

Lecture 19: Expectation Maximization. Hierarchical Clustering

- Expectation Maximization in general
- Mixtures of Gaussians
- Hierarchical clustering methods

Recall: Missing values

- Suppose we have a model of the probability distribution of the data with parameters θ , and the values y are missing in some (or all) of the instances

- The likelihood of the data can be written as:

$$\log L(\theta) = \sum_{\text{complete data}} \log P(\mathbf{x}_i, y_i | \theta) + \sum_{\text{incomplete data}} \log P(\mathbf{x}_i | \theta)$$

- For the second term, we must consider all possible values for y :

$$\sum_{\text{incomplete data}} \log P(\mathbf{x}_i | \theta) = \sum_{\text{incomplete data}} \log \left(\sum_y P(\mathbf{x}_i, y | \theta) \right)$$

- In our problem, y is never observed, so we only have the second term

Recall: Expectation Maximization (EM)

- A general purpose method for learning from incomplete data
- Main idea:
 - If we had complete data we could easily maximize the likelihood
 - But because the data is incomplete, we get a summation inside the \log , which makes the optimization much harder
 - So in the case of missing values, we will “fantasize” what they should be, based on the current parameter setting
 - In other words, we fill in the missing values based on our current expectation
 - Then we compute new parameters, which maximize the likelihood of the completed data

Important example: Gaussian mixture models

- Suppose that all inputs \mathbf{x} were real-valued, and we have some number of classes c_1, \dots, c_K with $K \geq 2$
- Assume a model similar with naive Bayes, but with no independence assumptions: class label c_i determines the probability distribution of the instances with that class, and this distribution is a multivariate Gaussian
- The parameters of the model are the probabilities of different classes, $P(c_k)$, and the parameters of the corresponding Gaussian distributions, μ_k, Σ_k
- The best parameters should maximize the likelihood of the data:

$$\log L = \sum_i (\log P(y_i) + \log P(\mathbf{x}_i|y_i))$$

Maximum likelihood solution

- Let $\delta_{i,k} = 1$ if $y_i = c_k$ and 0 otherwise
- The class probabilities are determined by the empirical frequency of examples in each class:

$$P(c_k) = \frac{\sum_i \delta_{i,k}}{\sum_k \sum_i \delta_{i,k}}$$

- The mean and covariance matrix for class k are the empirical mean and covariance of the examples in that class:

$$\mu_{\mathbf{k}} = \frac{\sum_i \delta_{i,k} \mathbf{x}_i}{\sum_i \delta_{i,k}}$$
$$\Sigma_k = \frac{\sum_i \delta_{i,k} (\mathbf{x}_i - \mu_{\mathbf{k}})(\mathbf{x}_i - \mu_{\mathbf{k}})^T}{\sum_i \delta_{i,k}}$$

Gaussian classifiers in practice

- Note that there are many parameters, and estimating the covariance matrices may be prone to overfitting
- Independence assumptions can be used to reduce the number of non-zero entries in the covariance matrix
- As seen in the homework, these classifiers have a strong assumption; to address this, several Gaussians can be used for every class.

Mixture models more generally

- The term **mixture model** is typically used to describe a generative model in which there are several components, selected with some probability
- Each component itself is a probability distribution, and is usually described in parametric form
- For classification, the components usually correspond to the different classes, although we can assume several components for each class as well
- For regression, the components usually cover different parts of the input space
- Clustering can be handled using a classification mixture model, assuming the labels on the instances are erased

EM for Mixture of Gaussians

- We start with an initial guess for the parameters $P(c_k), \mu_k, \Sigma_k$
- We will iterate an **expectation step (E-step)**, in which we “complete” the data, and a **maximization step (M-step)**, in which we re-compute the parameters
- In the “hard EM” version, completing the data means that each data point is assumed to be generated by exactly one Gaussian
- This is roughly equivalent to the setting of K -means clustering
- In the “soft EM” version (also usually known as EM), we assume that each data point could have been generated from any component
- In this case, each point will contribute to the mean and variance estimate of each component.

