CMSC35000-1 Introduction to Artificial Intelligence **Winter 2005**

Lecture 8: Jan.27, 2005

Lecturer: Partha Niyogi *Scribe: Haitao Gao*

8.1 Hierarchical Clustering

K-means clustering works on a lot of data but may have problems when you have circles or triangles in the cluster. The hierarchical Clustering is a multiscale tree like clustering. The idea is:

Started with data x_1,...x_n, regroup two clusters to the nearest.

In order to do this, we must have a notion of distance between two clusters. Given two disjointed clustering sets, say $C = X_i, ...X_{i_k}$ and $C' = X_j, ...X_{j_k}$, the distance between cluster C_i and C_j is

$$d(C_i, C_j) = ||X_i - X_j|| \tag{8.1}$$

If the clusters are not singleton, there are many possibilities. We can choose the means or for example, we can choose x, y where $x \in C, y \in C'$. The worst case distance is

$$\max_x \min_y ||x - y||$$

Once we have this notion of distance we start at the singleton clusters and combine nearby clusters as shown following:

$$\begin{array}{r}
1 \longrightarrow C_1^{(1)} \\
2 \longrightarrow C_1^{(2)} \;\; C_2^{(2)} \\
...\;...\;... \\
...\;...\;...\;... \\
...\;...\;...\;...\;... \\
n-2 \longrightarrow C_1^{(n-2)}...\;...C_{n-2}^{(n-2)} \\
n-1 \longrightarrow C_1^{(n_1)}...\;...\;...C_{n-1}^{(n-1)} \\
n \qquad \longrightarrow X_1, ...\quad ...\;...\;...\;...\; X_n \\
\textit{corresponding to cluster } C_1...\;...\;...\;...\;...\;...\; C_n
\end{array}$$

We may cluster not only vectors but also sequences, functions, words. We can take the pattern and embedded it to a vector space. So sometimes sequences are considered as boolean queues. Linguist problem also has a vector space representation. For example, let's look at all the word follows *over* in a text and how often does each word appear after *over*. Each words has a vector space representation with different probabilities, i.e.

$$over—w_1 ... w_{60,000}$$

Another thing we didn't discuss well in K-means as well is how do you choose the right number of clusters. Associate with each cluster's level a goodness d_i. You will like it to go to minimum at the right value of k. Suppose we have centroids $u_i ... u_k$ and $x \in X = x_21, ... x_k$, then

$$\begin{aligned} d(x) &= \min ||x - u_j||^2 \\ d_k &= \min_{\{u_1,...u_k\}} \sum_{x \in X} d(x) \end{aligned}$$

where $d_n = 0$, d_1 = variance. Then we have some questions: What is a good formula for d_i? What happens as data goes to infinity? If you know the true probability distribution, how to partition the space? Define

$$\frac{1}{n} \sum_{x \in X} d(x) \approx \min_{\{u_1,...u_k\}} \mathtt{E} d(x)$$

We end up at the same situation that there is no good answer for how to choose the right d_i. We might choose a good formula to d_i, estimate $\hat{k}$, show $\hat{k}$ converges to $k*$. And we have not set what $k*$ is. There is some empirical experience of this but no good theory exists for this. This is a good term project you may like to consider.

8.2 Graph based Clustering

Now we are going to look at graph based clustering. We first look at something simple but fundamental thing then we go further. Let's look at how to make a graph out of data. First we have

Data $\longrightarrow$ Graph $\longrightarrow$ Partition to graph(involves partitioning to the data)

The question of given finite objects to partition them into k groups is actually NP-hard. If give you a new object you will not know which group to put it. There is no notion of generalization for graph based clustering. K-means does have a nature generalization. Given a new data, you can assign it to the nearest centroid. Hierarchical clustering does not have the generalization. Graph based clustering definitely do not have this property.

The idea of **Graph based clustering** is the graph will be a local graph. The nearby points are connected to nearby point. Given $x_1, ...x_n$, we choose $\varepsilon > 0$, then put an edge such that

$$x_i \mapsto x_j \Longleftrightarrow ||x_i - x_j|| < \varepsilon$$

How to choose for ε? It is important. Again this is an open question.
Now if you have this graph, you actually do a relaxation. Partition this graph into finite connected component weighted graph. Let W=adjacency matrix where W_{ij} is the weight associated with Node x_i and x_j. $W_{ij} = 0$ if the edge is empty. We like to minimize

$$\min_{A} \sum_{i \in A, j \in \bar{A}} W_{ij}$$

Why this is not a good satisfactory? You may have very dense region. A balanced cut on only one edge is the best cut, a cut on two is a second best and a cut on many edges is a worst cut. Now we must have a notation of a balanced cut which means A and A' have roughly equal sets. Then we can associate the graph to linear algebra.
We find G=(V,E) f:V $\longrightarrow \{-1, 1\}$.

$$\min_{f:v \to [-1,+1]} \frac{1}{4} \sum_{i,j} (f_i - f_j)^2 W_{ij} \tag{8.2}$$

Now can do something as pattern classification. You will have a function mapping vertices to real value then do a sign function. Do search of $f : V \longrightarrow R$. This is the analogy of Laplacian function on function of graph. Now the basic identify we use

$$\int ||\nabla f||^2 = \int f \bigtriangleup f \tag{8.3}$$

$$\sum_{i,j} (f(i) - f(j))^2 W_{ij} = 2\bar{f}^T L \bar{f} \tag{8.4}$$

where L is the graph Laplacian.
$L = D - W$ where D is a diagonal matrix and $D_{ii} = \sum_j W_{ij}$
W is the adjacency matrix. W is a symmetric matrix.

