Hybrid Hierarchical Clustering: Forming a Tree From Multiple Views

Anjum Gupta
University of California, San Diego. 9500 Gilman Drive, La Jolla, CA 92093-0114
A3GUPTA@CS.UCSD.EDU

Sanjoy Dasgupta
University of California, San Diego. 9500 Gilman Drive, La Jolla, CA 92093-0114
DASGUPTA@CS.UCSD.EDU

Abstract

We propose an algorithm for forming a hierarchical clustering when multiple views of the data are available. Different views of the data may have different underlying distance measures which suggest different clusterings. In such cases, combining the views to get a good clustering of the data becomes a challenging task. We allow these different underlying distance measures to be arbitrary Bregman divergences (which includes squared-Euclidean and KL distance). We start by extending the average-linkage method of agglomerative hierarchical clustering (Ward’s method) to accommodate arbitrary Bregman distances. We then propose a method to combine multiple views, represented by different distance measures, into a single hierarchical clustering. For each binary split in this tree, we consider the various views (each of which suggests a clustering), and choose the one which gives the most significant reduction in cost. This method of interleaving the different views seems to work better than simply taking a linear combination of the distance measures, or concatenating the feature vectors of different views. We present some encouraging empirical results by generating such a hybrid tree for English phonemes.

1. Introduction

There has been a lot of recent machine learning research on exploiting multiple views of data. For instance, (Blum & Mitchell, 1998) notice that web pages can be viewed in two ways – by the words occurring in them, and by the words occurring in pages that point to them – and show how a certain type of conditional independence between these views can be exploited very effectively in semi-supervised learning. Likewise, (Collins & Singer, 1999) demonstrate a semi-supervised method for learning a named-entity classifier, using spelling and context as the two different views.

There has also been some encouraging work on using multiple views for unsupervised learning (eg. (Dhillon et al., 2003) and (Bickel & Scheffer, 2004)), in particular for clustering. It is natural to think that multiple views of the data should help yield better clusterings. However, there is a basic problem that needs to be resolved carefully. The various views may suggest rather different and incompatible clusterings of the data, especially if there is some independence between them. How can these different clusterings be reconciled?

We focus on hierarchical clusterings. These are popular tools for exploratory data analysis because they depict data at many level of granularity, and because there are simple algorithms for constructing them. In this paper, we propose a method for reconciling multiple views to generate a single hierarchical clustering.

Our model is as follows: there are \( n \) objects to be clustered. Each view corresponds to a different distance function on these objects. For instance, (Blum & Mitchell, 1998) notice that web pages...
represented by a different Bregman divergence.

We start by extending average-linkage agglomerative clustering (specifically, Ward’s method) to accommodate arbitrary Bregman divergences. Average-linkage typically assumes squared Euclidean norm as the underlying distance measure, and exploits special properties of this distance to substantially increase efficiency. We show that these same speedups can be realized for all Bregman divergences.

A straightforward way to accommodate multiple views would be to use a linear combination of their different distance measures. This approach runs into some basic problems. First of all, the different distances might at very different scales, and it might not be possible to make them comparable to one another by a simple linear transformation. Second, if the views represent very different information, then a linear combination of the two distances may simply serve to dampen or obscure the information in each. These intuitions are borne out in the experiments we conduct, in which linear combinations of the distance measures tend to destroy well-formed clusters that are present in individual views (based on just one distance function).

We propose a hybrid hierarchical clustering which is constructed top-down and in which each binary split is based upon a single view, the best view at that particular juncture. At each point in the tree construction, we have a certain cluster of objects that needs to be partitioned in two. We try out all the views; for each view, we determine a good split (into two clusters) using agglomerative clustering, and we note the reduction in cost due to that split. We choose the view that gives the biggest multiplicative decrease in cost. Thus, the tree keeps the best splits—the most significant clusterings—suggested by each view.

To try this out, we formed a hierarchical clustering for 39 phonemes, using data from the TIMIT database (Zue & Seneff, 1988). We used two views of each phoneme: a 39-dimensional vector in Euclidean space, the mean of the samples of that phoneme, where each speech sample is encoded using the standard mel-frequency cepstral coefficients. For the second view, we considered context information, specifically the distribution over the next phoneme. This is a probability vector in 39-dimensional space, with KL-divergence as the natural distance. The results were encouraging.

