Meta Clustering

Rich Caruana, Mohamed Elhawary, Nam Nguyen, Casey Smith
Cornell University
Ithaca, New York 14853
{caruana, hawary, nhnguyen, casey}@cs.cornell.edu

Abstract

Clustering is ill-defined. Unlike supervised learning where labels lead to crisp performance criteria such as accuracy and squared error, clustering quality depends on how the clusters will be used. Devising clustering criteria that capture what users need is difficult. Most clustering algorithms search for one optimal clustering based on a prespecified clustering criterion. Once that clustering has been determined, no further clusterings are examined. Our approach differs in that we search for many alternate reasonable clusterings of the data, and then allow users to select the clustering(s) that best fit their needs. Any reasonable partitioning of the data is potentially useful for some purpose, regardless of whether or not it is optimal according to a specific clustering criterion. Our approach first finds a variety of reasonable clusterings. It then clusters this diverse set of clusterings so that users must only examine a small number of qualitatively different clusterings. In this paper, we present methods for automatically generating a diverse set of alternate clusterings, as well as methods for grouping clusterings into meta clusters. We evaluate meta clustering on four test problems, and then apply meta clustering to two case studies. Surprisingly, clusterings that would be of most interest to users often are not very compact clusterings.

1. Introduction

Clustering performance is difficult to evaluate [29]. In supervised learning, model performance is assessed by comparing model predictions to supervisory targets. In clustering we do not have targets and usually do not know *a priori* what groupings of the data are best. This hinders discerning when one clustering is better than another, or when one clustering algorithm outperforms another. To make matters worse, clustering often is applied early during data exploration, before users know the data well enough to define suitable clustering criteria. This creates a chicken-or-the-egg problem where knowing how to define a good clustering criterion requires understanding the data, but clustering is one of the principal tools used to help understand the data.

This fundamental differences between supervised and unsupervised learning have profound consequences. In particular, while it makes sense to talk about the "best" model(s) in supervised learning (e.g. the most accurate model(s)), often it does not make sense to talk about the "best" clustering. Consider a database containing information about people's age, gender, education, job history, spending patterns, debts, medical history, etc. Clustering could be applied to the database to find groups of similar people. A user who wants to find groups of consumers who will buy a car probably wants different clusters than a medical researcher looking for groups with high risk of heart disease. In exploring the same data, different users want different clusterings. No "correct" clustering exists. Moreover, theoretical work suggests that it is not possible to achieve all of the properties one might desire of clustering in a single clustering of the data [20].

Most clustering methodologies focus on finding optimal or near-optimal clusterings, according to specific clustering criteria. However, this approach often is misguided. When users cannot specify appropriate clustering criteria in advance, effort should be devoted to helping users find appropriate clustering criteria. In practice, users often begin by clustering their data and examining the results. They then make educated guesses about how to change the distance metrics or algorithm in order to yield a more useful clustering. Such a search is tedious and may miss interesting partitionings of the data.

In this paper we introduce *meta clustering*, a new approach to the problem of clustering. Meta clustering aims at creating a new mode of interaction between users, the clustering system, and the data. Rather than finding one optimal clustering of the data, meta clustering finds many alternate good clusterings of the data and allows the user to select which of these clusterings is most useful, exploring the space of reasonable clusterings. To prevent the user from having to evaluate too many clusterings, the many base-level clusterings are organized into a meta clustering, a clustering of clusterings that groups similar base-level clusterings together. This meta clustering makes it easier for users to evaluate the clusterings and efficiently navigate to the clustering(s) useful for their purposes.

Meta clustering consists of three steps. First, a large number of potentially useful high-quality clusterings is generated. Then a distance metric over clusterings measures the similarity between pairs of clusterings. Finally, the clusterings are themselves clustered at the meta level using the computed pairwise similarities. The clustering at the meta level allows the user to select a few representative yet qualitatively different clusterings for examination. If one of these clusterings is appropriate for the task at hand, the user may then examine other nearby clusterings in the meta level space.

An analogy may be helpful. Photoshop, the photo editing software, has a tool called "variations" that presents to the user different renditions of the picture that have different color balances, brightnesses, contrasts, and color saturations. Instead of having to know exactly what tool to use to modify the picture (which requires substantial expertise), the user only has to be able to select the variation that looks best. The selected variation then becomes the new center, and variations of it are presented to the user. This process allows users to quickly zero in on the desired image rendition. The goal in meta clustering is to provide a similar "variations" tool for clustering so that users do not have to know how to modify distance metrics and clustering algorithms to find useful clusterings of their data. Instead, meta clustering presents users with an organized set of clustering variations; users can select and then refine the variation(s) that are best suited to their purposes.

The paper proceeds as follows. Section 2 defines meta clustering. Section 2.1 describes how to generate diverse yet high-quality clusterings. Section 2.2 describes how to measure the similarity between clusterings and use these similarities to cluster clusterings at the meta level. Section 3 describes four data sets used to evaluate meta clustering. Section 4 presents empirical results for these data sets. Section 5 presents the first case study: clustering proteins. Section 6 presents the second case study: clustering phonemes. Section 7 covers the related work. Section 8 is a discussion and summary.

2. Meta Clustering

The approach to meta clustering presented in this paper is a sampling-based approach that searches for distance metrics that yield the clusterings most useful to the user. Algorithmic (i.e. non-stochastic) approaches to meta clustering are possible and currently are being developed.

Here we break meta clustering into three steps:

1. Generate many good, yet qualitatively different, base-level clusterings of the same data.

2. Measure the similarity between the base-level clusterings generated in the first step so that similar clusterings can be grouped together.

3. Organize the base-level clusterings at a meta level (either by clustering or by low-dimension projection) and present them to users.

These steps are described in the rest of this section.

2.1. Generating Diverse Clusterings

The key insight behind meta clustering is that in many applications, data may be clustered into a variety of alternate groupings, each of which may be beneficial for a different purpose. To be useful, the alternate clusterings cannot be random partitions of the data, but must reflect *genuine structure within the data*. We follow two approaches to generate a diverse set of quality clusterings. In the first, we note that k-means generates many different reasonable clusterings (all but the "best" of which are typically discarded) because different random initializations of k-means often get stuck in different local minima. In the second approach, we apply random weights to feature vectors before clustering the data with k-means to emphasize different aspects of the data. These approaches for finding diverse clusterings are described in the remainder of Section 2.1.

2.1.1 Diverse Clusterings from K-Means Minima

K-means is an iterative refinement algorithm that attempts to minimize a squared error criterion [10]. Each cluster is initialized by setting its mean to a random point in the data set. Each step of the iterative refinement performs two tasks. First, the data points are classified as being a member of the cluster with the nearest cluster mean. Second, the cluster means are updated to be the actual mean of the data points in each cluster. This is repeated until no points change membership or for some maximum number of iterations. When no points change membership, k-means is at a *local minimum* in the search space: there is no longer a move that can reduce the squared error. The output of k-means is typically highly dependent on the initialization of the cluster means: the search space has many local minima [3, 5]. In practice, k-means is run many times with many different initializations, and the clustering with the smallest sum-of-squared distances between cluster means and cluster members is returned as the final result.

In meta clustering, however, we are interested in generating a wide variety of reasonable clusterings. The local minima of k-means provide a set of easily-attainable clusterings, each of which is reasonable since no point can change

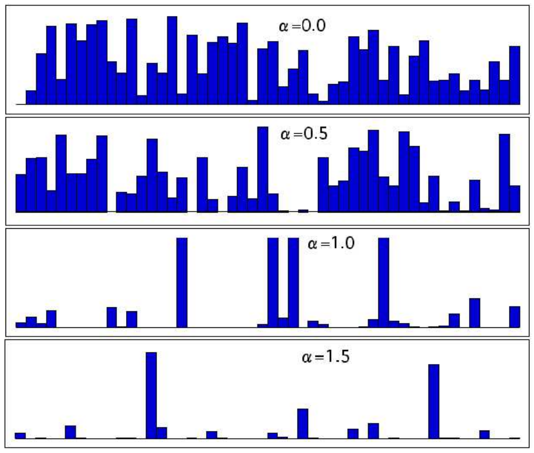


Figure 1. Zipf Distribution. Each row visualizes a Zipf distribution with a different shape parameter, α. Each row has 50 bars representing 50 random samples from the distribution, with the height of the bar proportional to the value of the sample.

membership to improve the clustering. K-means can be run many times with many different random initializations, and each local minimum can be recorded. As we shall see in Section 4.3, the space of k-means local minima is small compared to the space of reasonable clusterings, so we use an additional means of generating diverse clusterings: random feature weighting.

