Notes by Josh MacDonald

- Principle Component Analysis
  1. LSI
  2. Authoritative Sources
- Introduction to Clustering
  1. Applications
  2. Distance and similarity measures
- Clustering Algorithms
  1. Bottom-up Hierarchical
  2. Top-down Hierarchical
  3. Optimization
  4. Density Search

1 Principle Component Analysis

The study of *principle component analysis* (PCA) goes back at least as far as 1901 and has a number of closely related fields:

- factor analysis
- latent semantic indexing
- latent vectors
- Hotelling
- singular value decomposition
- principle components
- SV transform

The objective of PCA is to find the eigenvectors and associated eigenvalues of a similarity or correlation matrix. For n items an $n \times n$ similarity matrix S is one in which the entry S_{ij} represents the similarity between items i and j—the matrix is typically symmetric. The eigenvector of S with the highest eigenvalue specifies the direction that best "separates" the items. In particular it is the projection of the items into a 1 dimensional space that best maintains the similarity metric (*i.e.*, similar items have close values in the eigenvector).

Here we will briefly look at how latent semantic indexing and authoritative sources are just special cases of PCA. Recall that the singular value decomposition of an $m \times n$ matrix A yields the following:

$$A = U\Sigma V^T$$

where

$$U^T U = V^T V = I$$

and Σ is a diagonal matrix with diagonal entries $\sigma_1, \sigma_2, \ldots$. If A has rank r (number of independent rows or columns) then U has dimension $m \times r$, V has dimension $n \times r$, and Σ has dimension $r \times r$. (In lecture 22 we assumed that for $m \geq n$ that r is replaced with n in the stated sizes, but when $r < n$ only the first r columns, or rows, are actually important and the rest can be dropped.) By using the above equations, we get

$$\begin{aligned} A^T A &= (U\Sigma V^T)^T (U\Sigma V^T) \\ &= (V\Sigma^T U^T)(U\Sigma V^T) \\ &= V\Sigma^2 V^T \end{aligned} \tag{10}$$

and by multiplying by V on the right we get

$$\begin{aligned} A^T A V &= V\Sigma^2 V^T V \\ &= V\Sigma^2 \end{aligned}$$

This shows that each column of V is an eigenvector of $A^T A$ with eigenvalue σ_i^2. Similarly, each column of U is an eigenvector of AA^T with eigenvalue σ_i^2. We can therefore think of the columns of V as the principal components of a similarity matrix $A^T A$ and the columns of U as the principal components of a similarity matrix AA^T. Since $A^T A$ is just all the dot products of the columns, the "similarity" measure being used by V is dot product of columns, and the "similarity" measure being used by U is dot product of rows.

Latent Semantic Indexing

We discussed latent semantic indexing as a method for improving text-retrieval with the hopes of finding content which did not directly match our query, but was semantically "close". Here the matrix elements A_{ij} represents the weight of each term i in a document j. Thus each row represent a term in the dictionary, and each column represent a document. In this framework, U represents the principal components of the dot product of rows (*i.e.*, term-term similarities), and V represents the principal components of the dot product of columns (*i.e.*, the document-document similarities). If we normalize the rows of A, for example, then the similarity measure being used to generate U is simply the cosine measure.

Authoritative Sources

The hyper-linked graph of documents represent a directed graph from which we construct an adjacency matrix A (*i.e.*, place a 1 in A_{ij} if there is a link from i to j and a 0 otherwise).

The matrix A^TA now represents for each pair of documents i and j how many links they have in common, and the matrix AA^T represent for each pair of documents i and j how many common pages link to both of them. These matrices can be thought of as similarity measures between sources, and destinations, respectively. In the definitions given in the last class a “hub” is a page that points to multiple relevant authoritative pages. The principal component of A^TA (also the first column of U in the SVD decomposition of A) will give the “principal hub”. An “authority” is a page that is pointed to by multiple relevant hubs. The principal component of AA^T (also the first column of V in the SVD decomposition of A) will give the “principal authority”.

