COMPSCI 589
Lecture 15: Hierarchical Clustering

Benjamin M. Marlin

College of Information and Computer Sciences
University of Massachusetts Amherst

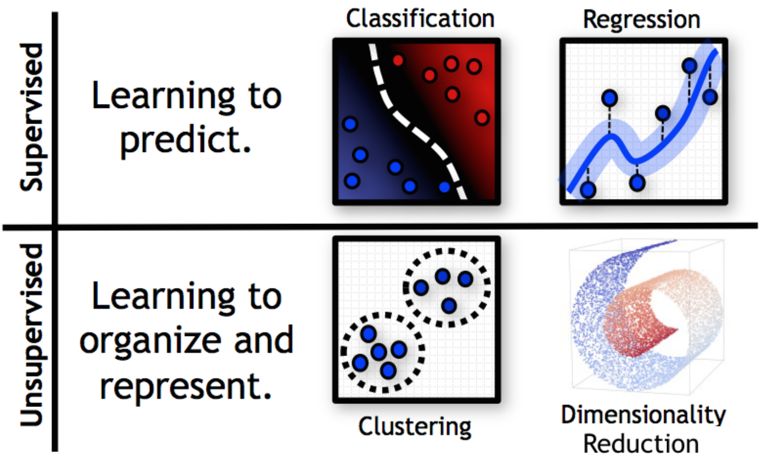
Slides by Benjamin M. Marlin (marlin@cs.umass.edu).
Created with support from National Science Foundation Award# IIS-1350522.

Views on Machine Learning

Mitchell (1997): "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

Substitute "training data D" for "experience E."

Machine Learning Tasks



The Classification Task

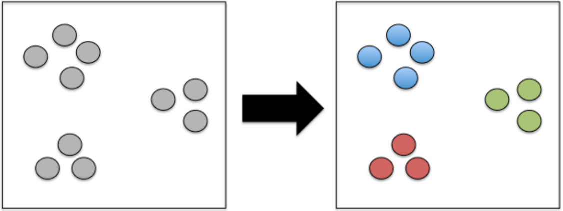
Definition: The Classification Task

Given a feature vector $\mathbf{x} \in \mathbb{R}^D$ that describes an object that belongs to one of C classes from the set $\mathcal{Y}$, predict which class the object belongs to.

The Clustering Task

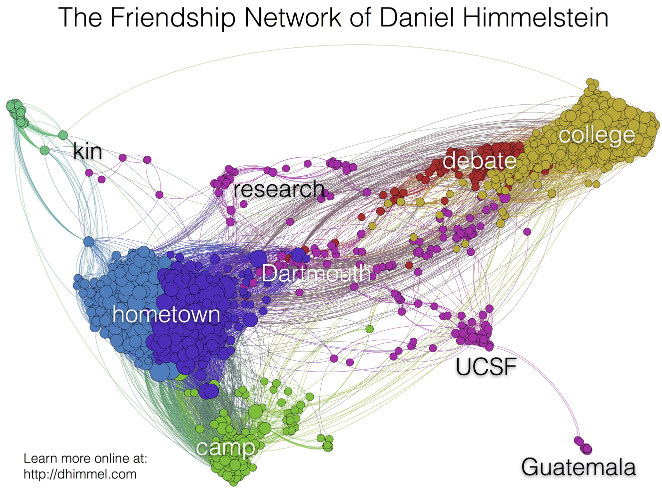
Definition: The Clustering Task

Given a collection of data cases $\mathbf{x}_i \in \mathbb{R}^D$, partition the data cases into groups such that the data cases within each partition are more similar to each other than they are to data cases in other partitions.