2.1.2 Diverse Clusterings from Feature Weighting

Consider data in vector format. Each item in the data set is described by a vector of features, and each dimension in the vector is a feature that will be used when calculating the similarity of points for clustering. By weighting features before distances are calculated (i.e. multiplying feature values by particular scalars), we can control the importance of each feature to clustering [33]. Clustering many times with different random feature weights allows us to find qualitatively different clusterings for the data using the same clustering algorithm.

Feature weighting requires a distribution to generate the random weights. We consider both uniform and power law distributions. Empirically, uniformly distributed weights often do not explore the weight space thoroughly. Consider the case where only a few features contain useful information, while the others are noise. It is unlikely that a uniform distribution would generate values that weight the few important variables highly while assigning low weights to the majority of the variables. On the other hand, weights generated from a power law distribution can weight only a few variables highly.

We will use a Zipf power law distribution because there is empirical evidence that feature importance is Zipf-distributed in a number of real-world problems [7, 14]. A Zipf distribution describes a range of integer values from 1 to some maximum value K. The frequency of each integer is proportional to $\frac{1}{i^\alpha}$ where i is the integer value and α is a shape parameter. Thus, for $\alpha = 0$, the Zipf distribution becomes a uniform distribution from 1 to K. As α increases, the distribution becomes more biased toward smaller numbers, with only the occasional value approaching K. See Figure 1. Random values from a Zipf distribution can be generated in the manner of [6].

Algorithm 1: Generate a diverse set of clusterings

```
Input: $\mathbf{X} = \{x_1, x_2, ..., x_n\}$ for $x_i \in R^d$, $k$ is the
       number of clusters, $m$ is the number of
       clusterings to be generated
Output: A set of $m$ alternate clusterings of the data
        $\{C_1, C_2, ..., C_m\}$ for which
        $C_i : \mathbf{X} \mapsto \{1, 2, ..., k\}$ is the mapping of each
        point $x \in \mathbf{X}$ to its corresponding cluster
begin
    for $i = 1$ to $m$ do
        $\alpha = rand(\text{“}uniform\text{”}, [0\ \alpha_{max}])$
        for $j = 1$ to $d$ do
            $w_j = rand(\text{“}zipf\text{”}, \alpha)$
        end
        $\mathbf{X}_i = \emptyset$
        for $x \in \boldsymbol{X}$ do
            $x' = x \bigodot w$ where $\bigodot$ is pairwise
            multiplication
            $\mathbf{X}_i = \mathbf{X}_i + \{x'\}$
        end
        $C_i =$ K-means$(\mathbf{X}_i, k)$
    end
end
```

Algorithm 1 is the procedure that generates different clusterings. First the Zipf shape parameter, α, is drawn uniformly from the interval $[0, \alpha_{max}]$. Here we use $\alpha_{max} = 1.5$. (This allows us to sample the space of random weightings, from a uniform distribution ($\alpha = 0$) to a severe distribution ($\alpha = 1.5$) that gives significant weight to just a few variables.) Then a weight vector $\mathbf{w} \in R^d$ is generated according to the Zipf distribution with that α. Next the features in the original data set are weighted with the weight vector $\mathbf{w}$. Finally, k-means is used to cluster the feature reweighted data set.

2.1.3 The Problem With Correlated Features

Random feature weights may fail to create diverse clusterings in the presence of correlated features: weights given to one feature can be compensated by weights given to other correlated features.

The problem with correlated features can be avoided by applying Principal Component Analysis [8] to the data prior to weighting. PCA rotates the data to find a new orthogonal basis in which feature values are uncorrelated. Random weights applied to the rotated features (components) yields a more diverse set of distance functions.

Typically, PCA components are characterized by the variance, σ_i, of the data set along each component, and components are sorted in order of decreasing variance. The data can be projected onto the first m of the d total components to reduce dimensionality to a m-dimensional representation of the data. To construct a data set that captures at least the fraction p of the variability of the original data (where $0 < p \leq 1$), m is set such that

$$\sum_{i=1}^{m} \sigma_i / \sum_{i=1}^{d} \sigma_i \geq p. \tag{1}$$

In the remainder of the paper PCA95 refers to PCA dimensionality reduction with $p = 0.95$.

Sometimes PCA yields a more interesting set of distance functions by compressing important aspects of the problem into a small set of components. Other times, however, PCA hides important structure. Because of this, we apply random feature weightings both before and after rotating the vector space with PCA. This is discussed further in Section 4.2.

2.1.4 Dealing with Non-vector Data

Feature weighting only works for data in feature-vector format, but data often is available only as pairwise similarities. This problem can be solved using MultiDimensional Scaling (MDS) [10]. MDS transforms pairwise distances to a feature-vector format to which random weights can then be applied.

We implement MDS following [10]. Let δ_{ij} be the original distance between points i and j, and let d_{ij} be the distance between i and j in the new vector representation. Then the following is a sum-of-squares-error function:

$$J = \frac{1}{\sum_{i<j} \delta_{ij}} \sum_{i<j} \frac{(d_{ij} - \delta_{ij})^2}{\delta_{ij}}. \tag{2}$$

The goal is to find a configuration of d_{ij} that minimizes J. Starting from a random initialization, we perform a gradient descent on the error function J. For each data point in the new space, the gradient of J is calculated with respect to that point, and the point is moved in the direction of the negative gradient. In order to avoid local minima and speed the search, we utilize a variable step size and randomize the locations of the points if progress becomes slow or halts.

2.2. Clustering Clusterings at the Meta Level

The methods in the preceding section generate a large, diverse set of candidate clusterings. Usually it is infeasible for a user to examine thousands of clusterings to find a few that are most useful for the application at hand. To avoid overwhelming the user, meta clustering groups similar clusterings together by clustering the clusterings at a meta level. To do this, we need a similarity measure between clusterings.

2.2.1 Measuring the Similarity Between Clusterings

Several measures of clustering similarity have been proposed in the literature [17, 18, 19]. Here we use a measure of clustering similarity related to the Rand index [28]: define I_{ij} as 1 if points i and j are in the same cluster in one clustering, but in different clusters in the other clustering, and I_{ij} is 0 otherwise. The dissimilarity of two clustering models is defined as:

$$\frac{\sum_{i<j} I_{ij}}{N(N-1)/2},$$

where N is the total number of data points. This measure is a metric as well. In the remainder of this paper we refer to it as Cluster Difference.

2.2.2 Agglomerative Clustering at the Meta Level

Once the distances between all pairs of clusterings are found using the Cluster Difference metric, the clusterings are themselves clustered at the meta level. This meta clustering can be performed using any clustering algorithm that works with pairwise similarity data. We use agglomerative clustering at the meta level because it works with similarity data, because it does not require the user to prespecify the number of clusters, and because the resulting hierarchy makes navigating the space of clusterings easier. ([13] presents one of the few studies to examine the tradeoffs between clustering complexity, efficiency, and interpretability.)

An alternate approach for presenting different clusterings to users is to find a low-dimensional projection of the pairwise clustering distances. For example, MDS can be used to find principal components of the meta clustering space from the clustering similarity data. The clusterings can then be presented to the user in a low dimensional plot where similar clusterings are positioned close to each other. One problem we have found with this approach is that often the meta level clustering space is not low in dimension, so any low-dimensional projection of the data has significant distortion. Although easier to visualize, this makes the 2-dimensional projections less useful than the meta clustering tree.

Data Set	# features	# cases	# trueclasses	# clusters	# points in biggest class	# features in 95 % PCA
Australia	17	245	10	10	80	10
Bergmark	254	1000	25	25	162	130
Covertype	49	1000	7	15	476	39
Letters	617	514	7	10	126	141
Protein	ud format	639	N/A	20	N/A	N/A
Phoneme	10	990	15,11	15	N/A	9

Table 1. Description of Data Sets

3. Data Sets

We evaluate meta clustering on six data sets. For each data set we also classify points using labels *external* to the clustering. We will call these test classifications the auxiliary labels. The labels are intended as an objective proxy for what users might consider to be good clusterings for their particular application. In practice, users may have only a vague idea of the desired clustering (and thus may not be able to provide the constraints necessary for semi-supervised clustering [9, 32]). Or users may have no idea what to expect from the clustering. The auxiliary labels are meant to represent one clustering users might find useful. In no way are they intended to represent an exclusive ground truth for the clustering. If such a classification existed, supervised learning would be more appropriate. Instead, the auxiliary labels are intended to represent a clustering that is good for one application, while acknowledging many applications with other good clusterings exist.