$f : R^k \to R$ take $f \to \bigtriangleup f = g$
$g = \bigtriangleup f = \sum_{i=1}^{k} \frac{\partial^2 f}{\partial x_i}$
For any $n \times n$ matrix A can act on f and produce a set of n number which can be inteperted as a function of $f : V \to R$ $|V| = n$

$$\begin{aligned}
&\sum_{i,j} f_i^2 + f_j^2 - 2 f_i f_j W_{ij} \\
&= \sum_i f_i^2 \sum_j W_{ij} + \sum_j f_j^2 \sum_i W_{ij} \\
&= \sum_i f_i^2 D_{ii} + \sum_j f_j^2 D_{jj} - 2 \sum_{i,j} f_i f_j W_{ij} \\
&So\ 2f^T D f - 2 f^T W f = 2(f^T L f)
\end{aligned}$$

Therefore, the task of finding a good cut is reduced to finding

$$\min_{f:v \to [-1,+1]} \frac{1}{4} 2 f^T L f$$

over all choices of f. So we need to find

$$\min_{f:v \to R} \frac{f^T L f}{f^T f} \tag{8.5}$$

Now let's look at L. L is (i) symmetric, (ii) positive semidefinite $\forall V \neq 0$
$V^T \cdot L \cdot V \geq 0$
Let's go over the analysis of $LV = \lambda V$. $\lambda_i \geq 0$ $V_i$are or diagonal.
$LV_i = \lambda_i V_i$ where λ_i is eigenvalue, V_i is eigenvector.

$$V_j^t L V_i = \lambda_i V_j^T V_i$$
$$V_i^T L V_j = \lambda_j V_i^T V_j$$
$$0 = (\lambda_i - \lambda_j)(V_i^T V_j)$$

Here the lambdas are distinct from each other. Let's say $\lambda_j < 0$, for V_j, we have

$$V_j^T L V_j = V_j^T \lambda_j V_j = \lambda_j (V_j^T V_j)$$

for λ, $LV_1 = \lambda V_1$.
Given V_k, $\lambda_1 ... \lambda_m$ and $k_1 .... k_m$ $\sum k_i = n$ and $H_{k1} ... H_{k2}$ where $\lambda_1 < \lambda_2 < \lambda_3 < ... < \lambda_n$
for $V_1 ..... V_n$, we have $f = \sum\limits_{i=1}^{n} \alpha_i V_i$. Any f can be written like this.
Top term of (8.5):

$$\begin{aligned}
&f^T L f \\
&= f^T (L(\sum_{i=1}^{n} \alpha_i V_i)) \\
&= f^T (\sum_{i=1}^{n} \alpha_i L V_i) \\
&= f^T (\sum_{i=1}^{n} \alpha_i \lambda_i V_i) \\
&= (\sum_{j=1}^{n} \alpha_j V_j^T)(\sum_{i=1}^{n} \alpha_i \lambda_i V_i) \\
&= \sum_{i=1}^{n} \alpha_i^2 \lambda_i
\end{aligned}$$

Bottom term of (8.5):

$$\begin{aligned}
f^T f = (\sum \alpha_i V_i)^T (\sum \alpha_j V_j) &= \sum \alpha_i^2 \\
\min_{f: v \to R \ f^T f = 1} \frac{f^T L f}{f^T f} & \\
f &= \sum \alpha_i V_i \\
f^T f &= \sum \alpha_i^2 = 1
\end{aligned}$$

This means,

$$\min(\sum \alpha_i^2 \cdot \lambda_i)$$
$$\sum \alpha_i^2 = 1$$

If choose a series of α as simple as possible, $\alpha_1 = 1, : \alpha_2 ...... = 0$, for L, take its eigenvalue, $\lambda_1 = 0 \; V_1 = \dfrac{1}{\sqrt{n}} 1$
LC = 0
L = (D-W)C
L applies a constant function on the graph.

L is very very beautiful. You can do it on graphs, functions, medical, on everything. However, if we follow this, we will get a constant function which doesn't give me a cut for everything is either positive or negative. We would put another constraint:

$$f^T 1 = 0$$

which means half of the graph is +1 and half of it is -1. It will ensure to give me a cut.

$$f^T 1 = (\sum \alpha_i V_i^T) 1 = \alpha_1 (V_1^T \cdot 1)$$
$$f = V_2$$

α_1 better to be zero.

To do the graph based clustering, we start a graph, we make it nearest neighbour graph. Then we make its adjancency matrix W. We caculate the D-W matrix and we take the second eigenvalue. This is equal to start with A and $\bar{A}$ and search all pair of combinations. What is true for the graph is true for the laplacian function.

Open question:
We have seen the 2 patitioning of a graph. How will you do an n partitioning of a graph?