K-Means Clustering

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

We start with some initial clustering — perhaps just a random division of the cases into K clusters.

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

1) Using the current clustering, C, find the mean vectors for each cluster:

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

2) Update the clustering by assigning each case to the cluster with the closest mean. Ie, the new clustering, C, is defined by

$$C(i) = \operatorname*{argmin}_{1\le k\le K} ||x_i - m_k||^2$$

We stop when step (2) doesn't change C. Figure 14.6 shows an example.

This algorithm finds a local minimum of the within-cluster scatter. It's advisable to run it several times with different initial clusterings, then use the best result.

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.

Clustering by Probabilistic Modeling

K-means flat clustering and average-linkage hierarchical clustering are non-probabilistic algorithms — based just on the intuitive notion of clustering together "similar" cases.

Instead, we can use a probabilistic model based on *latent variables* that describe the clustering.

For *mixture models*, the latent variable is a cluster indicator. (See Section 8.5 in the textbook.)

For *Dirichlet diffusion tree models*, the latent variable is a tree. (See papers on my webpage.)

Gaussian Mixture Models

Suppose we have univariate data points $y_1, \ldots, y_N$.

We may think that this data form several clusters, perhaps because it comes from a "mixture" of several sources.

Example: We capture N beetles. The weight of the i'th beetle is y_i. If there are K species of beetles, we may expect K clusters.

If we assume that the data form K clusters (ie, come from K sources) the probability density for a data point, y, can be written as

$$p(y) \;=\; \sum_{k=1}^{K} \pi_k \, g_k(y)$$

where π_k is the probability of a data point coming from cluster k, and $g_k(y)$ is the probability density of data from cluster k.

We might asssume the g_k are Gaussian. For simplicity, let's assume that these Gaussians have known variance, σ^2, so we just need to estimate their means, μ_k. We can write $g_k(y)$ as $g(y; \mu_k)$.

Expressing Mixtures Using Latent Variables

This mixture distribution can also be expressed using latent variables, $z_1, \ldots, z_N$, one for each data point, $y_1, \ldots, y_N$.

The value of z_i is an integer from 1 to K identifying which cluster y_i is in. So

$$P(z_i = k) \;\;=\;\; \pi_k$$

If we know which cluster y_i is in, it's distribution is just the distribution for that cluster. So

$$p(y_i \,|\, z_i = k) \;\;=\;\; g_k(y_i)$$

This produces the mixture we want as the marginal density for y_i:

$$p(y_i) \;\;=\;\; \sum_{k=1}^{K} P(z_i = k)\, p(y_i \,|\, z_i = k) \;\;=\;\; \sum_{k=1}^{K} \pi_k\, g_k(y_i)$$

The EM Algorithm for Mixture Models

We can estimate the parameters $\pi_1, \ldots, \pi_K$ and $\mu_1, \ldots, \mu_k$ by maximum likelihood. This can be done using the "Expectation-Maximization" (EM) algorithm, which for this model resembles a "soft" form of K-means.

Starting with some initial guesses, $\hat{\pi}_1, \ldots, \hat{\pi}_K$ and $\hat{\mu}_1, \ldots, \hat{\mu}_k$, we repeatedly do the following, until the estimates have converged:

E Step: Compute the "responsibilities" of each cluster for each data point, using the current values of the parameters:

$$r_{ik} \;=\; \hat{\pi}_k \, g(y_i; \hat{\mu}_k) \Big/ \sum_{k'=1}^{K} \hat{\pi}_{k'} \, g(y_i; \hat{\mu}_{k'})$$

M Step: Re-estimate the parameters based on the fractions of each data point in each cluster:

$$\hat{\pi}_k \;=\; (1/N) \sum_{i=1}^{N} r_{ik}, \qquad \hat{\mu}_k \;=\; \sum_{i=1}^{N} r_{ik} \, y_i \Big/ \sum_{i=1}^{N} r_{ik}$$

The result is a soft clustering of the data, along with estimates for the parameters of the mixture density.