The Australia Coastal data is a subset of the data available from the Biogeoinformatics of Hexacorals environmental database [27]. The data contain measurements from the Australia coastline at every half-degree of longitude and latitude. The features describe environmental properties of each grid cell such as temperature, salinity, rainfall, and soil moisture. Each variable was scaled to have a mean of zero and a mean absolute deviation of one. The auxiliary labels are based on the “Terrestrial Ecoregions of the World” available from [15].

The Bergmark data was collected using 25 focused web crawls, each with different keywords. The variables are counts in a bags-of-words model describing the web pages. The auxiliary labels are the 25 web crawls that generated the data.

The Covertype data is from the UCI Machine Learning Repository [26]. It contains cartographic variables sampled at 30×30 meter grid cells in four wilderness areas in Roosevelt National Forest in northern Colorado. Data was scaled to a mean of zero and a standard deviation of one. The true forest cover type for each cell is used as the auxiliary labels.

The letters data is a subset of the isolet spoken letter data set from the UCI Machine Learning Repository [26]. We took random utterances of the letters A, B, C, D, F, H, and J in these proportions: A: 109 B: 56 C: 126 D: 59 F: 49 H: 62 J: 53. Each utterance is described by spectral coefficients, contour features, sonarant features, pre-sonarant features, and post-sonarant features. The spoken letters are the auxiliary labels.

The Protein data is the pairwise similarities between 639 proteins. It was created by crystallographers developing techniques to learn relationships between protein sequence and structure. This data is used as a case study in Section 5.

The Phoneme data is from the UCI Machine Learning Repository [26]. It records the 11 phonemes of 15 speakers. This data is used as a case study in Section 6.

See Table 1 for a summary of the data sets.

4. Experimental Results

In this section we present empirical results on four test problems used to develop meta clustering. First, in Section 4.1 we show the effect of Zipf random weighting of feature vectors. In Section 4.2 we compare results between using PCA prior to clustering with not using PCA. In Section 4.3 we compare k-means with weighted feature vectors to the local minima found by k-means on unweighted data. Then in Section 4.5 we show the results of agglomerative hierarchical clustering at the meta level. The results demonstrate the importance of generating a diverse set of clusterings when the clustering objective is not well defined prior to clustering.

We examine two clustering performance metrics. The first is compactness. Compactness is defined as:

$$\frac{\sum_{i=1}^{k} N_i \frac{\sum_{j=1}^{N_i-1}\sum_{k=j+1}^{N_i} d_{jk}}{N_i(N_i-1)/2}}{N}, \tag{3}$$

where k is the number of clusters; N_i is the number of points in the i^{th} cluster; d_{jk} is the distance between points j and k, and $N = \sum_{i=1}^{k} N_i$. Compactness measures the average pairwise distances between points in the same cluster. *Regardless of the feature weighting used in clustering, compactness is always measured relative to the original data set.* In most of the traditional clustering algorithms such as k-means and hierarchical agglomerative clustering, the optimization criterion is closely related to this measure of compactness.

The second clustering performance metric is accuracy, which is measured relative to the auxiliary labels for each data set. Again, the auxiliary labels are only a proxy for what a specific user might want in a particular application. They do not represent a single "true" classification of the data, as different users may desire different clusterings for alternate applications. Accuracy is defined as:

$$\frac{\sum_{i=1}^{k} max(C_i|L_i)}{N}, \tag{4}$$

where C_i is the set of points in the i^{th} cluster; L_i is the labels for all points in the i^{th} cluster, and $max(C_i|L_i)$ is the number of points with the plurality label in the i^{th} cluster (if label l appeared in cluster i more often than any other label, then $max(C_i|L_i)$ is the number of points in C_i with the label l).

Determining the number of clusters in a data set is challenging [25]. Indeed, the "correct" number of clusters depends on how the clustering will be used. For simplicity, we make the unrealistic assumption that the desired number of clusters is predefined. It is easy to extend meta clustering to explore the number of clusters as well: in addition to k-means local minima and various variable weighting, clusterings can be generated with different numbers of clusters. Our similarity metric can accommodate clusterings with different numbers of clusters, so the meta-level clustering will still group similar clusterings together, even if they have different numbers of clusters. *Using a fixed k helps to demonstrate how effective the methods we present are at generating diverse clusterings.*

4.1. Effect of Zipf Weighting

In Figure 2 each point in the plots represents an entire clustering of the data. The scatter plots in each row show clusterings generated by weighting features with Zipf-distributed random weights with different shape parameters α for the four test data sets. For this figure we test Zipf distributions with α = 0.00, 0.25, 0.50, 0.75, 1.00, 1.25, and 1.50.

Note that for different α values, feature weightings explore different regions of the clustering space. In all data sets that we test, as the α value increases, feature weighting explores a region of lower compactness. We also observe an interesting phenomenon that some of the most accurate clusterings are generated when applying feature weighting with higher α values. In particular, high α in the Covertype data set reveals a cloud of more accurate (and much less compact) clusterings which do not show up at low α values.

In the first row, when $\alpha = 0.00$, the Zipf distribution is equivalent to a uniform distribution. It is clear that a uniform distribution alone is insufficient to explore the clustering space.

4.2. Feature Weighting Before and After PCA

Figure 3 shows scatter plots of clusterings generated by weighting features with Zipf-distributed random weights for the four test problems. Each point in each plot represents an entire clustering of the data. The x-axis is the compactness of the clusterings (as defined in Section 4). The y-axis is the accuracy of the clusterings using the auxiliary labels. The top row shows clusterings generated by random Zipf weighting applied to the features. The second row shows clusterings generated by Zipf weighting on the components of PCA95.

Although there is correlation between compactness and accuracy, the correlation is not perfect. On non-PCA Australia, for example, the most accurate clustering is moderately compact but is clearly not the most compact clustering: 55% of the clusterings are more compact than the most accurate one. On non-PCA Letter, the association between compactness and accuracy is stronger; the most accurate clusterings are nearly the most compact. On non-PCA Covertype, however, the most accurate clusterings are not the most compact ones. In fact, for the most accurate clusterings there is a weak inverse relationship between compactness and accuracy.

For Australia, Bergmark, and Letter, PCA95 yields more diverse clusterings – the figures extend further to the lower right while maintaining clusterings in the upper left. In particular, for Bergmark, PCA95 generates more compact, high accuracy clusterings. For Covertype, however, PCA95 yields less diverse clusterings and completely eliminates the cloud of high-accuracy clusterings. Because PCA yields more diverse clusterings on some problems, and less diverse clusterings with other problems, we generate clusterings by applying Zipf weighting both before and after PCA95. The 3rd row in the figure shows clusterings generated both ways. It is the union of the clusterings in rows 1 and 2. Combining clusterings generated with these two methodologies yields the most diverse set of clusterings across a variety of problems.

The bottom row of Figure 3 shows histograms of the compactness of the clusterings. The most compact clusterings are on the left sides of the histograms. The arrow in each figure marks the most accurate clustering within each distribution. In Australia and Covertype, the most accurate clusterings do not occur in the top half of the most compact clusterings. In Bergmark and Letter, the most compact clusterings occur 20-30% down in the distribution. Note that standard clustering techniques would never have found these most accurate clusterings. Looking at the extreme leftmost points in each plot, it is clear that the most compact clustering is significantly less accurate than the most accurate clustering. Since there exist reasonable clusterings

