36-350: Data Mining

Handout 6
September 14, 2003

Partitioning data into clusters

A **partition** is a division of the dataset into **clusters** which are ideally **compact**, **separated**, and **balanced**. Finding a good partition requires a good mathematical definition and trade-off between these qualities. Like similarity search, partitioning hinges on a good definition of distance between data objects.

We start with an algorithm which produces good results, and then look at how it achieves compactness, separation, and balance.

The **K-means** algorithm:

1. Guess the number of clusters, K
2. Guess the location of cluster means
3. Assign each point to the nearest mean
4. Compute new means
5. Iterate back to (3)

K-means tries to minimize a **sum-of-squares** objective:

$$\begin{aligned} SS &= \sum_c \sum_{i \in R_c} ||x_i - m_c||^2 \\ m_c &= \frac{1}{|R_c|} \sum_{i \in R_c} x_i \end{aligned}$$

m_c is the mean for cluster c. This formula is the variance of the cluster times the size of the cluster, summed over all clusters. Minimizing it clearly wants clusters to be compact. It also wants balance, because big clusters cost more than small ones with the same variance.

Each step of K-means reduces the sum-of-squares. The sum-of-squares is always positive. Therefore K-means must eventually stop. However, it may not stop at the best solution.

K-means is a **local search** algorithm: it makes small changes to the solution that improve the objective. Local search is also called **hill-climbing**, by analogy to an impatient hiker who tries to find the highest peak by always walking uphill. This strategy can stop at **local optima**, where the criterion is not optimized but no improvement is possible by making small changes. Which local minimum you stop at depends on where you started. Hence K-means gives different results depending on how you make the initial guess. A typical initial guess is to pick K data points at random to be the means.

Ward's method for hierarchical clustering

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, and has a simple formula:

$$\begin{aligned}\Delta &= \sum_i ||x_i - \bar{x}||^2 - \sum_{i \in A} ||x_i - \bar{a}||^2 - \sum_{i \in B} ||x_i - \bar{b}||^2 \\ &= \frac{n_a n_b}{n_a + n_b} ||\bar{a} - \bar{b}||^2\end{aligned}$$

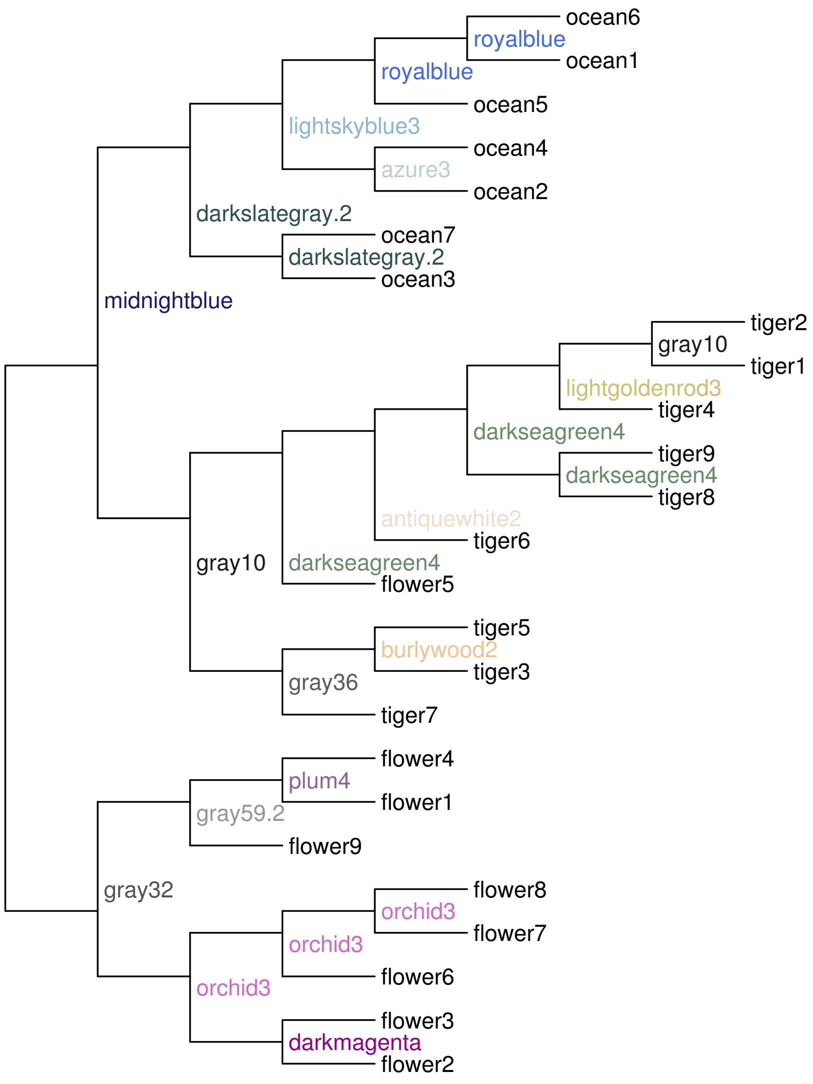
This formula reveals the tradeoff between separation and balance. For clusters which are the same distance away, it is better to merge the smaller ones. Consider the case where one cluster is just a single point: it wants to join not just the closest cluster but one with a small number of points already in it.

