63, 1987.

[24] Y. Guan, A. Ghorbani, and N. Belacel. Y-means: A clustering method for intrusion detection. In *Proc. CCECE-2003*, pages 1083–1086, May 2003.

[25] G. Gupta and M. Younis. Load-balanced clustering of wireless networks. In *Proc. IEEE Int'l Conf. on Communications*, volume 3, pages 1848–1852, May 2003.

[26] G. K. Gupta and J. Ghosh. Detecting seasonal and divergent trends and visualization for very high dimensional transactional data. In *Proc. 1st SIAM Intl. Conf. on Data Mining*, April 2001.

[27] D. Gusfield and R. W. Irving. *The Stable Marriage Problem: Structure and Algorithms.* MIT Press, Cambridge, MA, 1989.

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

[29] J. N. Kapur and H. K. Kesavan. *Entropy Optimization Principles with Applications.* Academic Press, 1992.

[30] 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.

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

[32] P. J. Lynch and S. Horton. *Web Style Guide:Basic Design Principles for Creating Web Sites.* Yale Univ. Press, 2002.

[33] K. V. Mardia. Statistics of directional data. *J. Royal Statistical Society. Series B (Methodological)*, 37(3):349–393, 1975.

[34] G. McLachlan and K. Basford. *Mixture Models: Inference and Applications to Clustering.* Marcel Dekker, New York, 1988.

[35] N. W. McLachlan. *Bessel Functions for Engineers.* Oxford University Press, 1955.

[36] D. Modha and S. Spangler. Feature weighting in k-means clustering. *Machine Learning*, 52(3):217–237, 2003.

[37] R. Motwani and P. Raghavan. *Randmized Algorithms.* Cambridge University Press, 1995.

[38] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, 1998.

[39] Neilson Marketing Research. *Category Management: Positioning Your Organization to Win.* McGraw-Hill, 1993.

[40] J. C. Principe, J.-M. Kuo, and S. Celebi. An analysis of the gamma memory in dynamic neural networks. *IEEE Transactions on Neural Networks*, 5:331–337, March 1994.

[41] V. Ramamurti and J. Ghosh. On the use of localized gating in mixtures of experts networks. In *(invited paper), SPIE Conf. on Applications and Science of Computational Intelligence, SPIE Proc. Vol. 3390*, pages 24–35, Orlando, Fl., April 1998.

[42] V. Ramamurti and J. Ghosh. Structurally adaptive modular networks for nonstationary environments. *IEEE Transactions on Neural Networks*, 10(1):152–160, 1999.

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

[44] D. E. Rumelhart and D. Zipser. Feature discovery by competive learning. *Cognitive Science*, 9:75–112, 1985.

[45] G. Schwatz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2), 1978.

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

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

[48] A. Strehl and J. Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230, 2003.

[49] A. Strehl, J. Ghosh, and R. Mooney. Impact of similarity measures on web-page clustering. In *Proc 7th Natl Conf on Artificial Intelligence : Workshop of AI for Web Search (AAAI 2000)*, pages 58–64. AAAI, July 2000.

[50] H. G. C. Traven. A neural network approach to statistical pattern classification by "semiparametric" estimation of probability density functions. *IEEE Transactions on Neural Networks*, 2(3):366–377, 1991.

[51] A. K. H. Tung, R. T. Ng, L. V. S. Laksmanan, and J. Han. Constraint-based clustering in large databses. In *Proc. Intl. Conf. on Database Theory (ICDT'01)*, Jan 2001.

[52] Y. Yang and B. Padmanabhan. Segmenting customer transactions using a pattern-based clustering approach. In *Proceedings of ICDM*, pages 411–419, Nov 2003.

[53] Y. J. Zhang and Z. Q. Liu. Self-splitting competitive learning: A new on-line clustering paradigm. *IEEE Transactions on Neural Networks*, 13(2):369–380, March 2002.

[54] S. Zhong and J. Ghosh. A comparative study of generative models for document clustering. In *Workshop on Clustering High Dimensional Data: 3rd SIAM Conference on Data Mining*, April 2003.

[55] S. Zhong and J. Ghosh. A unified framework for model-based clustering. *Journal of Machine Learning Research*, 4:1001–1037, 2003.