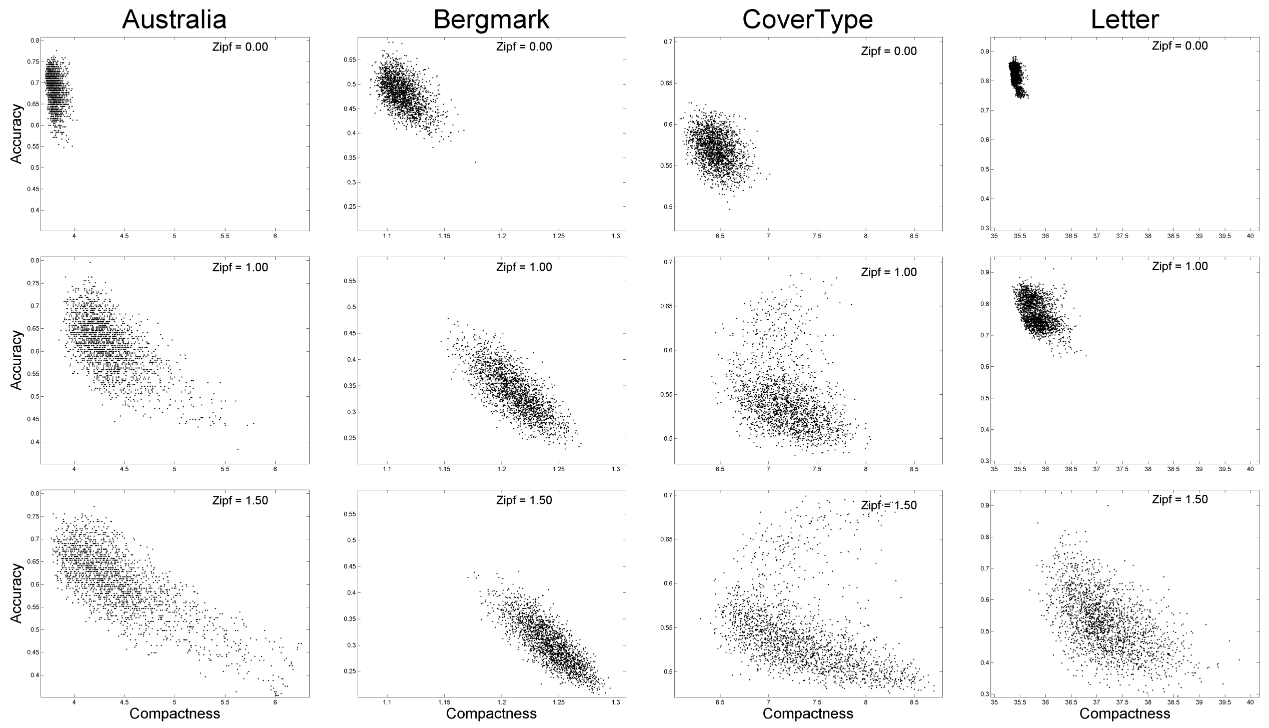


Figure 2. Accuracy vs. compactness for Zipf-weighted sequences for the Australia, Bergmark, Covertype, and Letter data sets. *Each point is a full clustering.*

which are more accurate than the most compact clustering, we conclude that the usual method of searching for only the most compact clustering can be counterproductive. It is important to explore the space of reasonable clusterings more thoroughly.

4.3. Local Minima vs. Feature Weighting

Figure 4 shows scatter plots of clusterings generated using k-means, comparing weighting features with Zipf distributed random weights versus without using feature weighting for the four test problems. The x-axis is the compactness of the clusterings (as defined in Section 4). The y-axis is the accuracy of the clusterings using the auxiliary labels. The top row shows the union of clusterings generated by random Zipf weighting applied to the original feature vector and the clusterings generated by Zipf weighting on orthogonalized components of PCA95. The bottom row shows the clusterings generated using iterated k-means where each point is the result of a different random initialization.

For Australia, Bergmark, and Letter, weighting features yields more diverse clusterings – the figures extend further to the lower right, while retaining points in the upper left. For Covertype, not applying feature weighting fails to discover the cloud of more accurate (yet not compact) clusterings. For Australia, different random initialization of k-means without feature weighting manages to find more clusterings in the upper left corner (both accurate and compact).

4.4. Other Clustering Methods

For a thorough comparison, we also apply other clustering approaches to the four data sets: hierarchical agglomerative clustering (HAC) [21], EM-based mixture model clustering [22], and two types of spectral clustering [23, 1]. HAC is implemented using three different linkage criteria: single (min-link), complete (max-link), and centroid (average-link) [21]. The EM-based mixture model clustering estimates Gaussian mixture models using the EM algorithm. The result for the EM-based mixture model clustering is the highest accuracy from multiple runs with different initialization. For both the spectral clustering methods, we also report the highest accuracy from multiple runs. Spectral clustering requires an input similarity matrix S, where $S = exp(-||x_i - x_j||/\sigma^2)$. To obtain a variety of results, we used different values of the shape parameter, $\sigma = \frac{\max\{||x_i - x_j||^2\}}{2^{k/8}}$, for $k = 0, .., 64$. The clusterings obtained from other methods fall within the range of clusterings generated from k-means with random feature weighting. Across the four data set the spectral method proposed

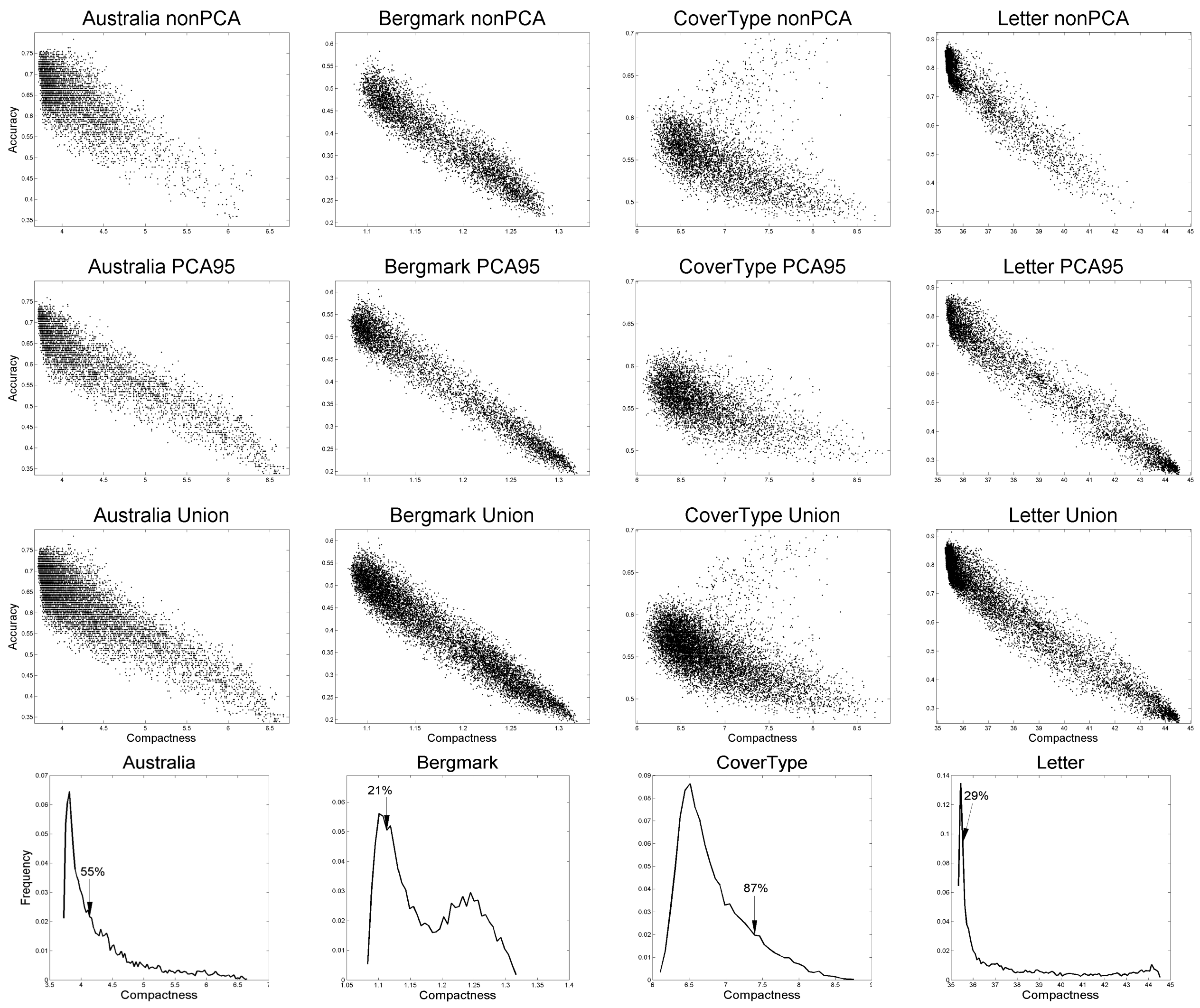


Figure 3. Accuracy vs. compactness for Zipf-weighted clusterings before PCA (1st row), after PCA95 (2nd row), the union of non-PCA with PCA95 (3rd row), and the histograms (4th row) of the compactness of the clusterings found in the 3rd row for the four data sets. Each point is a full clustering. The arrows on the histograms indicate the location of the most accurate clustering, and the number above is the percent of clusterings more compact than the most accurate one.

by [1] has the best performance, though it (and the other alternative clustering methods) are always outperformed by at least one clustering found with the metaclustering technique using kmeans for the base-level clusterings. We are currently investigating random feature weighting with other clustering techniques (in addition to k-means) to create diverse base-level clusterings.

4.5. Agglomerative Clustering at the Meta Level

In this section we cluster the clusterings at the meta level using agglomerative clustering with the average link criterion described in [21] and show that clusterings near each other at the meta level are similar, and clusterings far from each other at the meta level are qualitatively different.