Hard EM for Mixture of Gaussians (1)

1. Guess an initial parameter setting

$$\theta_j = \langle p_j, \mu_j, \Sigma_j \rangle, j = 1 \dots K$$

2. Repeat until convergence:

- (a) E-step: For each $i = 1, \dots, m$ and each $j = 1, \dots, K$, compute the probability of \mathbf{x}_i being drawn from the distribution of class j and assign the instance to the most likely class

$$k_i = \arg \max_j p(\mathbf{x}_i | \theta_j) \text{ where } p(\mathbf{x}_i | p_j, \mu_j, \Sigma_j) \propto p_j p(\mathbf{x}_i | \mu_j, \Sigma_j)$$

This corresponds to the assignment of instances to clusters in K -means

Hard EM for Mixture of Gaussians (1)

- (b) M-step: Update the parameters of the model to maximize the likelihood of the data

$$p_j = \frac{1}{m} \sum_{i=1}^m \delta_{ij} \quad \mu_j = \frac{\sum_{i=1}^m \delta_{ij} \mathbf{x}_i}{\sum_{i=1}^m \delta_{ij}}$$
$$\Sigma_j = \frac{\sum_{i=1}^m \delta_{ij} (\mathbf{x}_i - \mu_j) (\mathbf{x}_i - \mu_j)^T}{\sum_{i=1}^m \delta_{ij}}$$

This corresponds to re-computing the cluster centers in K -means

What if an example is roughly at the same distance from all clusters?

Soft EM for Mixture of Gaussians

1. Guess an initial parameter setting

$$\theta_j = \langle p_j, \mu_j, \Sigma_j \rangle, j = 1 \dots K$$

2. Repeat until convergence:

- (a) E-step: For each $i = 1, \dots, m$ and each $j = 1, \dots, K$, compute the probability of \mathbf{x}_i being drawn from the distribution of class j :

$$w_{ij} = p(\mathbf{x}_i | p_j, \mu_j, \Sigma_j) \propto p_j p(\mathbf{x}_i | \mu_j, \Sigma_j)$$

- (b) M-step: Update the parameters of the model to maximize the likelihood of the data

$$p_j = \frac{1}{m} \sum_{i=1}^m w_{ij} \quad \mu_j = \frac{\sum_{i=1}^m w_{ij} \mathbf{x}_i}{\sum_{i=1}^m w_{ij}}$$
$$\Sigma_j = \frac{\sum_{i=1}^m w_{ij} (\mathbf{x}_i - \mu_j) (\mathbf{x}_i - \mu_j)^T}{\sum_{i=1}^m w_{ij}}$$

EM in general

- Whenever we are trying to model data drawn probabilistically, and we have missing values in the data, EM is an option (both for missing labels and missing attributes)
- We need some structured or parametric form of the distribution (we saw mixtures of Gaussians as examples)
- We starts with a guess for the parameters of the distribution
- You can think of the E-step as trying to “complete” the data, by filling in the missing values
- The M-step will compute new parameters, given the completed data

Comparison of hard EM and soft EM

- Soft EM does not commit to a particular value of the missing item. Instead, it considers all possible values, with some probability
- This is a pleasing property, given the uncertainty in the value
- The complexity of the two versions is the same:
 - Hard EM requires computing most probable values
 - Soft EM requires computing conditional probabilities for completing the missing values
- Soft EM is almost always the method of choice (and often when people say “EM”, they mean the soft version)

Theoretical properties of EM

- Each iteration improves the likelihood:

$$L(\theta_{i+1}|D) \geq L(\theta_i|D)$$

- If the parameters do not change in one iteration, $\theta_{i+1} = \theta_i$, then the gradient of the log-likelihood function is 0 at θ_i :

$$\frac{\partial L(\theta|D)}{\partial \theta}(\theta_i) = 0$$

This means that θ_i is a min, max or saddle point

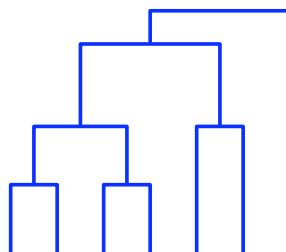
- In practice, convergence only occurs at local maxima

Variations

- If only some of the data is incomplete, the likelihood will have one component based on the complete instances and another ones based on incomplete instances
- Sparse EM: Only compute probability at a few data points (most values will be close to 0 anyway)
- EM can be used if you have labeled data but which is possibly unreliable, in order to get better labels.
- Instead of a complete M-step, just improve the likelihood a bit
- Note that EM can be stuck in *local minima*, so it has to be restarted!
- It works very well for low-dimensional problems, but can have problems if θ is high-dimensional.

