CS769 Spring 2010 Advanced Natural Language Processing

Clustering

Lecturer: Xiaojin Zhu *jerryzhu@cs.wisc.edu*

Clustering is an unsupervised learning method. Given items $x_1, \ldots, x_n \in \mathbb{R}^D$, the goal is to group them into reasonable clusters. We also need a pairwise distance/similarity function between items, and sometimes the desired number of clusters.

When documents are represented by feature vectors, a commonly used similarity measure is the *cosine similarity*. Let x, x' be two document vectors. There is an angle θ between the two vectors x, x'. The cosine similarity is defined as

$$\begin{aligned} sim(x, x') &= \cos(\theta) && (1)\\ &= \frac{x^\top x'}{\|x\| \cdot \|x'\|} && (2)\\ &= \frac{x^\top x'}{\sqrt{x^\top x}\sqrt{x'^\top x'}}. && (3) \end{aligned}$$

This similarity has the nice property that document length is implicitly normalized (so that a long document can be similar to a short document). This would not be the case if one uses the Euclidean distance between x, x'.

1 Agglomerative Hierarchical Clustering

This is a very simple procedure:

1. Initially each item $x_1, \ldots, x_n$ is in its own cluster $C_1, \ldots, C_n$.
2. Repeat until there is only one cluster left:
3. Merge the nearest clusters, say C_i and C_j.

The result is a cluster tree. One can cut the tree at any level to produce different clusterings. A little thought reveals that "the nearest clusters" are not well-defined, since we only have a distance measure $d(x, x')$ between items. This is where the variations come in:

- $d(C_i, C_j) = \min_{x \in C_i, x' \in C_j} d(x, x')$. This is known as *single-linkage*. It is equivalent to the minimum spanning tree algorithm. One can set a threshold and stop clustering once the distance between clusters is above the threshold. Single-linkage tends to produce long and skinny clusters.
- $d(C_i, C_j) = \max_{x \in C_i, x' \in C_j} d(x, x')$. This is known as *complete-linkage*. Clusters tend to be compact and roughly equal in diameter.
- $d(C_i, C_j) = \frac{\sum x \in C_i, x' \in C_j d(x, x')}{|C_i| \cdot |C_j|}$. This is the average distance between items. Somewhere between single-linkage and complete-linkage.
- and a million other ways you can think of ...

How do we evaluate the quality of a clustering? Unfortunately, we are out of luck here. If each item has a true label, we can compute an accuracy, but this almost goes back to classification. In reality most clustering problems do not come with ground truth labels. Perhaps the following famous quote will illustrate the problem:

"The correct clustering is whatever *my* program outputs."

A related (simpler?) problem is to determine the number of clusters. There are ways to define the optimal number, if one makes strong assumptions about the structure of the data. But in general this is an ill-posed problem too.

2 k-Means Clustering

This is a widely used clustering algorithm. It assumes that we know the number of clusters k. This is an iterative algorithm which keeps track of the cluster centers (means). The centers are in the same feature space as x.

1. Randomly choose k centers $\mu_1, \ldots, \mu_k$.
2. Repeat
3. Assign $x_1 \ldots x_n$ to their nearest centers, respectively.
4. Update μ_i to the mean of the items assigned to it.
5. Until the clusters no longer change.

Step 3 is equivalent to creating a Voronoi diagram under the current centers. k-means clustering is sensitive to the initial cluster centers. It is in fact an optimization problem with a lot of local optima[1]. It is of course sensitive to k too. Both should be chosen with care.

Since k-means is the limiting case of EM with Gaussian mixture models, when the variance of the Gaussian approaches 0, it is unable to trace winding clusters. This is addressed by spectral clustering.

3 Spectral Clustering

Spectral clustering is best understood not as a clustering algorithm by itself, but as a "preprocessing" step to change the feature representation of x. Imagine you have many points on a piece of paper (this is one cluster). You bend, twist, fold, roll the paper and hang it in a room. Your paper is now a 2D *manifold* in a 3D *embedding* (or ambient) Euclidean space. Each point is represented by its 3D coordinates (the feature representation). Do the same thing for many pieces of paper (more clusters) and hang them close to each other (but not touching). This roomful of papers can be bad for a clustering algorithm using the Euclidean distance, because a point x_i could be closer to a point x_j from a different cluster than to points in its own cluster (imagine two flat paper in parallel with a small gap). With spectral clustering, one can perform a non-linear "warping" so that each piece of paper (and all the points on it) shrinks to a single point (or a very small volume) in some new feature space. Clustering in that new space is trivial, with e.g. k-means.