Figure 5 shows the meta level clustering trees for two data sets, Australia and Letter. Each node in the clustering tree is colored by accuracy. Yellow indicates low accuracy. Blue indicates high accuracy. Note that clusterings of similar accuracy tend to be grouped together at the meta level even though meta clustering has no information about accuracy. The results suggest that groups of clusterings at the meta level reflect aspects of the base-level clusterings that might be important to users.

Figure 6 shows four representative clusterings of the Australia data set selected from the meta clustering tree.

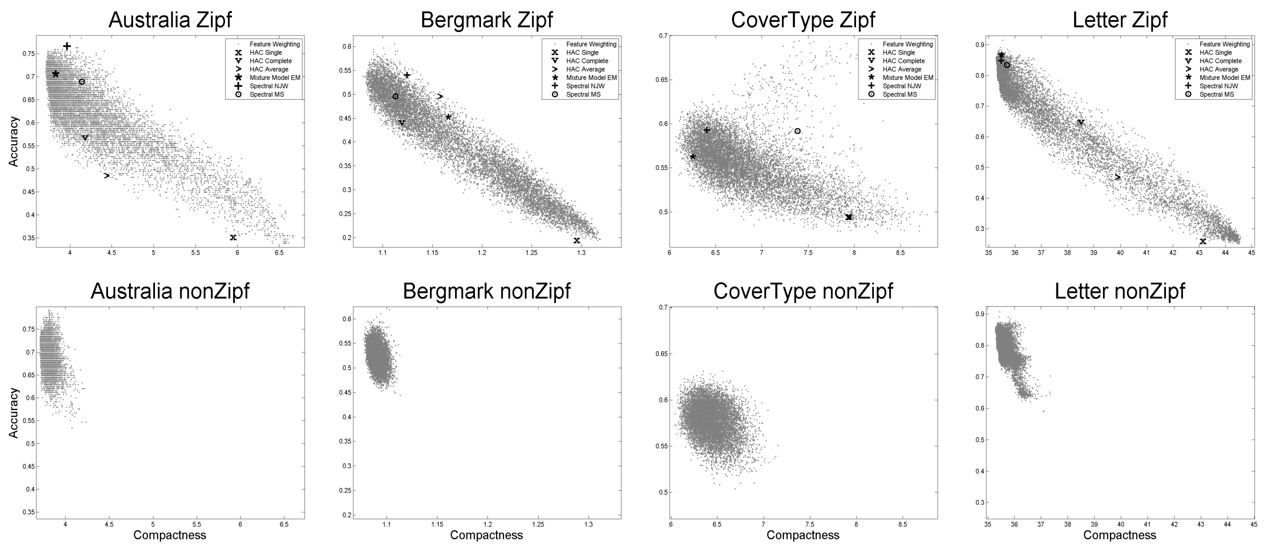


Figure 4. Accuracy vs. compactness for Zipf-weighted clusterings (top row) and iterated k-means (bottom row) of the four data sets: Australia, Bergmark, Cover Type, and Letters. Each point is a full clustering of the data. The top row also includes the accuracy and compactness of other popular clustering methods marked with special symbols.

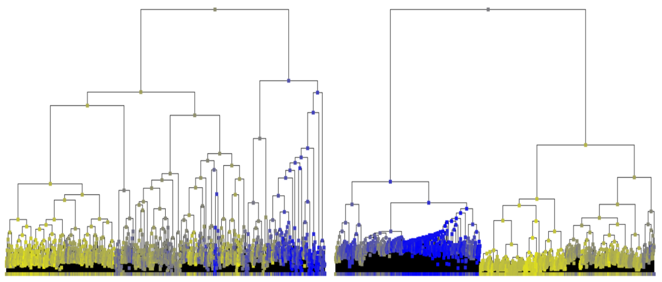


Figure 5. Meta level clustering trees for Australia (left) and Letter (right). Color (yellow to blue) indicates to accuracy (low to high). (The meta clustering dendrograms are much easier to interpret if viewed in color.)

The clusterings on the left were selected as two very similar (but not identical) clusterings from one region of the tree, while the two on the right were selected to be distinct from each other and from the two on the left. Each represents a different, reasonable way of clustering the data set. Once again we see that clusterings close to (far from) each other at the meta level are similar to (different from) each other.

Figure 7 shows compactness (Equation 3) vs. the number of clusters for hierarchical agglomerative clustering at the meta level for the four data sets. These plots can be used to detect structure at the meta level; jumps in the curves indicate merges between groups of dissimilar clusterings. The Australia, Covertype, and Letter data sets exhibit significant structure at the meta level. However, in the Bergmark data set, little structure at the meta level is observed. (Meta clustering can be useful even in the absence of significant structure at the meta level since clustering at the meta level will still group similar clusterings together, making exploring the space of clusterings easier.)

5. Case Study: Protein Clustering

In this section we apply meta clustering to a real protein clustering problem. The protein data consists of pairwise distances between 639 proteins. The distance between each pair of proteins is computed by aligning 3-D structures of each protein and computing the mean distance in angstroms between corresponding atoms in the two structures.

This data was created by crystallographers developing automated techniques to learn relationships between protein sequence and structure. For their work they need to find groups of proteins containing as many proteins with similar structure as possible. To achieve this goal, and also to better understand the data, the developers employed a number of techniques including clustering, MDS, univariate, and multivariate statistical analysis. In the course of their work they generated and tested a large variety of hand-tuned clustering criteria before finding satisfactory clusterings. After a month of effort, the two largest homogeneous groups of proteins (less than 1.25 angstrom mean distance between aligned atoms) discovered contained 28 and 30 proteins.

We apply meta clustering to the same data to try to automatically find sets of proteins similar to those the experts found manually. For this data we only have pairwise distances between proteins, not a feature-space, so we can-

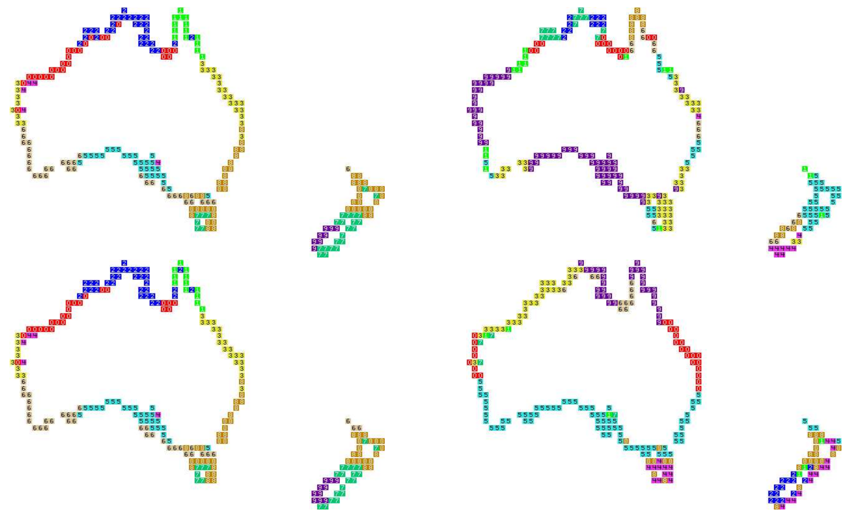


Figure 6. Alternate clusterings for the Australia data set. Each point is colored and numbered by cluster membership. The two clusterings on the left are similar, but not identical, while the two on the right are distinct from each other and from the two on the right. All four are reasonable ways of clustering Australia. (NOTE: Although the figures include numbers visible in black and white, the figures are much easier to interpret in color.)

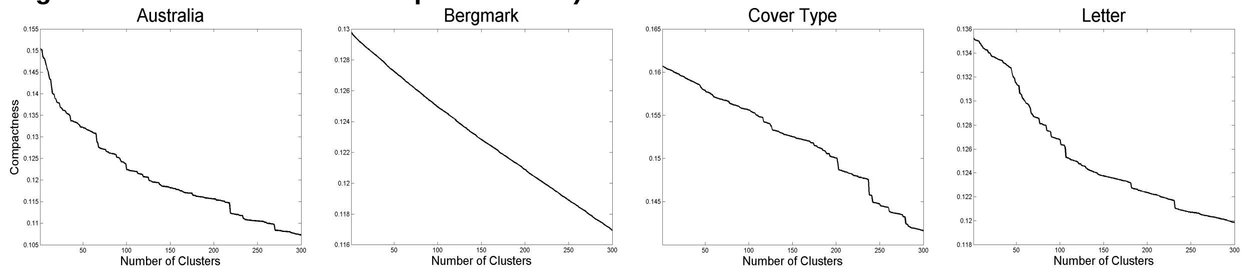


Figure 7. The compactness plot of the hierarchical agglomerative clustering at the meta level of the union of non-PCA with PCA95 of the Australia, Bergmark, Covertype, and Letter data sets.

not apply random Zipf weighting directly to the original attributes. Instead, we use the MDS method described earlier in Section 2.1.4 to convert the pairwise distance matrix to a vector space, and then apply random Zipf weighting.