Hierarchical clustering

- Organizes data instances into trees.
- For visualization, exploratory data analysis.
- **Agglomerative methods** build the tree bottom-up, successively grouping together clustering deemed most similar.
- **Divisive methods** build the tree top-down, recursively partitioning the data.



What is a hierarchical clustering?

- Given instances $D = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$.
- A hierarchical clustering is a set of subsets (clusters) of D , $C = \{C_1, \dots, C_K\}$, where
 - $D \in C$
 - The C_j can be assigned to the nodes of a tree such that the cluster at any node is precisely the union of the clusters at the node's children (if any).

Example of a hierarchical clustering

- Suppose $D = \{1, 2, 3, 4, 5, 6, 7\}$.
- One hierarchical clustering is $C = \{\{1\}, \{2, 3\}, \{4, 5\}, \{1, 2, 3, 4, 5\}, \{6, 7\}, \{1, 2, 3, 4, 5, 6, 7\}\}$.
- Leaves of the tree need not correspond to single instances.
- The branching factor of the tree is not limited.
- However, most hierarchical clustering algorithms produce binary trees, and take single instances as the smallest clusters.

Agglomerative clustering

- Inputs: A set of instances, and pairwise distances $d(\mathbf{x}, \mathbf{x}')$ between them.
- Outputs: A hierarchical clustering
- Algorithm:
 - Assign each instance as its own cluster on a working list W .
 - Repeat
 - * Find the two clusters in W that are most “similar”.
 - * Remove them from W .
 - * Add their union to W .
 - Until W contains a single cluster with all the data objects.
 - The hierarchical clustering contains *all* clusters appearing in W at any stage of the algorithm.

How do we measure dissimilarity between clusters?

- Distance between nearest objects (“Single-linkage” agglomerative clustering, or “nearest neighbor”):

$$\min_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

- Distance between farthest objects (“Complete-linkage” agglomerative clustering, or “furthest neighbor”):

$$\max_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

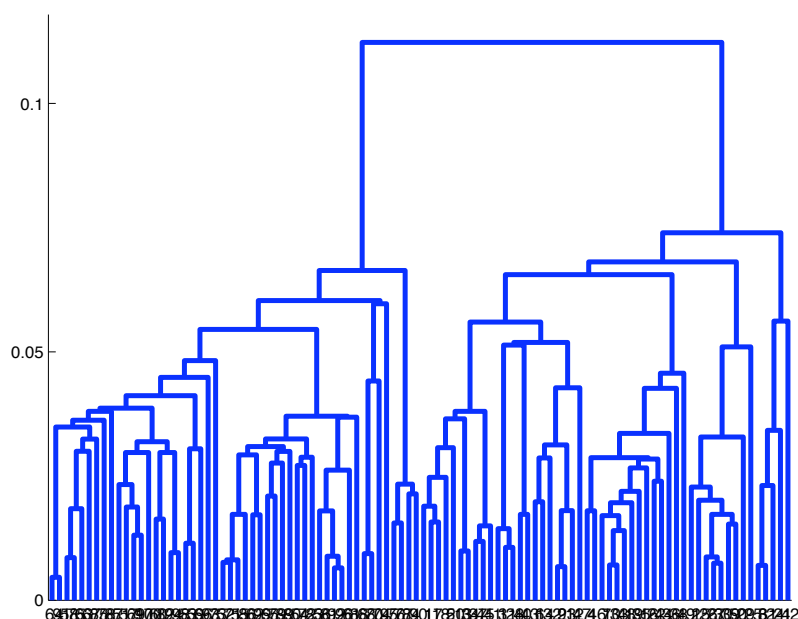
- Average distance between objects (“Group-average” agglomerative clustering):

$$\frac{1}{|C||C'|} \sum_{\mathbf{x} \in C, \mathbf{x}' \in C'} d(\mathbf{x}, \mathbf{x}')$$