3.1 The Graph

Spectral clustering takes a graph W and the number of clusters C as input. The graph nodes are $x_1, \ldots, x_n$. The undirected edges have non-negative weights and reflect the (local) similarity between nodes. The weights are symmetric: $w_{ij} = w_{ji}$. $w_{ij} = 0$ if there is no edge. The weights can be arranged in an $n \times n$ weight matrix W, which fully specifies the graph. The graph is often generated with one of the following three methods:

1. k-nearest-neighbor (kNN) graph. The Euclidean distance between x_i, x_j is $\|x_i - x_j\|$. Connect x_i with x_j if x_i is within the kNN of x_j, or vice versa. The edges usually have the same weight $w_{ij} = 1$ (which is confusingly called an unweighted graph). The neighborhood size k should be chosen so that it is small, while the graph is connected. This is the most frequently used graph.

[1] To be exact, k-means clustering is a special case of Gaussian Mixture Model (GMM) when the covariance of the Gaussian components tends to zero.

2. Fully connected graph with RBF weights. All points are connected. $w_{ij} = \exp(-\|x_i - x_j\|/\sigma^2)$.

3. ϵNN graph. Connect x_i with x_j if $\|x_i - x_j\| \leq \epsilon$. This graph is usually unweighted too.

In all kinds of graphs, the idea is to keep the similarities within a small neighborhood (in this case the distance in the embedding space is close to the manifold geodesic distance), and ignore long range similarities (where the embedding distance is no longer a good approximation to geodesic distance).

3.2 The Graph Laplacian and Its Spectrum

Any $n \times n$ symmetric matrix M has n real eigenvalues $\lambda_1, \ldots, \lambda_n$ (not necessarily distinct), and the corresponding eigenvectors $\phi_1, \ldots, \phi_n$ (of dimension n. We assume the eigenvectors have norm 1). We always sort the eigenvalues from small to large. Recall the definition

$$M\phi_i = \lambda_i \phi_i. \tag{4}$$

The set of eigenvalues are called the spectrum of M, because of the following decomposition:

$$M = \sum_{i=1}^{n} \lambda_i \phi_i \phi_i^\top. \tag{5}$$

Let the degree matrix be a diagonal matrix with

$$d_{ii} = \sum_{j=1}^{n} w_{ij}, \tag{6}$$

i.e., the sum of edge weights connected to x_i.

From the weight matrix W, we will define 3 slightly different *graph Laplacian* matrices:

1. The *unnormalized Laplacian* (also called combinatorial Laplacian) matrix $L = D - W$.

2. The *normalized Laplacian* $L_{rw} = I - D^{-1}W$.

3. Another *normalized Laplacian* $L_{sym} = I - D^{-\frac{1}{2}} W D^{-\frac{1}{2}}$.

We will be particularly interested in the spectrum of the Laplacian matrices. It turns out that the eigenvalues of a Laplacian is always non-negative. The key property is the following:

Theorem 1 *If the graph W has C connected components, then L, L_{rw}, L_{sym} have exactly C zero eigenvalues (other eigenvalues are positive). Furthermore, for L and L_{rw} those corresponding C eigenvectors are proportional to the indicator vectors of each component*[2]*.*

3.3 Change of Representation and Clustering

Theorem 1 is the basis of spectral clustering. In the ideal case, each cluster forms a connected component in graph W. Let us use L or L_{rw}. Let U be the $n \times C$ matrix formed with those C eigenvectors as columns. We represent x_i by the i-th row in U for $i = 1 \ldots n$. Then all points within a cluster have the same new representation, i.e., we have shrunken each piece of paper into a unique point!

However, in reality the graph may not consists of exactly one connected component per cluster. In this case, we view W as a perturbed version of the ideal graph: $W = W_{ideal} + B$. The B matrix contains the edge weights we add/subtract to the ideal graph. Perturbation theory states that as long as the perturbation B is not too large, the first C eigenvectors will not change too much. We can still use the new representation, but each piece of paper is now shrinking into a small region instead of a single point. This is still sufficient for k-means algorithm to easily identify the C clusters.

The complete spectral clustering algorithm is given below.

[2]Up to rotation within the eigenspace of eigenvalue 0. For L_{sym} the indicator vectors are scaled by $D^{1/2}$.

1. Input: graph W, number of clusters C
2. Compute unnormalized Laplacian $L = D - W$ or normalized Laplacian $L_{rw} = I - D^{-1}W$.
3. Compute the first C eigenvectors $\phi_1, \ldots, \phi_C$.
4. Let $U = [\phi_1 | \ldots | \phi_C]$. Represent x_i by the i-th row in U, for $i = 1 \ldots n$.
5. Use k-means to cluster them under the new representation into C clusters.

What if we do not know the number of clusters? The spectrum of the Laplacian offers some clue. Look for a large gap (a "jump") between λ_C and λ_{C+1} for some C. This may be used as a heuristic to pick C.

In practice, the quality of spectral clustering depends on how well W approximates the ideal one-cluster-per-connected-component graph. This in turn depends on the parameter k or σ or ϵ when creating W. There is unfortunately no principled way to choose those without making assumptions.