We apply the meta clustering method described in Section 2.1.2 using random Zipf weighting with Zipf shapes selected uniformly on the range (0.0,1.5). For each weighting we run iterated k-means to find clusterings (local minima). We repeat this process 5000 times, yielding 5000 different clusterings of the protein data. The left plot in Figure 8 shows the number of points in the largest cluster satisfying the 1.25 angstrom constraint, plotted as a function of clustering compactness. Note that although the majority of clusterings found by meta clustering do not have large clusters with 30 or more points, meta clustering has found a number of clusterings that contain clusters with more than 30 points. The largest homogeneous cluster found by meta clustering contains 48 proteins, 60% more than the experts were able to find using manually guided methods. In a few hours meta

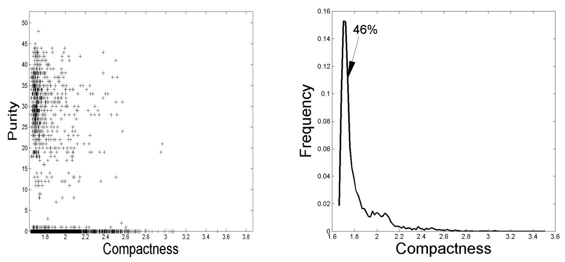


Figure 8. Meta Clustering of the Protein Data

clustering finds better clusterings than could be found manually with a month of work. The compactness histogram in the right of the Figure 8 shows that the “optimal” clustering had mediocre overall compactness, falling near the middle of the distribution of clustering compactnesses.

An examination of compactness (see Section 4.5) as a function of the number of clusters for agglomerative clustering at the meta level shows large jumps, suggesting structure at the meta level, i.e., meta clustering has found qualitatively different ways of clustering the protein data that cluster together at the meta level. If users needed clusterings that satisfied different criteria, it is likely that groups of alternate clusterings have already been found by meta clustering that would perform well according to these other criteria.

The clustering goal used in this case study (clusterings smaller than 1.25 angstroms containing many points) was not known in advance when crystallographers began working with this data. This criterion emerged only after examining the results of many clusterings. Optimizing directly to this criterion is not straightforward. Meta clustering automatically finds a diverse set of clusterings, a few of which have the desired property, without knowing the criterion in advance and without optimizing directly to that criterion.

6. Case Study: Phoneme Clustering

In this section we apply meta clustering to a phoneme clustering problem. The data set contains 15 different speakers saying 11 different phonemes 6 times each (for a total of 990 data points). For this data set, we consider users interested in identifying either speakers or phonemes and evaluate the clusterings based on both of these criteria.

	Compactness	Speaker ACC	Phoneme ACC
All	**2.66**	0.291	0.405
MC1	2.86	0.296	0.400
MC2	2.91	0.264	0.369
MC3	2.75	0.307	0.379
MC4	3.25	0.291	0.242
MC5	3.44	0.247	0.309
MC6	3.15	0.287	0.355
MC7	3.59	0.191	0.364
MC8	3.37	0.217	0.323
MC9	3.83	0.244	0.248
MC10	3.49	0.202	0.248
MC11	3.05	0.266	0.292
MC12	3.31	**0.333**	0.275
MC13	2.97	0.243	0.431
MC14	3.84	0.182	0.220
MC15	3.08	0.233	0.374
MC16	3.45	0.212	**0.445**

Table 2. Meta Clustering Aggregation of the Phoneme data set

Meta clustering successfully finds clusterings that are accurate for each criterion. Figure 9 shows the scatter plots of clusterings (top two rows) and the meta level clustering dendrograms (bottom row) colored with respect to the two accuracy measures (identifying speakers and recognizing phonemes). In the scatter plots of compactness and accuracy (top row), there is a small cloud of clusterings with high accuracy and medium compactness. If the task were to identify speakers, the most accurate clustering occurs at

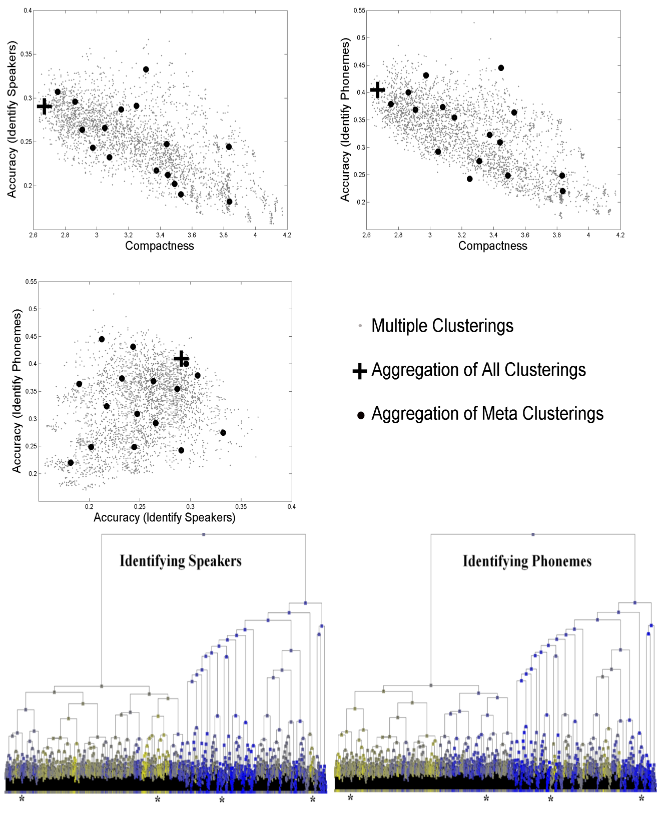


Figure 9. Accuracy vs. compactness scatter plots for the two accuracy measures (identifying speakers vs. recognizing phonemes) (1st row), scatter plot of the two accuracy measures (2nd row), meta level clustering dendrograms colored by accuracy in the two measures. Yellow indicates low accuracy. Blue is high accuracy. (The dendrorgams are much easier to interpret if viewed in color.)

38% in the compactness distribution, i.e. the most accurate clustering for this criterion is not one of the more compact clusterings. For the task of identifying phonemes, the most accurate clustering does not even occur in the top half of the most compact clusterings and falls at 53% in the compactness distribution.

In the scatter plot of the two accuracy measures (second row), there is a weak inverse correlation between the task of identifying speakers and identifying phonemes. The most accurate clustering for identifying speakers is generated from applying Zipf weighting with the shape parameter set at 0.25 to the before-PCA data. The most accurate clustering of identifying phonemes is generated from applying Zipf weighting with the shape parameter set at 1.25 to the PCA data. This confirms the need to sample a variety of Zipf weighting parameters and to explore PCA space.

For comparison, the scatter plots in Figure 9 show the consensus clustering found using the cluster aggregation method proposed in [24] (marked with a green "+" in the figures). This is the clustering that represents the consensus of *all* found clusterings. As expected, the consensus clustering is very compact (because less compact clusterings often disagree with each other, but the most compact clusterings often agree thus forming a strong consensus). Note, however, that the consensus clustering is not as compact as the most compact clusterings found by meta clustering, and also not very accurate on either the speaker identification or phoneme recognition tasks.

Again we use agglomerative clustering at the meta level to group similar clusterings. Figure 9 shows two copies of the same meta level clustering dendrogram (bottom row) colored by accuracy on the speaker identification and phoneme recognition tasks. Asterisks under the dendrograms indicate groups of clusterings that have significantly different accuracy for the two different tasks. Users may examine different clusterings by clicking on a clustering in the dendrogram, allowing users to zero-in on regions that appear promising.

If a user selects a cluster of clusterings in the meta level dendrogram (as opposed to a single base-level clustering), they can examine either the most central clustering in this branch of the dendrogram, or can examine a consensus clustering formed from the clusterings in the branch. The pink dots in the scatter plots represent consensus clusterings that a user might select.[1]

Table 2 shows the compactness for the 16 meta-level consensus clusterings (the pink dots in Figure 9), as well as the consensus clustering for *all* clusterings. The accuracy of the consensus clusterings on both the speaker identification and phoneme recognition task also are shown. The most accurate clusterings are shown in bold face. Once again note that the consensus clustering for all clusterings is not as accurate on either task as the clusterings found by meta clustering. Meta clustering is more likely to find clusterings of the data that might be useful to users for different tasks.