2 Introduction to Clustering

The objective of *clustering* is, given a set of objects and a measure of similarity or distance between the objects, cluster into groups of similar (nearby) objects. Clustering is also called:

- typology
- grouping
- clumping
- classification
- unsupervised learning
- numerical taxonomy

2.1 Applications

There are many applications of clustering, here are a few:

- biology: multiple alignment, evolutionary trees
- business: marketing research, risk analysis
- liberal arts: classifying painters, writers, musicians
- computer science: compression, vector quantization, information retrieval, color reduction
- sociology and psychology: personality types, classifying criminals, classifying survey responses and experimental data.
- indexing and searching: group results into categories (Hearst and Pederson, 1996)

2.2 Similarity and Distance Measures

To cluster the items in a set we need some notion of distance and/or similarity between items. Typically when one talks about a distance measure $d(i,j)$ between items i and j in a set E one assumes that d is a *metric* for E, and therefore abides by the following conditions:

$$\begin{aligned} d(i,j) \geq & \quad 0 \\ d(i,i) = & \quad 0 \\ d(i,j) = & \quad d(j,i) \\ d(i,j) \leq & \quad d(i,k) + d(k,j) \end{aligned}$$

The fourth condition is the familiar triangle inequality. These conditions are met for common metrics such as the Euclidean distance and the Manhattan distance. Metrics may be combined, so, for example, the maximum of two metrics is itself a metric space. Many clustering algorithms that use distance measures assume all four conditions are met.

A similarity is a measure of how close two points are in a space. We are familiar with cosine measures, dot-products, and identity as measures of similarity. In general, however, similarities might be supplied by a black box or by human judgment.

One question is whether similarities and distances can be used interchangeably. Given an arbitrary distance measure we can always come up with a corresponding similarity measure:

$$s(i,j) = \frac{1}{1 + d(i,j)}$$

Given a similarity measure (ranging from 0 to 1) we might also try the corresponding distance metric

$$d(i,j) = \frac{1}{s(i,j)} - 1$$

This measure, however, does not necessarily form a metric. Therefore one should be careful when using such a transformation to generate a distance from a similarity, especially if it is going to be applied in a clustering algorithm that assumes the distance function forms a metric.

3 Clustering Algorithms

We will be considering four classes of algorithms for clustering.

- Bottom-up Hierarchical (agglomerative)
- Top-down Hierarchical (divisive)
- Optimization
- Density searches

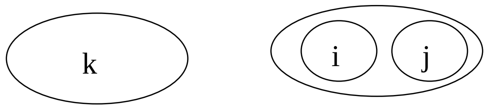

Figure 151: Lance and Williams' Rule

The various algorithms we consider not only vary in how they work but in what they are optimizing. In some applications it is important to have a fixed number of clusters, while in others it is better to find "natural" clusters which will return a number of clusters that depends on the nature of the data. In some applications, such as some index searching techniques, a hierarchy of clusters is required while in others it is not. In some applications it is best to minimize the maximum distance within each cluster, while in others it is better to minimize the average distance. There is therefore no single "correct" clustering of a set of items, but instead what constitutes a good clustering will depend on the application.

3.1 Bottom-Up Hierarchical Algorithms

The basic algorithm for bottom-up, or agglomerative, methods is to begin with a set of points and appropriate distance measure, and initially place each point in its own group. And then while the number of groups is greater than one, merge the closest pair of groups into a single group. This basic structure leads to a whole collection of different methods depending on how distance between groups is measured. Some common measures are minimum, maximum, centroid, average, or sum-of-squares distance between the points in each group. The last, sum-of-squares method is also known as "Wards' method", and uses the function:

$$SS(G_{12}) - SS(G_1) - SS(G_2)$$

where

$$SS(G) = \frac{1}{2|G|} \sum_{x \in G, y \in G} (d_{xy})^2$$

Note that when using minimum as the distance measure the method simply finds the minimum spanning tree of the graph in which the items are the vertices and the distance $d(i,j)$ is used as the weight of an edge from i to j.

An interesting generalization of the above five group-distance measures is due to Lance and Williams, 1967. They show a function that generalizes the previous metrics for the distance between cluster k and the cluster consisting of sub-clusters i and j. The metric must be recursively applied. There are four parameters, α_i, α_j, β, and γ. n_x represents the number of elements inside cluster x.

$$d_{k(ij)} = \alpha_i d_{ki} + \alpha_j d_{kj} + \beta d_{ij} + \gamma |d_{ki} - d_{kj}|$$