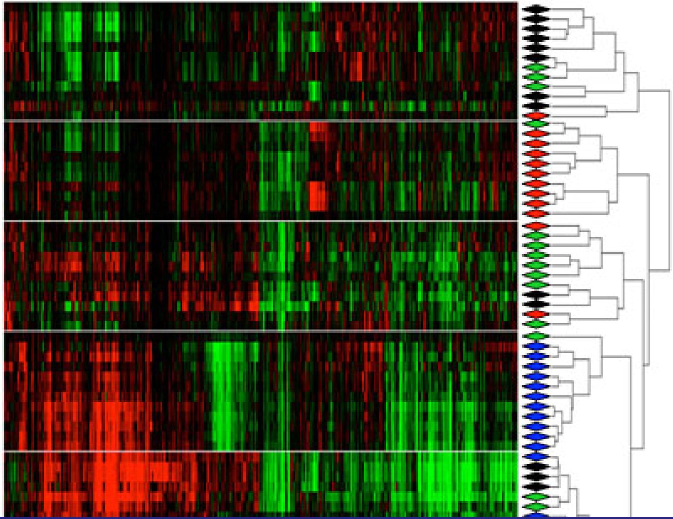


Examples: Market Segmentation

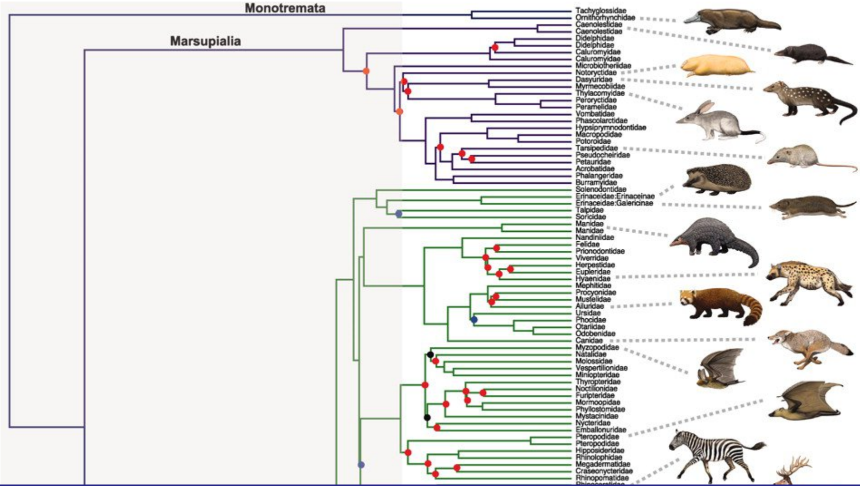
Examples: Community Detection



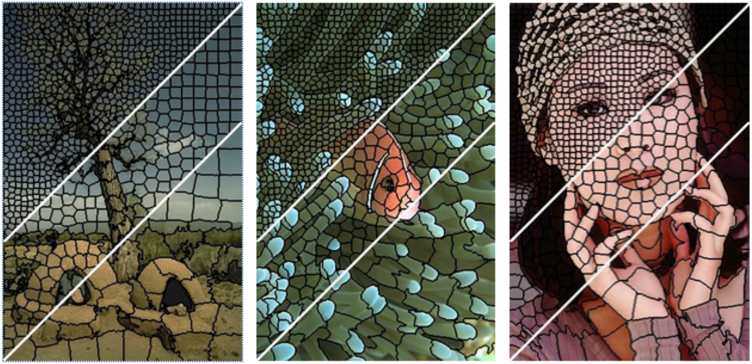
Examples: Gene Expression



Examples: Phylogenetic Trees



Examples: Super Pixels



Defining a Clustering

- Suppose we have N data cases $\mathcal{D} = \{\mathbf{x}_i\}_{i=1:N}$.
- A clustering of the N cases into K clusters is a partitioning of $\mathcal{D}$ into K mutually disjoint subsets $\mathcal{C} = \{C_1, ..., C_K\}$ such that $C_1 \cup ... \cup C_K = \mathcal{D}$.

Exhaustive Clustering

- Suppose we have a function $f(\mathcal{C})$ that takes a partitioning $\mathcal{C}$ of the data set D and returns a score with lower scores indicating better clusterings.
- The optimal clustering according to f is simply given by

$$\arg\min_{\mathcal{C}} \; f(\mathcal{C})$$

- **Question:** What is the complexity of exhaustive clustering?

