Clustering

One common form of "unsupervised" learning is *clustering*.

We divide the training cases into "clusters" (groups) so that cases within a cluster are "similar", while cases in different clusters are less similar.

Two applications

Find clusters of patients, based on similarity of symptoms.

> Doctors call these clusters "syndromes". A syndrome gets promoted to a "disease" once doctors think there is an actual underlying cause common to patients with this syndrome.

Find clusters of genes, based on similar expression profiles over tissues.

> If a cluster of genes are all highly expressed in liver cells, but not expressed in brain cells or skin cells, we may hypothesize that they all are involved in liver function, and may interact with each other.

Similarity / Dissimilarity

The dissimilarities of n cases can be expressed as an $n \times n$ matrix, $\mathbf{D}$, with elements $d_{ii'}$. We'll require that $\mathbf{D}$ be symmetric, and that the diagonal elements, d_{ii}, be zero.

We might fill in this matrix directly — eg, ask someone to rate how similar each pair of cases is.

Instead, if we have measurements on p variables for each case, we can define dissimilarity by the sum of dissimilarities between values of these variables:

$$d_{ii'} \;=\; D(x^{(i)}, x^{(i')}) \;=\; \sum_{j=1}^{p} D_j(x_j^{(i)}, x_j^{(i')})$$

where D_j measures the dissimilarity of two values for variable j.

If variable j is real-valued, the common choice is $D_j(x_j^{(i)}, x_j^{(i')}) = (x_j^{(i)} - x_j^{(i')})^2$, so the total dissimilarity is the squared Euclidean distance, $||x^{(i)} - x^{(i')}||^2$.

For a categorical variable, we could let $D_j(x_j^{(i)}, x_j^{(i')})$ be zero if $x_j^{(i)} = x_j^{(i')}$, and one otherwise.

Flat and Hierarchical Clustering

Do do *flat* clustering, we choose some number of clusters, K, and then either

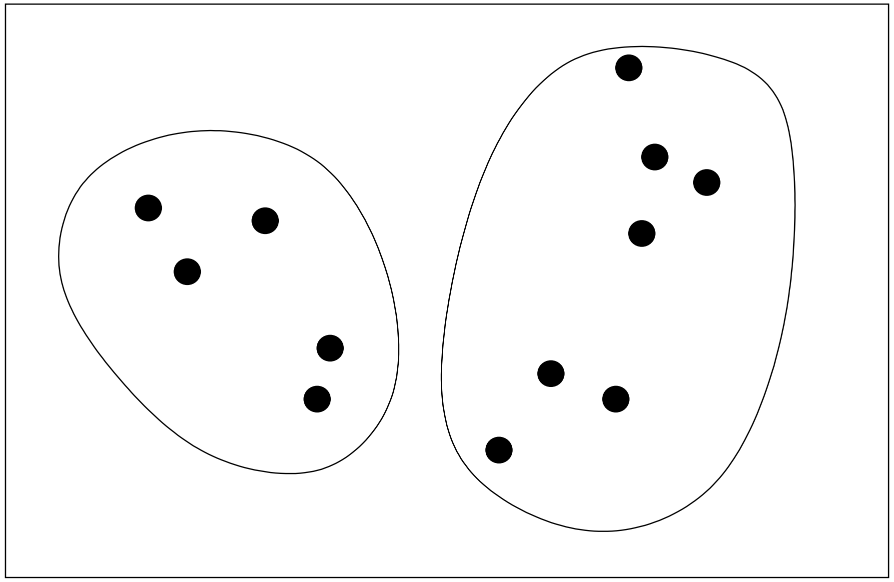
- Try to assign cases to clusters to directly maximize the "within cluster" similarity. Doing this exactly is possible only for small datasets.
- Make some initial assignment of cases to clusters, and then apply an iterative algorithm to try to improve the clustering.