More Elaborate Mixture Models

- In practice, we also estimate the variance in each cluster. We have to exclude nonsense estimates where one or more variances go to zero.
- It's easy to generalize this method to multivariate data — just assume that each variable is independent, given the cluster identifier.
- With a moderate number of variables, we can also model them as multivariate Gaussian with a covariance estimated for each cluster.
- It's possible to do Bayesian inference for mixture models like this, producing a posterior distribution for both the parameters and the clustering.
- It's not obvious how to pick K. One option with a Bayesian mixture model is to let K to go infinity — we assume that there are an indefinite number of clusters, though some are much more common than others.

Generating Data with Diffusion Trees

Hierarchical clustering produces a tree from data. Going, the other way, we might try to produce data using a tree.

We define:

– a distribution over trees

– a distribution for data given a tree

We then apply Bayes' Rule to get:

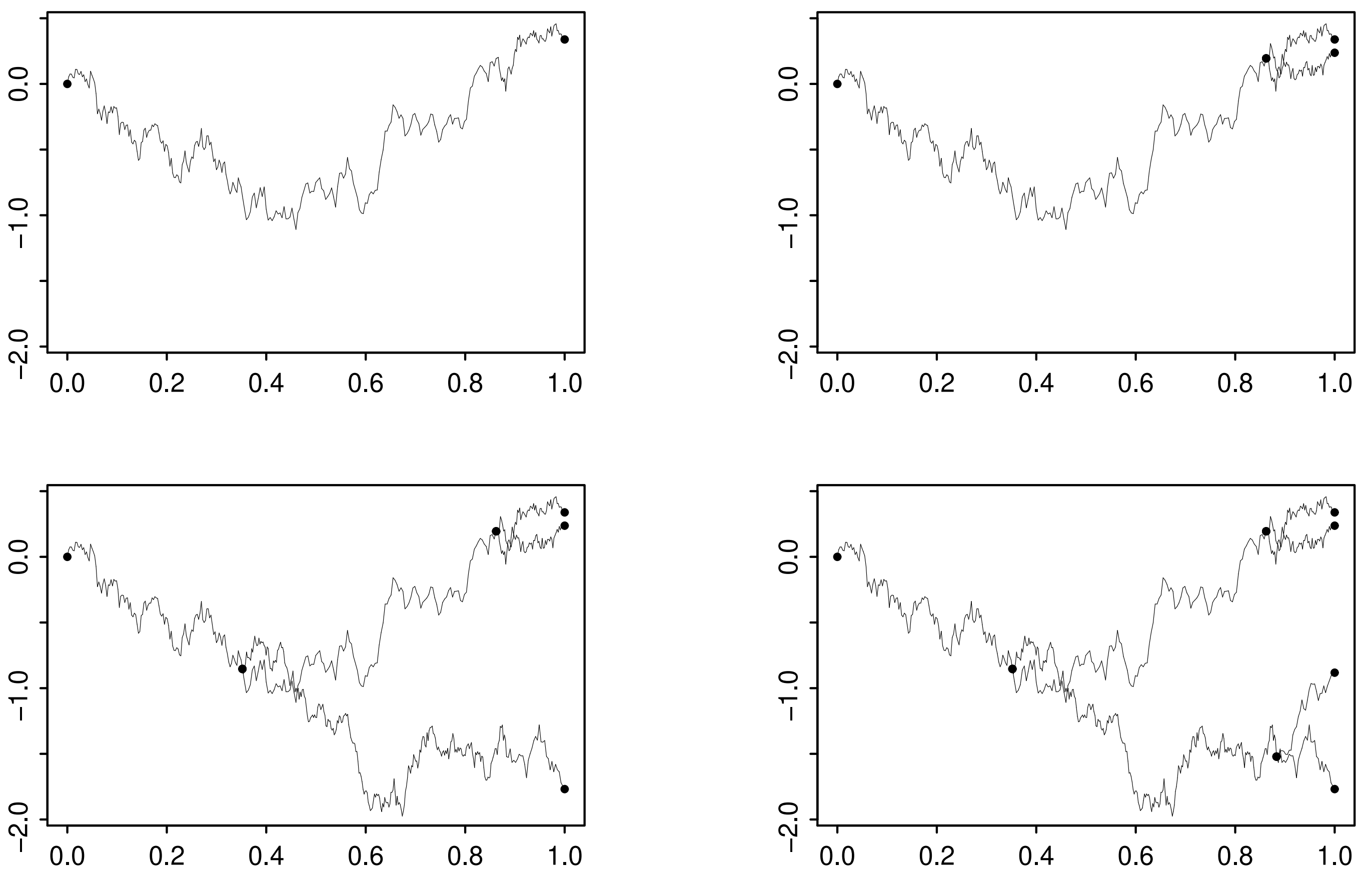
– a distribution of trees given data

This is a a Bayesian version of hierarchical clustering.