2. Bregman Divergences

Many of the most common families of probability distributions—such as Gaussian, Binomial, and Poisson—are exponential families. This formalism has turned out to be very powerful in statistics and machine learning because it is general enough to include many distributions of interest (another example: the distributions which factor over a specified undirected graph) while at the same time being specific enough that it implies all sorts of special properties.

It turns out that each exponential family has a natural distance measure associated with it. In the case of spherical Gaussians, it is perhaps obvious what this distance measure is: squared Euclidean distance, because the density at any given point is determined by its squared Euclidean distance from the mean.

Let’s look at another example. In the multinomial distribution, it can be checked that the density of a point depends on its KL-divergence from the mean. In a crucial sense, therefore, KL divergence is the natural distance measure of the multinomial. Notice that it is not a metric: it is not symmetric and does not satisfy the triangle inequality. However, as we will see, it is well-behaved in some ways and has a lot in common with squared Euclidean distance.

The various distance measures underlying different exponential families are collectively known as the Bregman divergences (Lafferty et al., 1997; Banerjee et al., 2004). We now give the standard formal definition of these divergences, which does not follow the intuition about exponential families but rather associates each divergence with a specific convex function.

**Definition** Let \( \phi : \mathcal{S} \to \mathbb{R} \) be a strictly convex function which is defined on a convex domain \( \mathcal{S} \subseteq \mathbb{R}^d \) and is differentiable on the interior of \( \mathcal{S} \). The Bregman distance \( D_\phi : \mathcal{S} \times \text{int}(\mathcal{S}) \to [0, \infty) \) is then defined by

\[
D_\phi(x, y) = \phi(x) - \phi(y) - \nabla \phi(y) \cdot (x - y).
\]  

Some examples: choosing \( \phi = \frac{1}{2} \|x\|^2 \) gives \( D_\phi(x, y) = \frac{1}{2} \|x - y\|^2 \), squared Euclidean distance.

\[
D_\phi(x, y) = \frac{1}{2} \|x\|^2 - \frac{1}{2} \|y\|^2 - y \cdot (x - y).
\]

Similarly, \( \phi(x) = \sum_{i=1}^d x_i \log x_i \) gives

\[
D_\phi(x, y) = \sum_{i=1}^d x_i \log \frac{x_i}{y_i} - \sum_i x_i + \sum_i y_i,
\] which is a generalization of KL-divergence (it reduces to the regular definition when \( x \) and \( y \) are probability measures and therefore sum to one).
2.1. Properties of Bregman divergences

Bregman divergences share a lot of the special properties of squared Euclidean distance. For instance, they satisfy a Pythagorean theorem (Lafferty et al., 1997). This makes it hopeful that many algorithms which seem expressly designed for squared Euclidean distance (and therefore for data which is, in a sense, Gaussian), such as k-means or average-linkage clustering, might be extendable to other Bregman divergences (that is, to other exponential families).

Recent work (Banerjee et al., 2004) has extended k-means to arbitrary Bregman divergences. This is possible due to certain properties that all Bregman divergences possess:

1. Given any set of points $S$, the single point $\mu$ which minimizes the aggregated Bregman distance

$$\sum_{x \in S} D_\phi(x, \mu)$$

is simply the mean of $S$, which we’ll denote $\mu_S$.

2. The additional cost incurred by choosing a different point $\mu \neq \mu_S$ as the center of cluster $S$ has a very simple form:

$$\sum_{x \in S} D_\phi(x, \mu) = \sum_{x \in S} D_\phi(x, \mu_S) + |S| \cdot D_\phi(\mu_S, \mu)$$

(4)

We will make extensive use of these properties.

3. Extending average-linkage clustering to Bregman divergences

There are several methods for average-linkage agglomerative clustering, of which Ward’s method is perhaps the most principled. Given $n$ data points, it starts by putting each point in a singleton cluster of its own. It then repeatedly merges the two closest clusters, until there is just one cluster containing all the points. The sequence of merges defines the hierarchical clustering tree; along the way we get $k$-clusterings (partitions into $k$ clusters) for all $k = 1, \ldots, n$. In Ward’s method, which is designed to use squared Euclidean distance, the distance between two clusters $S, T$ is the increase in k-means cost occasioned by merging them, in other words, $\text{cost}(S \cup T) - \text{cost}(S) - \text{cost}(T)$, where the cost of a set of points is defined as

$$\text{cost}(S) = \sum_{x \in S} ||x - \mu_S||^2.$$ 

In particular, when the algorithm has $k + 1$ clusters, and is deciding which pair to merge, it will choose the pair whose merger gives the smallest overall k-means cost. Thus Ward’s method strives to produce $k$-clusterings with small k-means cost, for all $k$.