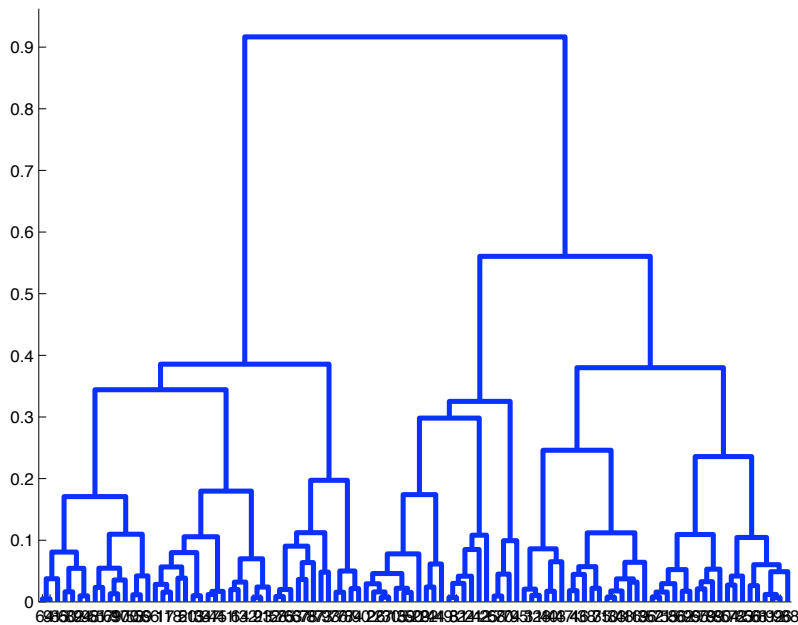
Dendrograms and Monotonicity

- Single-linkage, complete-linkage and group-average dissimilarity measure all share a monotonicity property:
 - Let A, B, C be clusters.
 - Let d be one of the dissimilarity measures.
 - If $d(A, B) < d(A, C)$ and $d(A, B) < d(B, C)$, then $d(A, B) < d(A \cup B, C)$.
- Implication: every time agglomerative clustering merges two clusters, the dissimilarity of those clusters is \geq the dissimilarity of all previous merges.
- Dendrograms (trees depicting hierarchical clusterings) are often drawn so that the height of a node corresponds to the dissimilarity of the merged clusters.

Example: Dendrogram for single-linkage clustering



Example: Dendrogram for complete-linkage clustering

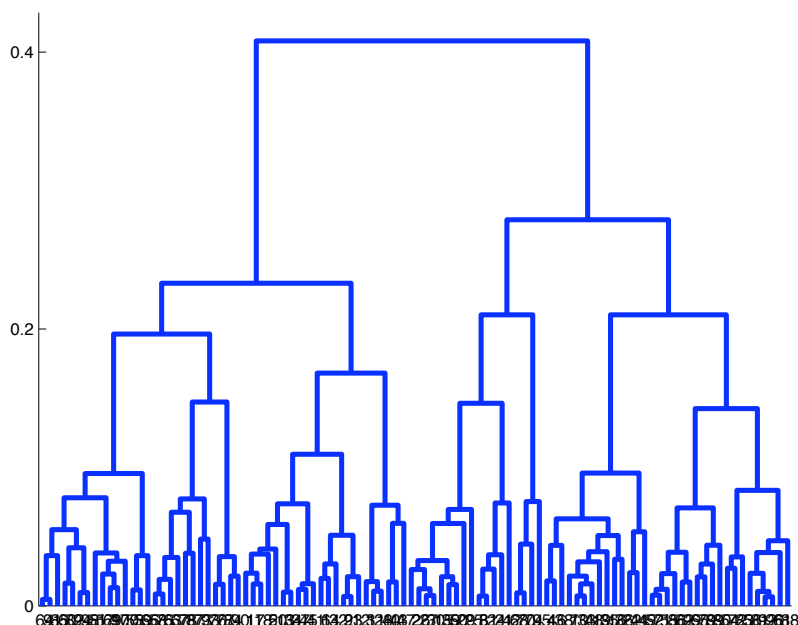


November 17, 2005

23

COMP-652 Lecture 19

Example: Dendrogram for average-linkage clustering



November 17, 2005

24

COMP-652 Lecture 19

Remarks

- We can form a flat clustering by cutting the tree at any height.
- Jumps in the height of the dendrogram can suggest natural cutoffs.

Divisive clustering

- Works by recursively partitioning the instances.
- But dividing such as to optimize one of the agglomerative criteria is computationally hard!
- Many heuristics for partitioning the instances have been proposed . . . but many violate monotonicity, making it hard to draw dendrograms.