My scheme of *Dirichlet diffusion trees* does this.

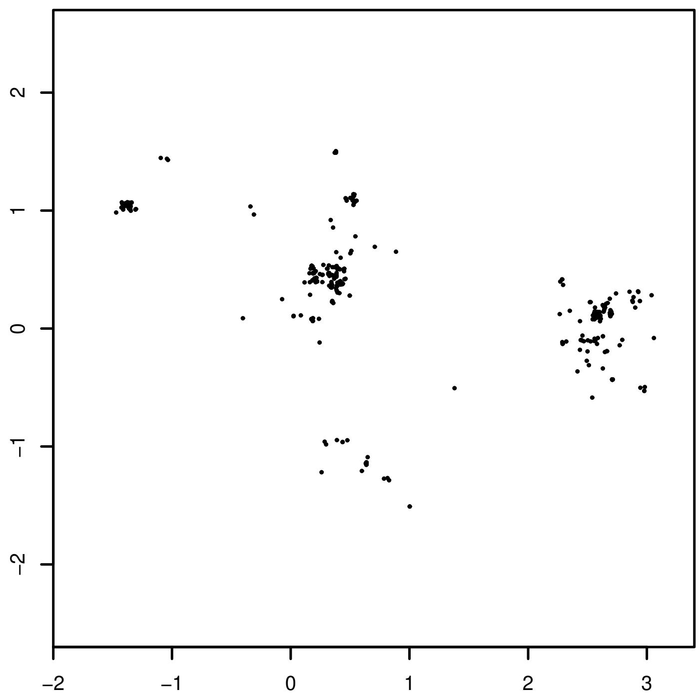
The Dirichlet Diffusion Tree Model

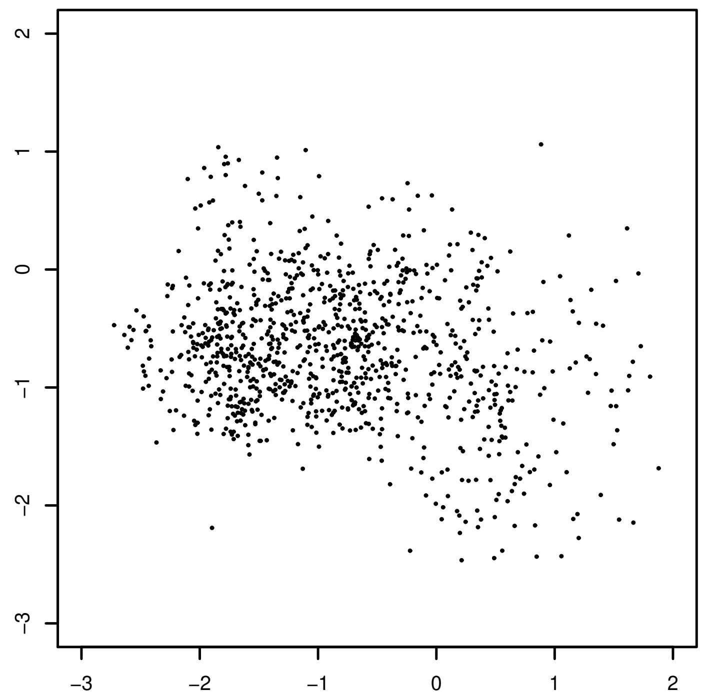
We imagine a point moving by "diffusion", starting at location 0 at time 0, continuing until time 1. The next point follows the same path, until it "diverges" at some time, with probability controlled by a divergence function, $a(t)$. Subsequent points again follow previous paths, but eventually diverge.