This suggests how to extend the method to other Bregman divergences: simply change the cost function,

$$\text{cost}(S) = \sum_{x \in S} D_\phi(x, \mu_S).$$

Notice that this makes sense because of property 1 above. The resulting algorithm is once again trying to minimize k-means cost, but for our more general distance functions.

This is straightforward enough, but more work is needed to make this new algorithm practical. In Ward’s method, the properties of Euclidean distance are used to compute the cost of candidate mergers,

$$\text{Ward}_{Euc}(S, T) = \text{cost}(S \cup T) - \text{cost}(S) - \text{cost}(T),$$

very quickly. There is no need to actually sum over these three clusters; instead, the expression reduces to

$$\text{Ward}_{Euc}(S, T) = \frac{|S||T|}{|S| + |T|} \cdot ||\mu_S - \mu_T||^2,$$

which is very quick, assuming that the means and sizes of clusters are kept available. Notice also that $\mu_{S \cup T}$ and $|S \cup T|$ can easily be computed from the means and sizes of $S, T$.

We now see that a similar simplification is possible for any Bregman divergence, and thus we can handle any of these distance functions without any additional time complexity. More precisely:

**Lemma** For any Bregman divergence $D_\phi$, the cost of merging two clusters $S, T$ is:

$$\text{Ward}_\phi(S, T) = \text{cost}(S \cup T) - \text{cost}(S) - \text{cost}(T) = |S| \phi(\mu_S) + |T| \phi(\mu_T) - (|S| + |T|) \phi(\mu_{S \cup T}).$$

**Proof:** For any Bregman divergence $D_\phi$, the cost of merging two clusters $S, T$ is:

$$\text{Cost}(S \cup T) - \text{Cost}(S) - \text{Cost}(T) = \sum_{x \in S \cup T} D_\phi(x, \mu_{S \cup T}) - \sum_{x \in S} D_\phi(x, \mu_S) - \sum_{x \in T} D_\phi(x, \mu_T)$$

The term $\sum_{x \in S \cup T} D_\phi(x, \mu_{S \cup T})$ represents the cost of merging $S$ and $T$. The term $\sum_{x \in S} D_\phi(x, \mu_S)$ represents the new cost of cluster $S$ after merging. Similarly, $\sum_{x \in T} D_\phi(x, \mu_T)$ represents the new cost of cluster $T$. Subtracting these costs from the total cost of merging $S$ and $T$ gives the original cost of $S$ and $T$.
4. Multiple views

Often we may have different set of features, obtained in different ways, and giving different type of information about the data we are trying to cluster. For example, we can obtain three different views of a web page, the first being the words in the web page itself, the second being the words in the web pages pointing to it, and third being some other statistical information about the page such as the size, number of times it is accessed etc. Each of the views is a useful source of information about the web page, and together they should be able to yield a better clustering than we could get from one view alone, but it is not obvious how to combine them.

Using equation (4), we get

\[ = \sum_{x \in S \cup T} D_\phi(x, \mu_{S \cup T}) \]

\[ - \sum_{x \in S} D_\phi(x, \mu_{S \cup T}) + |S| \cdot D_\phi(\mu_s, \mu_{S \cup T}) \]

\[ - \sum_{x \in T} D_\phi(x, \mu_{S \cup T}) + |T| \cdot D_\phi(\mu_t, \mu_{S \cup T}) \]

\[ = |S| \cdot \phi(\mu_s) + |T| \cdot \phi(\mu_t) - (|S| + |T|) \cdot \phi(\mu_{S \cup T}) \]

\[ - |S| \cdot \mu_s \cdot \phi'(\mu_{S \cup T}) + |S| \cdot \mu_{S \cup T} \cdot \phi'(\mu_{S \cup T}) \]

\[ - |T| \cdot \mu_t \cdot \phi'(\mu_{S \cup T}) + |T| \cdot \mu_{S \cup T} \cdot \phi'(\mu_{S \cup T}) \]

\[ = |S| \cdot \phi(\mu_s) + |T| \cdot \phi(\mu_t) - (|S| + |T|) \phi(\mu_{S \cup T}) \]

