Hierarchical Clustering

36-350, Data Mining

17 September 2008

1 Hierarchical clustering

The k-means algorithm gives us what's sometimes called a *simple* or *flat* partition, because it just gives us a single set of clusters, with no particular organization or structure within them. But it could easily be the case that some clusters could, themselves, be closely related to other clusters, and more distantly related to others. (If we are clustering images, we might want not just to have a cluster of flowers, but roses and marigolds within that. Or, if we're clustering patient medical records, we might want "respiratory complaints" as a super-cluster of "pneumonia", "influenza", "SARS", "miscellaneous sniffling".) So sometimes we want a **hierarchical** clustering, which is depicted by a **tree** or **dendrogram**.

1.1 Ward's method

Ward's method is another algorithm for finding a partition with small sum of squares. Instead of starting with a large sum of squares and reducing it, you start with a small sum of squares (by using lots of clusters) and then increasing it.

1. Start with each point in a cluster by itself (sum of squares = 0).
2. Merge two clusters, in order to produce the smallest increase in the sum of squares (the smallest merging cost).
3. Keep merging until you've reached k clusters.

The **merging cost** is the increase in sum of squares when you merge two clusters (A and B, say), and has a simple formula:

$$\begin{aligned}
\Delta(A, B) &= \sum_{i \in A \cup B} \|x_i - m_{A \cup B}\|^2 - \sum_{i \in A} \|x_i - m_A\|^2 - \sum_{i \in C} \|x_i - m_B\|^2 \\
&= \frac{n_A n_B}{n_A + n_B} \|m_A - m_B\|^2
\end{aligned}$$

(You may recall having seen a similar formula for t-tests of the difference in means.) Note that Ward's method does not rely on a random starting guess, so its answer is unique.

The Δ formula tells us that there is a trade-off between separation and balance. If clusters are equally far apart (separated), it's better to merge the smaller ones. This means that Ward's algorithm will sometimes merge clusters which are further apart but smaller.

The k-means algorithm gives no guidance about what k should be. Ward's algorithm, on the other hand, can give us a hint through the merging cost. If the cost of merging increases a lot, it's probably going too far, so a rule of thumb is to keep reducing k until the cost jumps, and then use the k right before the jump.

The partitions produced by Ward's method are **nested**: the partition of size k is contained within the partition of size $k + 1$. It is also a *greedy* algorithm: it always picks the merge whose immediate cost is smallest, without worrying about whetheer this will impose bigger costs later on. These two properties mean that Ward's method generally does not produce a sum-of-squares as small as k-means. If that really bothers you, run k-means starting from the Ward's method solution.

Figure 1 shows what Ward's method does with the flower/tiger/ocean images (represented, as usual, by bags of colors). This makes one clear mistake (it thinks flower5 goes with the tigers rather than the other flowers), but otherwise looks reasonable. The merging costs (Figure 1.1 suggest that there are 3 clusters (or perhaps 6 or 8).

The sum of squares measures distance equally in all directions, so it wants the clusters to be round. This is not always very sensible (see Figure 2).

1.2 The Single-link algorithm

Single-link clustering can handle any cluster shape:

1. Start with each point in a cluster by itself (sum of squares = 0).
2. Merge the two clusters with smallest gap (distance between the two closest points)
3. Keep merging until you've reached k clusters.

It's called "single link" because it will merge clusters so long as *any* two points in them are close (i.e., there is one link).

This algorithm only wants separation, and doesn't care about compactness or balance. This can lead to new problems, as shown in Figure 3.

Question: how is the single-link method like nearest neighbor classification? If k-means is the like the unsupervised version of the prototype method, what would the unsupervised version of nearest neighbors be like?