The merging cost provides a suggestion of the number of clusters in the data. If the cost jumps, you've probably merged too far. So a good number of clusters is the number before a jump in the merging cost.

The partitions produced by Ward's method are **nested**: the partition of size K is contained within the partition of size $K + 1$. Ward's method also does not do local search. These two properties mean that Ward's method generally does not produce a sum-of-squares as small as K-means. However, we can run the K-means search starting from the Ward's method solution, to get a competitive sum-of-squares. Note that Ward's method does not rely on a random starting guess, so its answer is unique.

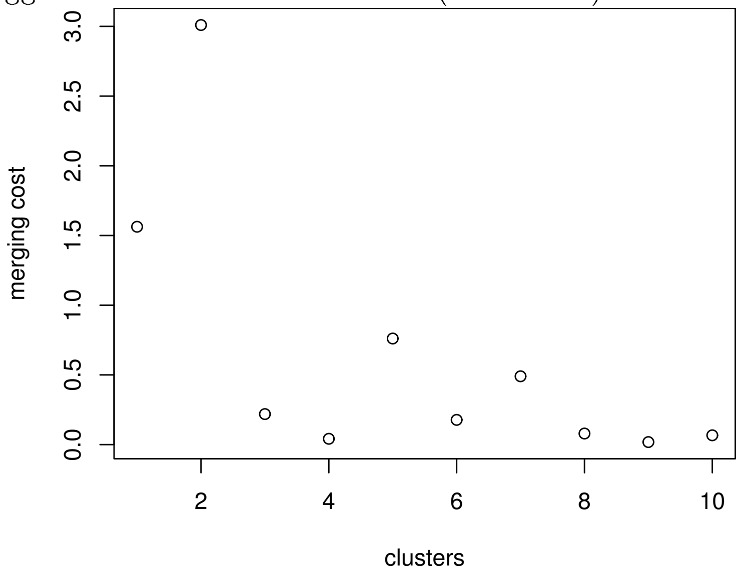
Ward's method produces not just a single partition but an entire hierarchy of clusters, which is usually more informative and doesn't require a choice of K.

Ward's method on the flower/tiger/ocean images:

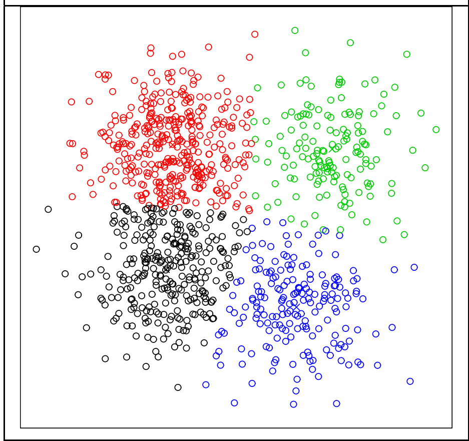
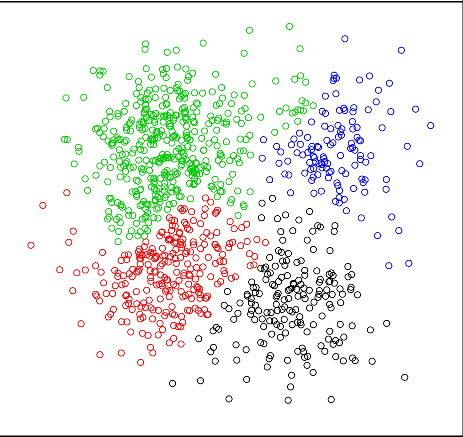
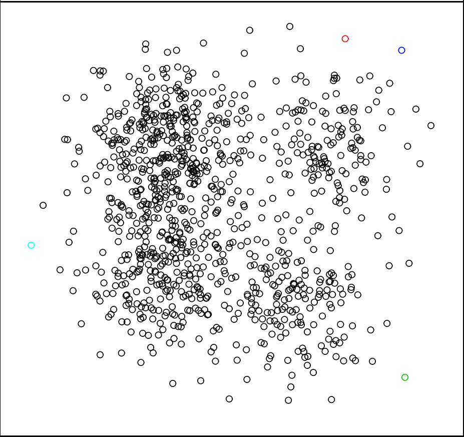
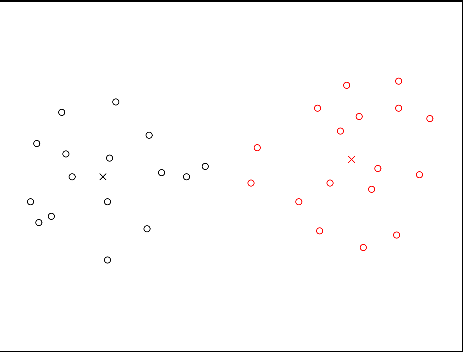
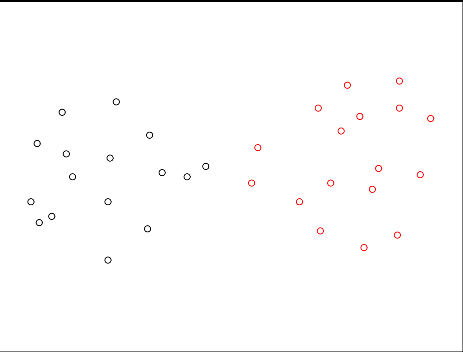
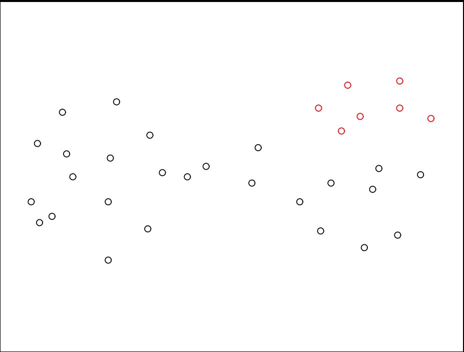


Only one mistake is made (flower5). The clustering was computed using all colors, but to enhance interpretation, each cluster has been labeled with a "joining" color (one which both subgroups have in common but is rare in the rest of the data). This is similar to finding colors with high information content.

The merging cost suggests that there are 3 clusters (also 6 or 8):



Other examples: (x's are cluster means)