7. Related Works

[2] presents a very different algorithm for finding alternate clusterings of the data. In this approach a probability matrix that defines the likelihood of jumping from one point to another is used to generate a random walk. The transition probability is defined as a function of the Euclidean distance between each pair of points. The random walk allows particles to transition between instances according to the transition probability. Instead of clustering the data directly, distributions of the locations of the particles are clustered. Gaps in the eigen values indicate potentially good partitionings. At any random walk step with a local maximum eigen gap, the partition that maximizes this gap is reported. One of the ways in which this approach differs from meta clustering is that it uses a fixed method for measuring the distance between instances (euclidean distance). Also, the method only generates one clustering for each K. (All of the clusterings found with meta clustering in this paper are for a single fixed K.)

[1]We use the meta level clusterings found for $k = 16$ because examination of compactness vs. number of clusters for the meta-level agglomerative clustering indicated that there were 16 natural meta-level clusters.

A number of ensemble clustering methods improve performance by generating multiple clusterings. We mention only a few here. The cluster ensemble problem is formulated as a graph partitioning problem in [30] where the goal is to encode the clusterings into a graph and then partition it into K parts with the objective of minimizing the sum of the edges connecting those parts.

[11] proposes a different cluster ensemble method where both clusters and instances are modeled as vertices in a bipartite graph. Edges connect instances with clusters with a weight of zero or one depending on whether the instance does or does not belong to the cluster, thus capturing the similarity between instances and the similarity between clusters when producing the final clustering.

Another cluster ensemble method was proposed by [16] where the objective of the final clustering is to minimize the disagreement between all the clusterings and the final clustering. This final clustering is the one that agrees with most of the clusterings. In this framework, the clustering aggregation problem is mapped to the correlation clustering problem where we have objects and distances between every pair of them and the goal is to produce a partition that minimizes the sum of the distances inside each partition and maximizes the sum of the distances across different partitions.

Instead of partitioning, [24] used agglomerative clustering to produce the final clustering after generating a similarity matrix from many base-level clusterings. Cluster aggregation was formulated as a maximum likelihood estimation problem in [31] where the ensemble is modeled as a new set of features that describe the instances and the final clustering is produced by applying K-means while solving an EM problem. Linear programming was used in [4] to find the relation between the clusters in the different clusterings and the clusters of the final clustering. Simulated annealing and local search was used in [12] to find the final clustering.

The main difference between these ensemble methods and meta clustering is that most ensemble methods combine the clusterings they find into a one final clustering because their goal is to find a better, single, very compact clustering. Because the most compact clusterings are not necessarily the most useful clusterings, meta clustering does not attempt to combine different clusterings into one clustering. Instead, it groups different clusterings into meta clusters to allow users to select the clustering that is most useful for them.

8. Summary

Searching for the single best clustering may be inappropriate: the clustering that is “best” depends on how the clusters will be used and the data may need to be clustered in different ways for different uses. When clustering is used as a tool to help users understand the data, an appropriate clustering criterion cannot be defined in advance.

The standard approach has been for users to try to express in the distance metric the notion of similarity appropriate to the task at hand. This is awkward. Having to define the clustering distance metric, and then refine it when the clusters found are not what you want, is akin to having to modify your word processing software when the formatting it generates is not what you wanted. Few of the many potential users of clustering are adept at defining distance metrics and at understanding the (often subtle) implications a distance metric has on clustering. Even clustering researchers have difficulty modifying distance metrics to achieve better clusterings when the first metric they try does not work adequately.

In this paper we used auxiliary labels not available to clustering to measure clustering accuracy. We use accuracy as a proxy for users who have unspecified goals and intended uses of the clusterings. This allows an objective evaluation of meta clustering. Experiments with four test problems show that meta clustering is able to automatically find superior clusterings. Surprisingly, in these experiments we find only modest correlation between clustering compactness and clustering accuracy. The most accurate clusterings sometimes are not even in the most compact 50% of the clusterings. This reinforces our belief that searching for a single, optimal clustering is inappropriate when correct clustering criteria cannot be specified in advance. Instead, it is more productive to focus clustering on finding a large number of good, qualitatively different clusterings and allow users (or some form of post processing) to select the clusterings that appear to be best.

Experiments with a phoneme clustering problem showed that the clustering that is good for one criterion can be very suboptimal for another criterion. Different clusterings may be needed by different users. Meta clustering automatically found good (different) clusterings for each criterion. Experiments with a protein clustering problem provide a case study where meta clustering was able to improve clustering quality 60% above the best that could be achieved by human experts working with this data. Meta clustering achieved this improvement fully automatically in less than a day of computation. We believe the results demonstrate that meta clustering can make clustering more useful to non-specialists and will reduce the effort required to find excellent clusterings that are appropriate for the task at hand.

Acknowledgment

We thank John Rosenberg and Paul Hodor, collaborators from the University of Pittsburgh, for the Protein Data Set. Donna Bergmark from Cornell University provided the Bergmark text data set. Pedro Artigas, Anna Goldenberg, and Anton Likhodedov helped perform early experiments in meta clustering as part of a class project at CMU. Anton Likhodedov proposed the idea of applying PCA before random weighting in order to bias search toward weightings of the more important natural dimensions in the data. Casey Smith was supported by an NSF Fellowship. This work was supported by NSF CAREER Grant No. 0347318.

References

[1] M. J. Andrew Y. Ng and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002.

[2] A. Azran and Z. Ghahramani. A new approach to data driven clustering. In *Proceedings of the International Conference on Machine Learning*, 2006.

[3] L. Bottou and Y. Bengio. Convergence properties of the k-means algorithm. In *Advances in Neural Information Processing Systems*, 1995.

[4] C. Boulis and M. Ostendorf. Combining multiple clustering systems. In *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, 2004.

[5] P. Bradley and U. Fayyad. Refining initial points for k-means clustering. In *Proceedings of the International Conference on Machine Learning*, 1998.

[6] K. Christensen. Statistical and measurement tools.

[7] S. Cohen, G. Dror, and E. Ruppin. A feature selection method based on the shapley value. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 2005.

[8] T. Cox and M. Cox. *Multidimensional Scaling*. 1994.

[9] A. Demiriz, K. Bennet, and M. Embrechts. Semi-supervised clustering using genetic algorithms. In *Proceedings of Artificial Neural Networks In Engineering*, 1999.

[10] R. Duda and P. Hart. *Pattern classification and scene analysis*. 2001.

[11] X. Fern and C. Brodley. Solving cluster ensemble problems by bipartite graph partitioning. In *Proceedings of the International Conference on Machine Learning*, 2004.

[12] V. Filkov and S. Skiena. Integrating microarray data by concensus clustering. In *Proceedings of the International Conference on Tools with Artificial Intelligence*, 2003.

[13] E. Forgy. Cluster analysis of multivariate data: efficiency versus interpretability of classifications. 21:768–769, 1965.

[14] C. Furlanello, M. Serafini, S. Merler, and G. Jurman. Entropy-based gene ranking without selection bias for the predictive classification of microarray data. *BMC Bioinformatics*, 4:54, 2003.

[15] N. Geographic. Wild world terrestrial ecoregions.
[16] A. Gionis, H. Mannila, and P.Tsaparas. Clustering aggregation. In *Proceedings of the International Conference on Data Engineering*, 2005.
[17] L. Hubert and P. Arabie. Comparing partitions. 2:193–218, 1985.
[18] A. Jain and R. Dubes. *Algorithms for clustering data*. 1988.
[19] P. Kellam, X. Liu, N. Martin, C. Orengo, S. Swift, and A. Tucker. Comparing, contrasting and combining clusters in viral gene expression data. In *Proceedings of sixth workshop on Intelligent Data Analysis in Medicine and Pharmacology*, 2001.
[20] J. Kleinberg. An impossibility theorem for clustering. In *Proceedings of Neural Information Processing Systems*, 2002.
[21] G. N. Lance and W. T. Williams. A general theory of classificatory sorting strategies. i. hierarchical systems. *Computer Journal*, 9:373–380, 1967.
[22] G. McLachlan and K. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, NY, 1988.
[23] M. Meila and J. Shi. A random walks view of spectral segmentation. In *AISTATS*, 2001.
[24] S. Monti, P. Tamayo, J. Mesirov, and T. Golub. Consensus clustering: A resampling-based method for class discovery and visualization of gene expression microarray data. *Machine Learning*, 52:91–118, 2004.
[25] G. B. Mufti, P. Bertrand, and L. E. Moubarki. Determining the number of groups from measures of cluster stability. In *Proceedings of International Symposium on Applied Stochastic Models and Data Analysis*, 2005.
[26] D. J. Newman, S. Hettich, C. L. Blake, and C. J. Merz. UCI repository of machine learning databases, 1998.
[27] B. of Hexacorals. Environmental database.
[28] W. Rand. Objective criteria for the evaluation of clustering methods. *The American Statistical Association*, 6:846–850, 1971.
[29] N. Slonim and N. Tishby. Agglomerative information bottleneck. In *Proceedings of Neural Information Processing Systems*, 1999.
[30] A. Strehl and J. Ghosh. Cluster ensembles - a knowledge reuse framework for combining multiple partitions. *Machine Learning Research*, 3:583–417, 2001.
[31] A. Topchy, A. Jain, and W. Punch. A mixture model of clustering ensembles. In *Proceedings of the SIAM Conference on Data Mining*, 1999.
[32] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl. Constrained k-means clustering with background knowledge. In *Proceedings of the International Conference on Machine Learning*, 2001.
[33] O. Zamir, O. Etzioni, O. Madani, and R. Karp. Fast and intuitive clustering of web documents. In *Proceedings of the Knowledge Discovery and Data Mining*, 1997.