So we get,

\[ Cost(S \cup T) - Cost(S) - Cost(T) \]

\[ = |S| \cdot \phi(\mu_s) + |T| \cdot \phi(\mu_t) - (|S| + |T|) \phi(\mu_{S \cup T}) \]

The final expression above can be evaluated quickly. For example, for KL distance, we get

\[ Ward_{KL}(S, T) = |S| \cdot \mu_s \cdot \log(\mu_s) + |T| \cdot \mu_t \cdot \log(\mu_t) \]

\[ - (|S| + |T|) \mu_{S \cup T} \cdot \log(\mu_{S \cup T}) \]

(where the logarithms are taken coordinatewise).

4. Hybrid clustering algorithm

Input: A set of \( n \) points given as two views (representations) \( X \) and \( Y \), each in a space with an underlying Bregman divergence.

1. If \( n = 1 \) put the single point in a leaf and return.
2. Apply the generalized Ward’s method to \( X \) and \( Y \) in turn, and in each case retrieve the 2-clustering. Call these \( C_{x1}, C_{x2} \) (for \( X \)) and \( C_{y1}, C_{y2} \) (for \( Y \)).
3. Choose the “better” of the two splits, to divide \( X \), \( Y \) into two clusters, \( [X_1, Y_1] \) and \( [X_2, Y_2] \).
4. Create a tree node for this split.
5. Recursively handle \( [X_1, Y_1] \) and \( [X_2, Y_2] \).
6. Return tree.

4.1. Combining multiple views: first try

Perhaps the most obvious approach to accommodating multiple views is to concatenate the feature vectors corresponding to the different representations, or more generally, to use a linear combination of their different underlying distance measures. This approach is problematic on two fronts. First, the various distance measures might be incomparable (as with Euclidean distance and KL divergence), making it somewhat absurd to form linear combinations of them. Second, if the views are orthogonal and suggest different clusterings, then in the linear combination this structure might get obscured. In fact, in our experiments we see when features are naively combined by such methods, we lose some of the good clusters clearly present in visible in the individual views.

4.2. A hybrid approach

When combining multiple views, we want to preserve cluster structures that are strongly suggested by the individual views. The idea is that if there was a strong separation between the data points in one of views, that separation should not be lost while combining the information from other views. We propose building a hierarchical tree in a top-down fashion that uses the best view available at each split point in the tree. This hybrid algorithm is outlined in the figure above.

To choose the best view for a given split, the algorithm computes the 2-clusterings suggested by all the different views, and picks the best one. How should this be defined? Intuitively, we want to pick the view that provides the most well-separated clustering, that is, the largest reduction in the cost. We have to be
When there are three true clusters, K-means could split the third clusters into two pieces (shown in A), whereas 2-clusters generated using Ward’s method is more likely to keep the clusters intact (shown in B).

Instead, we use ratios. We measure the goodness of a particular 2-clustering (binary split suggested by a particular view) by the ratio of its cost to the cost of the single combined cluster. That is, the goodness of a split of cluster \(X\) into \([X_1, X_2]\) is

\[
\frac{\text{cost}(X)}{\text{cost}(X_1) + \text{cost}(X_2)}.
\]

This particular measure (or rather its reciprocal) is equivalent to the (2-norm) Davies-Bouldin index (Davies & Bouldin, 1979) of similarity between two clusters.

When we are generating a 2-clustering from a particular view, we use agglomerative clustering (Ward’s method) rather than \(k\)-means, even though the latter is simpler and quicker. Since at every step we split the data into exactly two clusters, using \(k\)-means could give us a bad division in case the data had three true clusters: it is more likely to split the third cluster into two pieces than Ward’s method, which would likely instead split the three clusters by putting the two closer ones together (Figure 1). This approach adds a \(O(n^2)\) complexity to our algorithm so may not be used for larger data sets.