Example Datasets Generated with Dirichlet Diffusion Trees

Here are two 2D datasets generated by Dirichlet diffusion tree models:



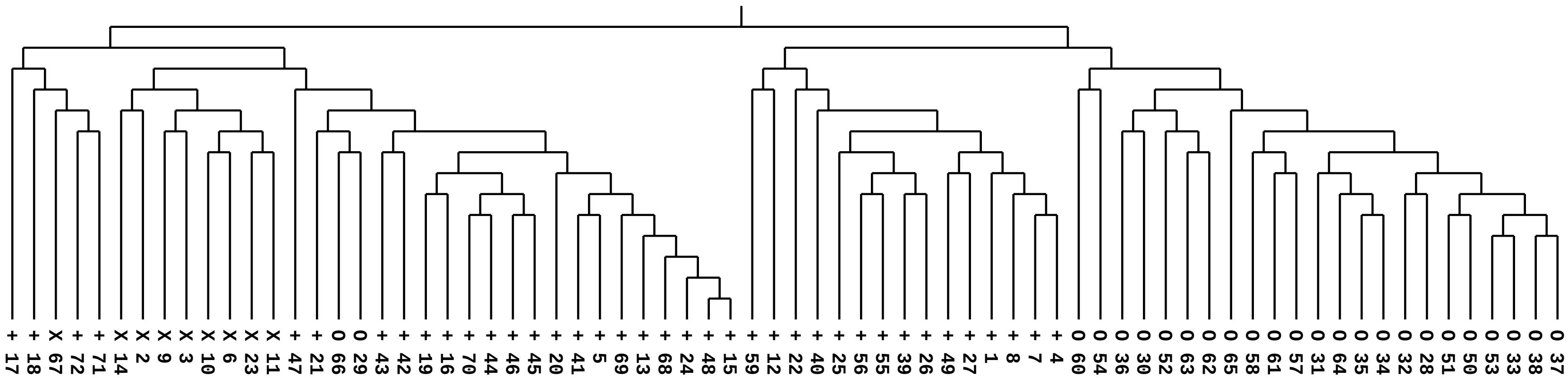


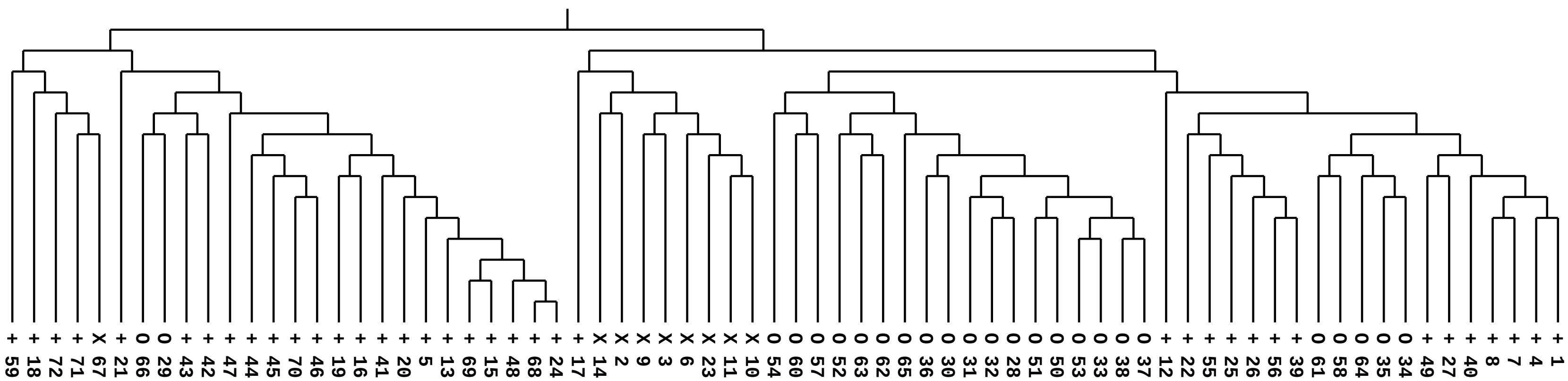
The two differ in the divergence function used.

On the left, $a(t) = (1/4)\,/\,(1-t)$, leading to late divergence, and tight clusters. On the right, $a(t) = (3/2)\,/\,(1-t)$, leading to earlier divergence, and broad clusters that overlap more.

Example of Clustering Gene Expression Data

Here are two clusterings from the posterior distribution of a Dirichlet diffusion tree model applied to data on gene expression in leukemia cells from 72 patients.





The “+”, “x”, and “o” labels indicate three categories of leukemia. The clustering was done without knowledge of these labels, but found related groupings.