Appendix

The Zipf feature weighting method in coupling with Kmeans has performed better than other clustering methods that we evaluated. In this section we evaluate the performance of the Zipf feature weighting method in coupling spectral clustering. Here is the spectral method proposed by [1] in summary:

1. Compute the similarity matrix $S \in R^{n \times n}$ where

$$S_{ij} = \begin{cases} exp(-||x_i - x_j||/\sigma^2) & \text{if } i \neq j; \\ 0 & \text{if } i = j. \end{cases}$$

2. Compute the matrix

$$L = D^{\frac{1}{2}} S D^{\frac{1}{2}},$$

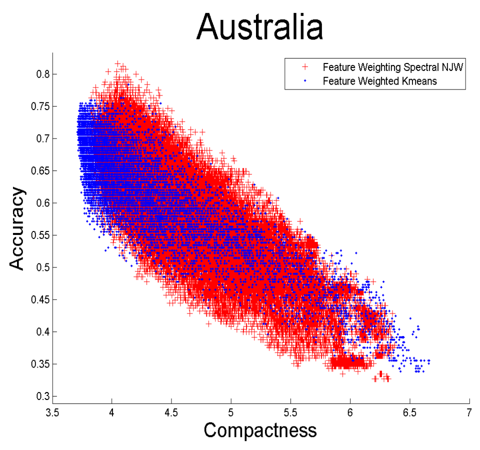
where D is the diagonal matrix whose $D_{ii} = \sum_j S_{i,j}$

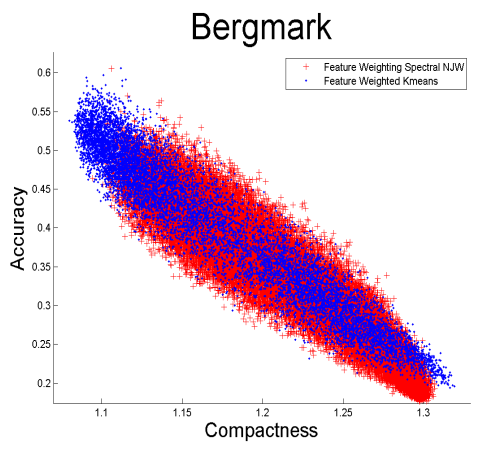
3. Let $1 = \mu_1 \geq \mu_2 \geq ... \geq \mu_K$ be the K largest eigenvalues of L and $u^1, u^2, ..., u^K$ the corresponding normalized eigenvectors. Form the matrix $U = [u^1 u^2 ... u^K]$

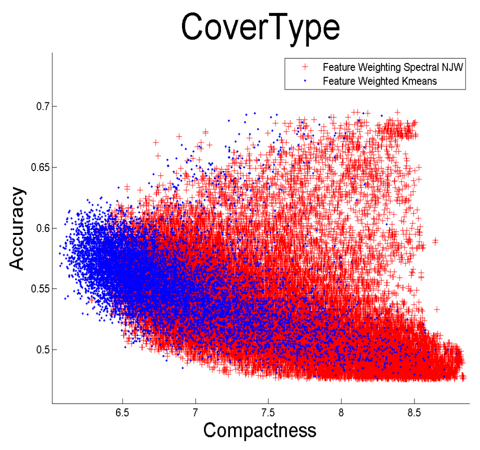
4. Form the matrix Y from U by re-normalizing each of U's rows to have unit length, $Y_{ij} = U_{ij} / \sqrt{\sum_j U_{ij}^2}$

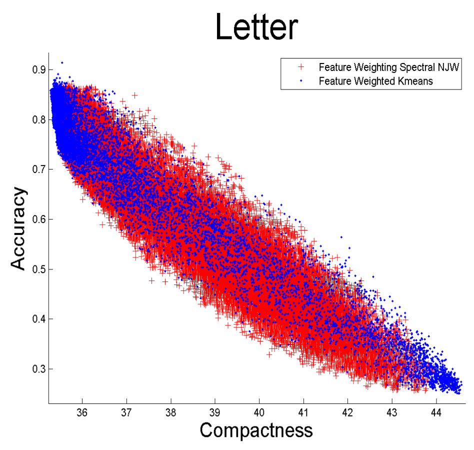
5. Treating each row of Y as a point in R^K, cluster them into K clusters vis Kmeans.

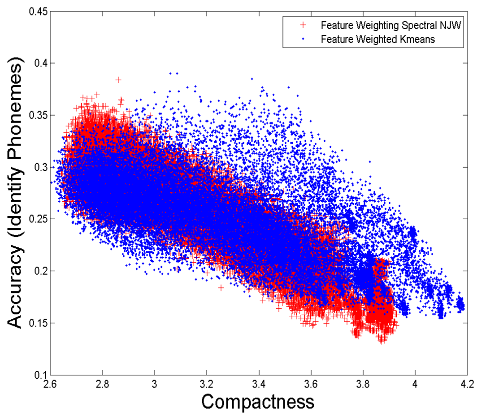
The Zipf weighting vector is applied to the original data set before we compute the similarity matrix S. For each Zipf weighting, we repeat the spectral clustering method with different values of $\sigma = \frac{\max\{||x_i - x_j||^2\}}{2^{i/4}}$ where $i = 0, 2, 4, ..., 32$.

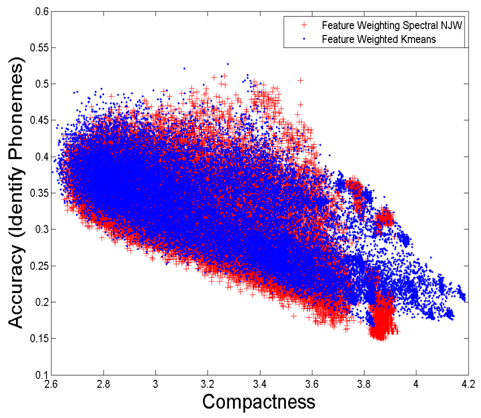

Australia
Feature Weighting Spectral NJW
Feature Weighted Kmeans
Accuracy
Compactness

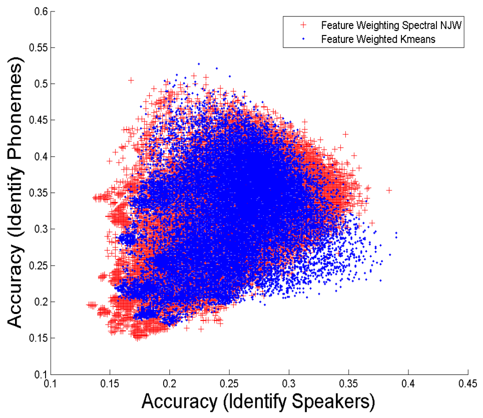

Bergmark
Feature Weighting Spectral NJW
Feature Weighted Kmeans
Accuracy
Compactness


CoverType
Feature Weighting Spectral NJW
Feature Weighted Kmeans
Accuracy
Compactness


Letter
Feature Weighting Spectral NJW
Feature Weighted Kmeans
Accuracy
Compactness


Feature Weighting Spectral NJW
Feature Weighted Kmeans
Accuracy (Identify Phonemes)
Compactness
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
2.6
2.8
3
3.2
3.4
3.6
3.8
4
4.2


Feature Weighting Spectral NJW
Feature Weighted Kmeans
Accuracy (Identify Phonemes)
Compactness
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
2.6
2.8
3
3.2
3.4
3.6
3.8
4
4.2


Feature Weighting Spectral NJW
Feature Weighted Kmeans
Accuracy (Identify Phonemes)
Accuracy (Identify Speakers)
0.6
0.55
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45