2 How Many Clusters?

This is the big question. The heuristic for merging costs is just that, a heuristic. One reason you should be *intensely* skeptical of clustering results — including

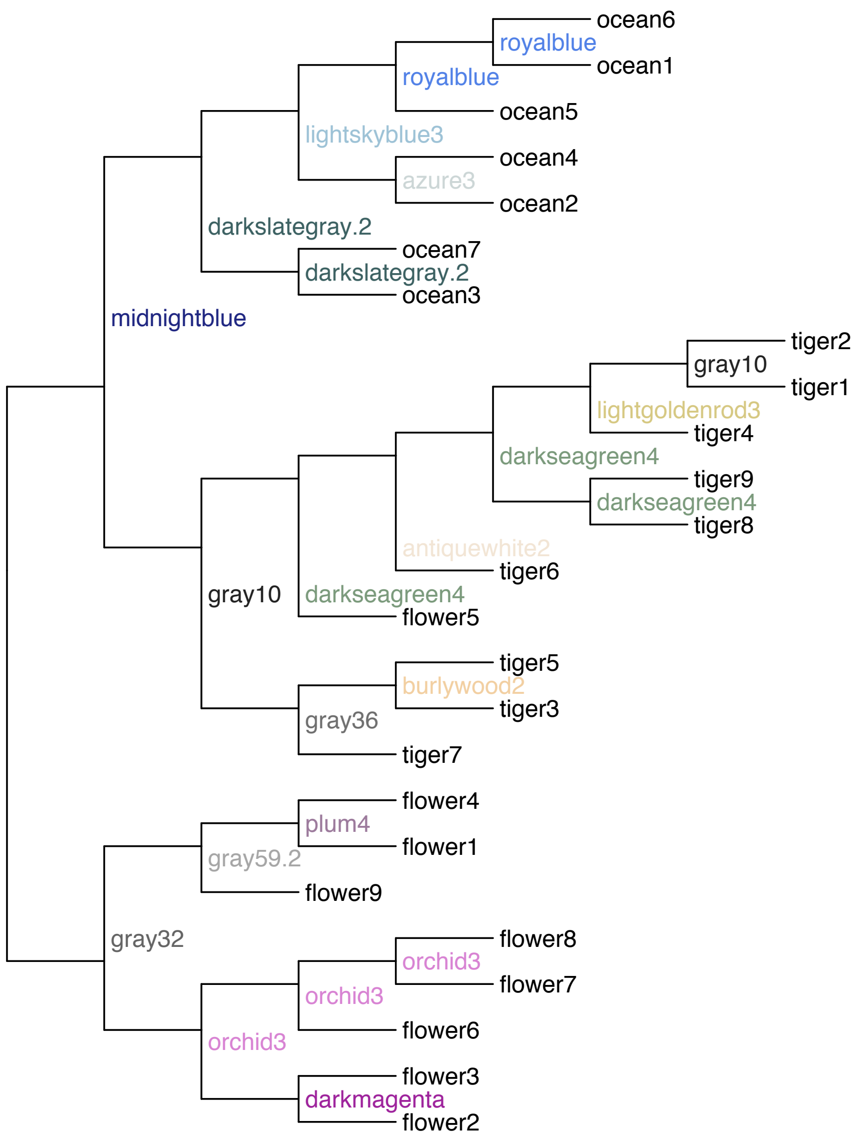
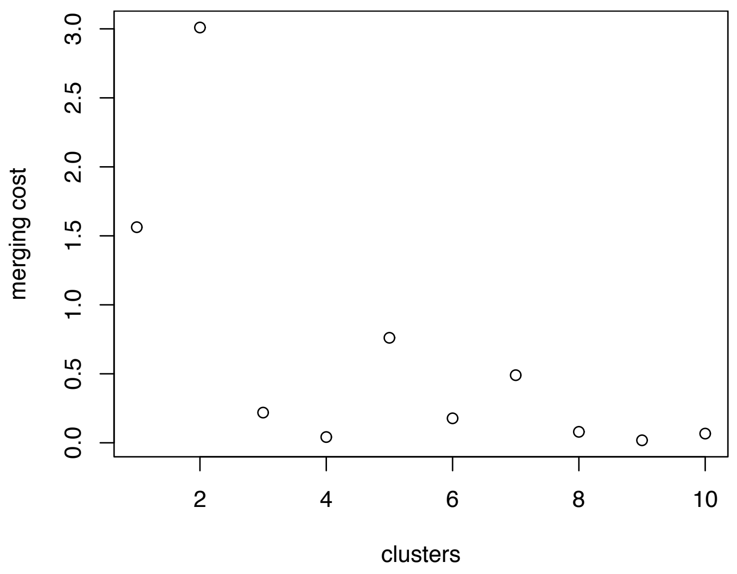