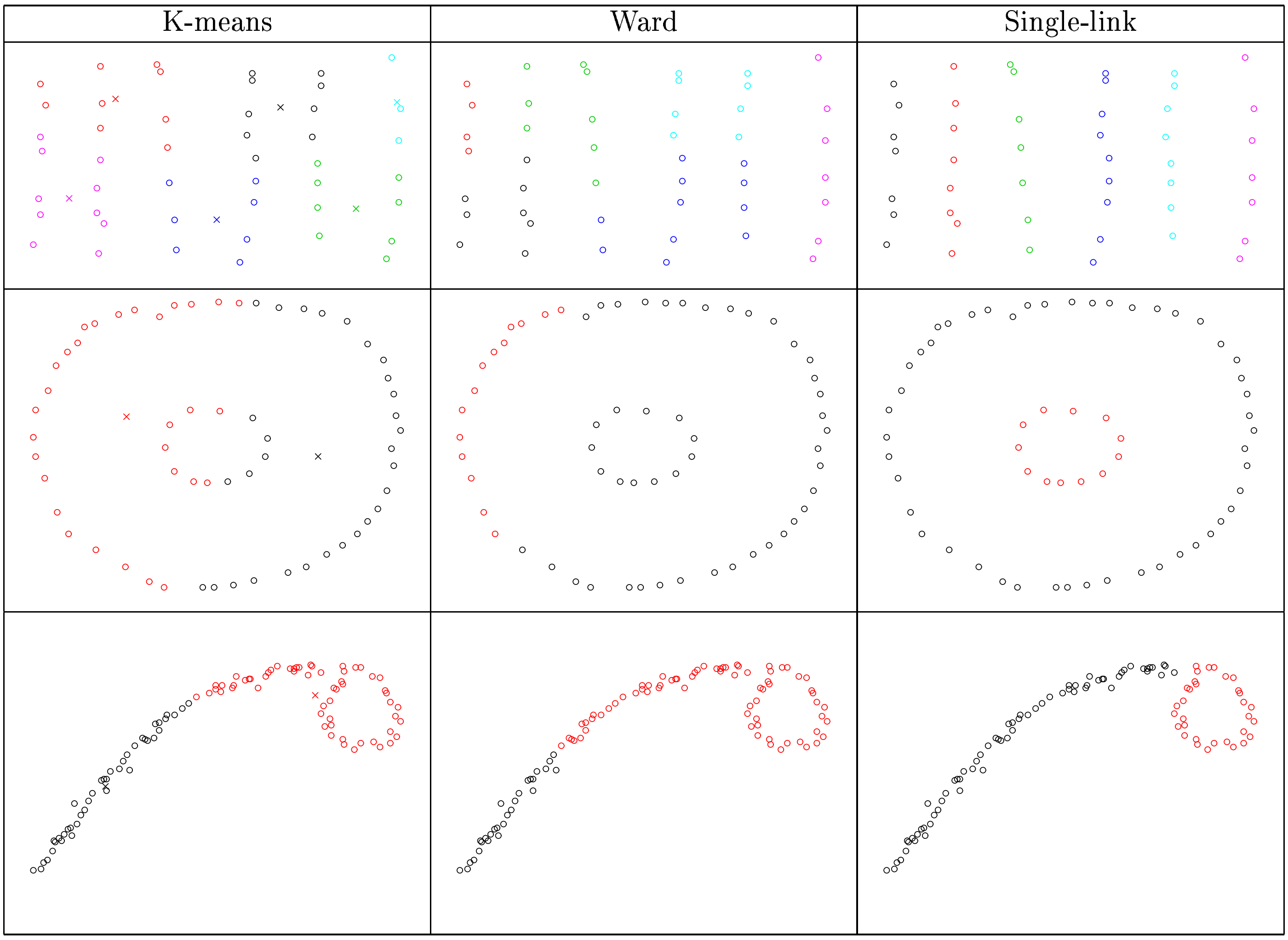
K-means	Ward	Single-link
		
		

The sum of squares measures distance equally in all directions, so it wants the clusters to be round. This can sometimes be a disadvantage.

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.

This algorithm only wants separation, and doesn't care about compactness or balance. This can lead to new problems, as shown below. Current research focuses on algorithms, such the **Jarvis&Patrick** algorithm, which blend the advantages of single-link and Ward's method.



References

[1] David Hand, Heikki Mannila, and Padhraic Smyth. *Principles of Data Mining*, Section 9.3. MIT Press, 2001.

[2] Richard O. Duda, Peter E. Hart, David G. Stork. *Pattern Classification.* Wiley, 2000.

[3] Clustan clustering software. Case study and critique of K-means.
`http://www.clustan.com/k-means_critique.html`