With this formula, we can produce the nearest neighbor metric,

$$\alpha_i = \alpha_j = \frac{1}{2};\ \beta = 0;\ \gamma = -\frac{1}{2}$$

the furthest neighbor metric,

$$\alpha_i = \alpha_j = \frac{1}{2};\ \beta = 0;\ \gamma = \frac{1}{2}$$

the centroid metric,

$$\alpha_i = \frac{n_i}{n_i + n_j};\ \alpha_j = \frac{n_j}{n_i + n_j};\ \beta = -\alpha_i \alpha_j$$

and Wards' method,

$$\alpha_i = \frac{n_k + n_i}{n_k + n_i + n_j};\ \alpha_j = \frac{n_k + n_j}{n_k + n_i + n_j};\ \beta = \frac{-n_k}{n_k + n_i + n_j}$$

3.2 Top-Down Hierarchical Algorithms

Top-down clustering approaches are broken into *polythetic* and *monothetic* methods. A monothetic approach divides clusters based on analysis of one variable at a time. A polythetic approach divides clusters into arbitrary subsets.

Splinter Group Accumulation

A technique called *splinter group accumulation* removes the most outstanding item in a cluster and adds it to another, repeating until moving the item will increase the total cost measure. Pseudo-code for this algorithm follows, where we assume the measure we are trying to optimize is the average distance within a group.

$C_1 = P$
$C_2 = \{\}$
loop
  for each $p_i \in C_1$
    $d_i = \frac{1}{|C_2|} \sum_{p_j \in C_2} d(i,j) - \frac{1}{|C_1|} \sum_{p_j \in C_1} d(i,j)$
  if $\max d_i > 0$
    $C_2 = C_2 \cup P_i$
    $C_1 = C_1 - P_i$
  else
    return

We could also use a different measure in the code above, such as Wards' method.

Graph Separators

A cut of a graph separates vertices V into two sets, V_1 and V_2. The weight of a cut is the sum of each edge weight crossing the cut:

$$W_{cut} = \sum_{i \in V_1} \sum_{j \in V_2} w_{ij}$$

Assuming we use a similarity measure, the goal is to find a cut which minimizes W_{cut}, where the weights are the similarities, while keeping the two clusters approximately the same size. Various conditions can be used, such as forcing the ratio of sizes to be bounded by some number, usually less than two. In the general case, finding the optimal separator is NP-hard.

Finding good separators for graphs is a topic of its own. Other applications of graphs separators include

- Solving sparse linear systems (nested dissection). The goal is to minimize *fill*.
- Distributing graphs or sparse matrices on multiprocessors. The goal is to minimize parallel communication during program execution.
- VLSI layout–finding placements that minimize communication.

For geometric problems, methods such as the *kd*-tree or circular cuts can be used to compute separators. Without geometric information, the *spectral method* uses a technique similar to singular value decomposition (or principal component analysis). The principal component (actually the second eigenvector since the first turns out to be "trivial") is used to separate the points — the points are split by the median value of the vector.

3.3 Optimization Algorithms

Expressed as an optimization problem, clustering attempts to divide data into k groups to maximize or minimize some measure, *e.g.*,

- sum of intra-group distances
- sum squared of intra-group distances
- maximum intra-group distances

Most measures lead to an NP-hard problem and heuristics are often applied.

Optimization as an Integer Program

One technique uses integer programming to minimize the sum of intra-group distances. Stated as an integer programming problem:

minimize $\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} x_{ij}$
subject to
$1 \leq i \leq n$
$1 \leq j \leq n$
$1 \leq k \leq g$
$\sum_{j=1}^{g} y_{ij} = 1$
$\frac{1}{g} \sum_{k=1}^{g} z_{ijk} \leq x_{ij}$
$y_{ik} + y_{jk} - 1 \leq z_{ijk}$

where g is the number of groups, $x_{ij} = 1$ if object i and j are in the same group, $y_{ik} = 1$ if object i is in group k, and $z_{ijk} = 1$ if objects i and j are in group k. The first condition states that each object must be in one group. The second condition states that i and j may appear in at most one group together. The final condition relates y and z, stating that if two objects are in the same group, z is true.

3.4 Density Search Algorithms

Density search techniques try to find regions of "high density". There are many such techniques. A particularly simple method starts with the minimum-spanning-tree for the graph:

1. find MST
2. remove all edges with weights much greater than the average for its vertices
3. find connected components.

This method has the advantage of finding "natural" groupings rather than fixing the number of groups, because the size and granularity of the clusters it produces are dependent on the parameter used to remove edges.

3.5 Summary

There are many algorithms for clustering. The key question to be answered before picking an algorithm is what the model of a good clustering is. The challenge for authors of these algorithms today seems to be making clustering fast for large collections of data.