Figure 1: Using Ward's method to form a hierarchical clustering of the flower/tiger/ocean pictures. Each cluster is labeled with the name of a color which was common to both sub-groups but rare in the rest of the data — i.e. the color is informative about sub-cluster membership.



your own! — is that there is currently very little theory about how to find the right number of clusters. It's not even completely clear what "the right number of clusters" means!

At one extreme, we could put every data point in its own cluster. Then the clusters would be perfectly informative about the data. The downsides to this are that (1) it makes cluster analysis pointless, and (2) the clusters will not help us with *new* data-points. At the other extreme we could always decide that all our data points really form one cluster, which might look weirdly irregular and have an oddly lumpy distribution on it, at least as we've chosen to represent it. This is sometimes even right, or at least *sounds* right!

Some people try to get around this by modifying the objective function used in clustering, to add some penalty per cluster, or per level of hierarchy, etc. The idea is to encourage **parsimony**: "everything should be as simple as possible, but no simpler". The difficulty is that these penalties are generally things pull out of the air (to be polite about it), and there is no reason to think that they really do give us good clusters in general.

The only real hope here, it seems to me, is to start from the fact that having too many clusters *generalizes badly.* If the data really do fall into a few clusters, then new data from the same source should fall into the same clusters. The right number of clusters then would be the one which generalizes best to new observations. There are multiple ways we could measure this:

- how much do cluster centers or boundaries move if we re-run clustering on new data
- how much do cluster assignments change, ditto