An alternative is to find a *hierarchical* clustering (a tree), by either

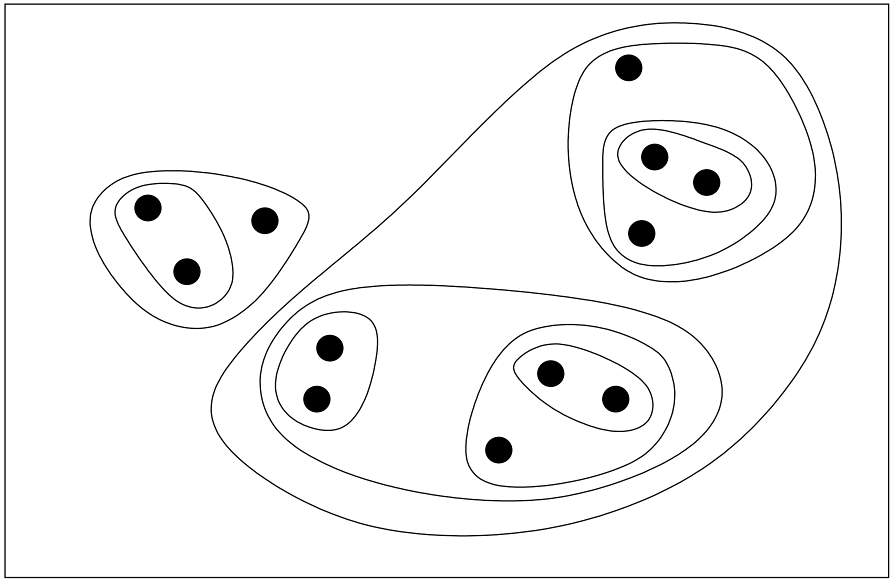
- Beginning with all cases in one cluster, and then iteratively dividing clusters until every case is in its own cluster (*divisive* clustering).
- Begin with a cluster for every case, and then iteratively merge clusters until there is just one cluster (*agglomerative* clustering).

Examples of Flat and Hierarchical Clusterings

Here are two clusterings of a 2D dataset with 12 cases, with dissimilarity measured by Euclidean distance.



A flat clustering with two clusters



A hierarchical clustering

From the hierarchical clustering, we can obtain many flat clusterings by eliminating parts of the clustering above or below some level.

K-Means Clustering

When the dissimilarities are squared Euclidean distances, we can use the K-*means* algorithm to iteratively find a good clustering with K clusters.

We start by choosing some initial centres ("means") for the clusters, $m_1, \ldots, m_K$ — perhaps by just setting each m_k to a randomly chosen training cases, $x^{(i)}$.

We then repeatedly apply the following two steps to improve the clustering:

1) Cluster the cases by assigning each case to the cluster with the closest mean. Ie, we set the cluster, $C(i)$, for case i by

$$C(i) \;=\; \underset{1 \leq k \leq K}{\operatorname{argmin}} \; ||x^{(i)} - m_k||^2$$

1) Using the current clustering, C, we find new mean vectors for each cluster by averaging the inputs for the cases assigned to that cluster:

$$m_k \;=\; \frac{1}{\#\{i\,:\,C(i)=k\}} \sum_{i\,:\,C(i)=k} x^{(i)}$$

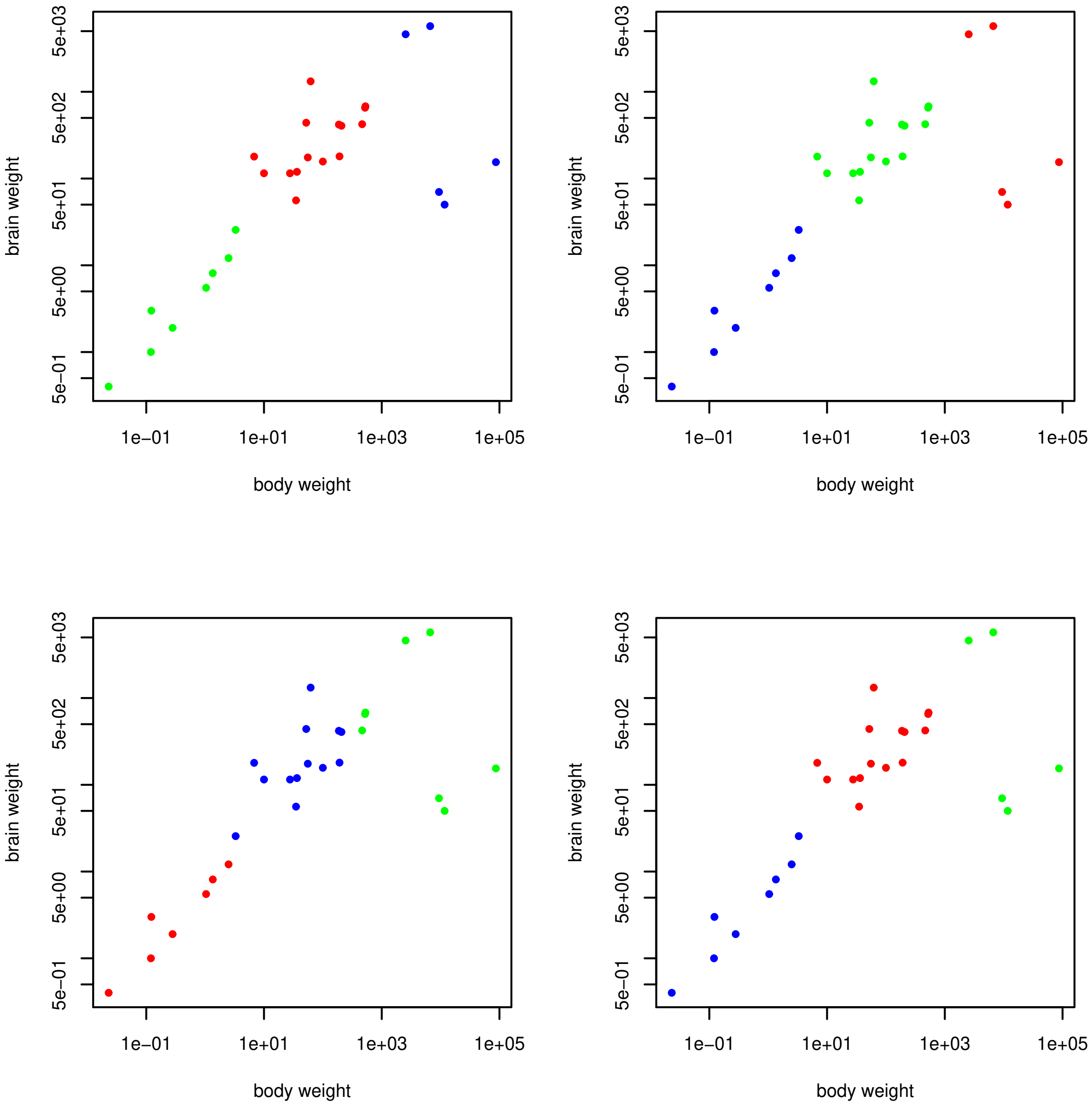
We stop when step (2) doesn't change C.