5. Experiments

Our experimental results are for the 39 English language phonemes. We used two views of each phoneme, that were available in the TIMIT data set. The first view was intended to represent the speech signal itself, and consisted of a 39-dimensional vector in Euclidean space. To form this vector for a given phoneme, we looked at all utterances of that phoneme in the data set and transformed each utterance into a sequence of 39-dimensional vectors consisting of mel-frequency cepstral coefficients. This is a standard representation for speech recognizers. We picked one vector of coefficients from roughly the middle of each utterance, and then averaged these (over utterances) to get a single 39-dimensional vector for the phoneme. We thought of this vector as residing in Euclidean space since this is implicitly the distance measure used on this data by most speech recognizers.

The second view consisted of context information, specifically about the next phoneme. For each phoneme, we constructed a 39-dimensional vector representing transition probabilities to other phonemes (the \(i^{th}\) entry was the chance that this particular phoneme would be followed by the \(i^{th}\) phoneme). We used Laplace smoothing to avoid zero probability values. For this view, we used KL-divergence since it is a natural distance to use with probability measures. (It is purely a coincidence that both views of the phonemes have the same dimensionality.)

A reference hierarchical clustering already exists for phonemes, and is shown in Figure 2, copied over from (Rabiner & Juang, 1993). Each family of phonemes has been assigned a color to make it easier to compare the various trees generated by experiments.
5.1. Pure hierarchical clusterings

We first generated hierarchical clusterings based on individual views. The first of these, using Euclidean distance on the speech signal representation, is shown in Figure 3. The second, using KL divergence on the context representation, is shown in Figure 4. In the trees produced by these individual views, there are a few clusters that partially match the clusters suggested by the reference classification in Figure 2. The first view does a good job of separating stops, fricatives and affricatives from the rest, although it is not good at distinguishing between these three groups. The second view is overall less competent, although it does a better job of distinguishing stops, fricatives, and affricatives. However, each view by itself is quite far from the reference clustering.

5.2. Way to combine multiple views

The trees generated by each individual view with their corresponding distance measures seem to complement each other by showing part of the whole picture. This motivated us to use both measures together to generate one consolidated tree. We first tried the obvious trick of combining the two distance measures by using a linear combination of the two distance measures. The result was still a Bregman divergence and thus amenable to our generalized agglomerative clustering scheme. The tree in Figure 5 is based upon a particular linear combination that tries to partially account for the different scales of the two distance measures.

Figure 5 doesn’t improve on the clustering given by each individual view separately, and in fact demolishes some clusters suggested by KL distance.

5.3. Better way to combine the multiple views

Finally, we combined the two views into a hybrid tree, using the proposed algorithm. The result is in Figure 7. The hybrid tree manages to preserve the separations that were strongly suggested by each view, to unite the good points of each. A good separation of nasals, vowels, stops and fricatives from the first view and a better separation of stops and affricatives due to the second view is clearly present in the hybrid tree. The height of the tree nodes in the hybrid tree do not correspond to the closeness of the points under that node, since the algorithm works in a recursive way, the entire left tree is generated before the right tree.

Figure 6, shows the histogram of the similarity measure between the two clusters at every split in the hierarchy. Notice that this is the inverse of the “goodness of split”. The lower the similarity between clusters, the better the split. We observe that on the whole, the splits found by our hybrid algorithm lead to less similar pairs of clusters than those found by the regular Ward’s algorithm using a single view. This corroborates our intuition that the significant separations in either view should make it into the hybrid view.
Figure 5. Hierarchical clustering based on a linear combination of the two distance functions. The tree looks very similar to the one built only in Euclidean space.

Figure 6. Histogram of similarity measures between the two clusters at every split in the hierarchy (smaller values are better). The distribution of these similarity values is noticeably better for the hybrid algorithm than for the other two.

Figure 7. Tree formed by using both views and using the hybrid clustering algorithm. It preserves the separations that were clearly present in each of the individual views.

6. Future Work

We seem to have found a way to effectively exploit multiple views in hierarchical clustering. We still face the issue of how to quantitatively assess the extent of the benefit. A related approach to multiview learning is presented in (Bickel & Scheffer, 2004). They evaluate their clusterings by computing the entropy of the clusters given the true classification. Our approach of using the histogram of inverse goodness-of-split values is more subjective but better fits the unsupervised model.

We also plan to test our hybrid algorithm on different and larger data sets, such as the WebKB data recently provided to us by Steffen Bickel, for which we are most grateful.

References