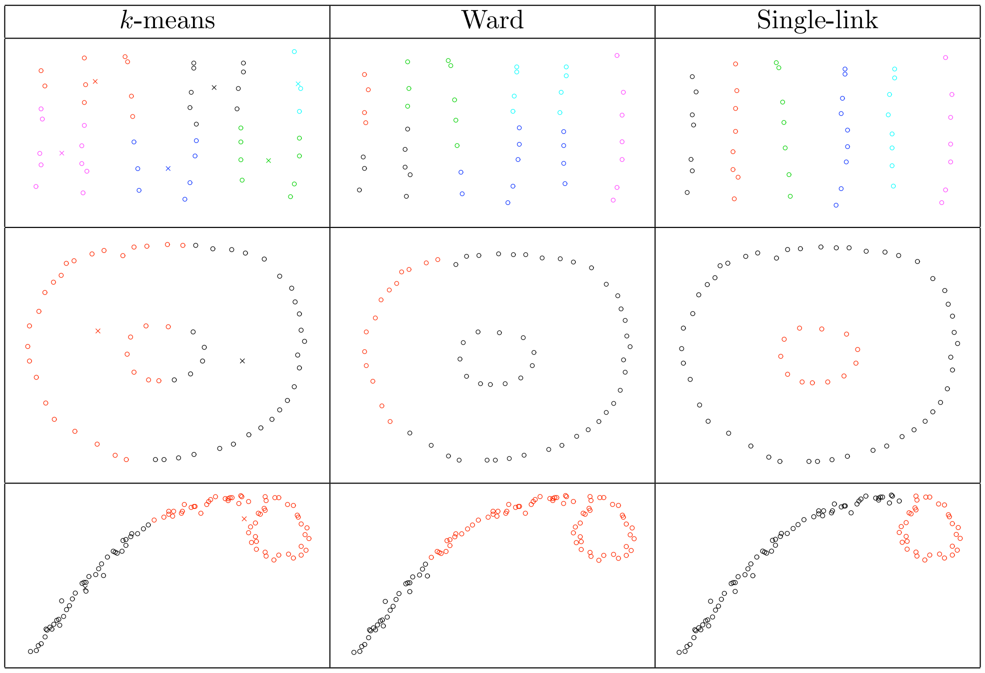


Figure 2: Some clustering problems where the single-link method does better than k-means or Ward's method. (In the k-means plots, the cluster means are marked $\times$.)

- how big are the sum-of-squares, when assigning new data to the old clusters?

Of course, to do any of these things we would need new data. Instead of waiting to get new data, we could fake it by re-using some of the data we already have. The general idea of **cross-validation** is to randomly divide the data into a training set and a testing set, and use performance on the testing set as a proxy for performance on genuinely new data. Repeated many times, over many random divisions (ten-fold cross-validation, with 90% training and 10% testing, is common), this often gives a reasonable sense of how sensitive our results are to sampling accidents, and how well they generalize to new data from the same source.

What we would really like to do is look at how well the old clusters *predict* new data. (This is what the sum-of-squares notion points towards.) The difficulty with doing that is none of the clustering algorithms we have looked at so far is really predictive. (Even k-means doesn't really predict the average distance from points in a cluster from its center.) To put it another way, none of our clustering procedures gives us a model which would let us generate or simulate new data points ("synthetic data" or "surrogate data"). We will later see various forms of cluster analysis which *do* use generative models, and while

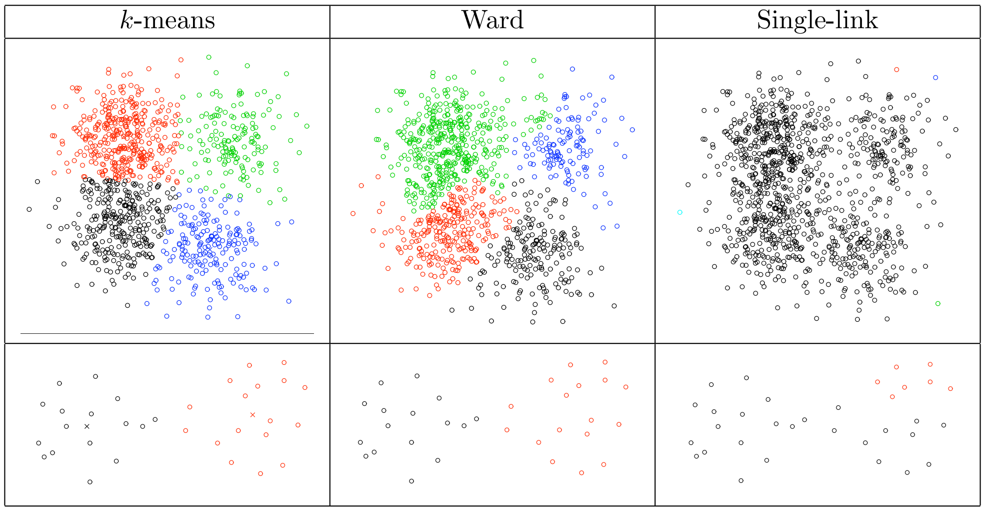


Figure 3: Some cases where k-means or Ward's algorithm does better than the single-link method.

they are more work it is actually clearer what they mean, and how to tell if they are working.

A distinct but related question from how many clusters we should use is how confident we should be about the statement "these two data points belong to the same cluster". A first cut would be to do cross-validation and see how often the points in question wind up in the same cluster. A more refined approach doesn't seem to be possible unless you add some assumptions about the data-generating process. Prof. Nugent has worked on the issue of "clustering with confidence", and I encourage those of you taking 36-401 from her to ask her about it.