Number of Clusterings

- The total number of clusterings of a set of N elements is the Bell number B_N where $B_0 = 1$ and $B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$.
- The first few Bell numbers are: 1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, ...
- The complexity of exhaustive clustering scales with B_N and is thus computationally totally intractable for general scoring functions.
- We will need either approximation algorithms or scoring functions with special properties.

Hierarchical Agglomerative Clustering

- Hierarchical Clustering methods are a family of greedy tree-based clustering methods.
- Hierarchical Agglomerative Clustering (HAC) is the most popular member of this family.
- It begins with all data cases assigned to their own clusters, and then greedily and recursively merges the pair of clusters that is optimal with respect to a given criteria.

Distance and Linkage Functions

- Like KNN, HAC need to be supplied with a function for computing the distance between two data cases. This is often taken to be Euclidean distance, but could be any distance function.
- To merge clusters, HAC also needs what is called a linkage function for measuring the distance between clusters.
- Linkage functions can differ significantly in their computational complexity and the clusterings they produce.

Examples of Linkage Functions

Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

The Hierarchical Agglomerative Clustering Algorithm

Algorithm 10.2 *Hierarchical Clustering*

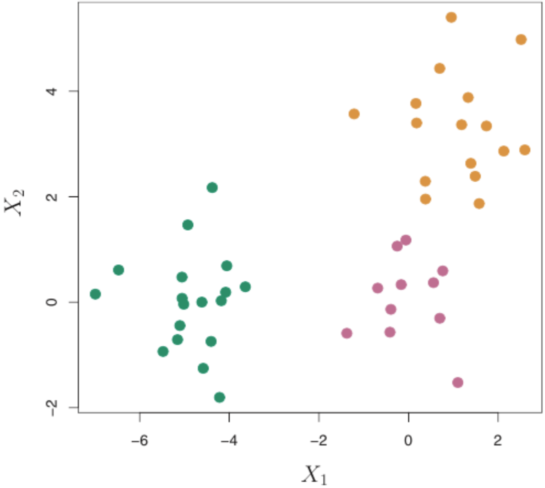
1. Begin with n observations and a measure (such as Euclidean distance) of all the $\binom{n}{2} = n(n-1)/2$ pairwise dissimilarities. Treat each observation as its own cluster.

2. For $i = n, n-1, \ldots, 2$:

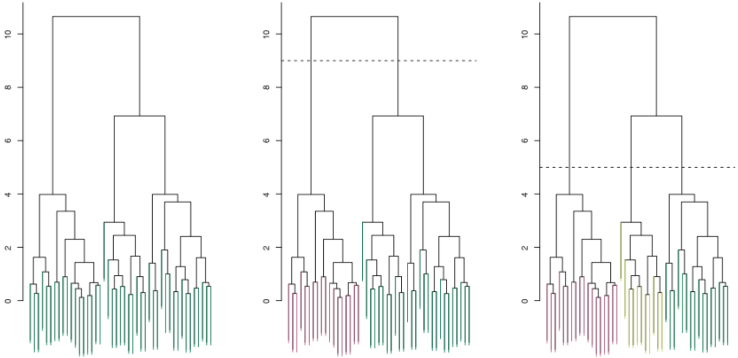
   (a) Examine all pairwise inter-cluster dissimilarities among the i clusters and identify the pair of clusters that are least dissimilar (that is, most similar). Fuse these two clusters. The dissimilarity between these two clusters indicates the height in the dendrogram at which the fusion should be placed.

   (b) Compute the new pairwise inter-cluster dissimilarities among the $i-1$ remaining clusters.

Example: Data



Example: Dendrograms



Issues

- We need to have a good notion of similarity for the results of cluster analysis to be meaningful at all.
- As with KNN, pre-processing like re-scaling/normalizing features can completely change the results.
- Further, we need to slect between the different linkage functions.
- We need some way to determine the “right” number of clusters to focus on. We want to cluster on salient differences between data cases, not noise.
- This procedure is not able to nicely handle noise observations that are different from each other and from the rest of the data that do belong to valid clusters.
- All of these issues mean we need to be cautious in interpreting the results of clustering. It should be the starting point for an exploratory data analysis, not the end point.