R Function for K Means Clustering

```
K_means = function (x, K)
{ x = as.matrix(x)
  m = x [sample(1:nrow(x),K), ] # matrix of cluster means
  rownames(m) = NULL
  c = rep(NA,nrow(x)) # vector of cluster assignments

  repeat
  { om = m

    # Assign all cases to the cluster with the nearest mean.

    for (i in 1:nrow(x))
    { c[i] = 1
      for (k in 2:K)
      { if (sum((x[i,]-m[k,])^2) < sum((x[i,]-m[c[i],])^2))
        { c[i] = k
        }
      }
    }

    # Update cluster means to the mean of cases assigned to the cluster.

    for (k in 1:K)
    { m[k,] = apply(x[c==k,,drop=F],2,mean)
    }

    # Exit if nothing has changed.

    if (all(m==om))
    { break
    }
  }
  list (means=m, clusters=c)
}
```

Results of K Means on a Body/Brain Weights of Animals

Clusters from four random starts with $K = 3$ are shown.



Hierarchical Clustering

Deciding on the number of clusters for a flat cluster is a problem. We can avoid it by finding a *hierarchical* clustering, in which we have clusters of clusters, in a tree.

Looking at different levels in the tree gives clusterings at various levels of detail. We might pick one such flat clustering, or look at the whole hierarchy.

Hierarchies are common ways of organizing categories. Biological taxonomies (species, genera, families, etc.) are elaborate hierarchies.

There are two sorts of algorithms for finding hierarchical clusterings:

- *Divisive* algorithms start with everything in one cluster, and then divide clusters until everything is in its own cluster.

- *Agglomerative* algorithms start with everything in its own cluster, and then merge clusters until everything is in one cluster.

Deciding what clusters to merge requires that we extend our dissimilarity measure for cases to a dissimilarity measure for clusters. This can be done in various ways, giving various agglomerative clustering methods.

Average Linkage Agglomerative Clustering

The most common way of measuring dissimilarity between clusters is to just average the dissimilarities for all pairs with one case in each cluster — that is, the dissimilarity of clusters G and H, containing N_G and N_H cases, is

$$\frac{1}{N_G N_H} \sum_{i \in G} \sum_{i' \in H} d_{ii'}$$

Using this gives the "average linkage" hierarchical clustering algorithm:

1) Assign every case to its own cluster.

2) Repeat the following, until everything is in one cluster:

   a) Find the two clusters with smallest dissimilarity.

   b) Replace these two clusters with one cluster, containing the cases in both of the merged clusters.

We keep track of all the clusters formed in step (2a). They form a tree, in which each merged cluster is the "parent" of the two clusters that were merged.

A picture of this tree is called a *dendrogram*. Sometimes the lengths of the branches in the dendrogram represent how far apart the merged clusters were.