Machine Learning: Lecture 10

Unsupervised Learning

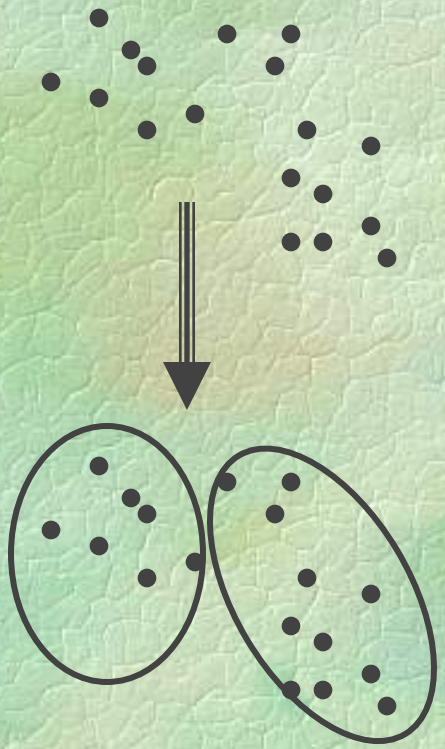
(Based on Chapter 9 of Nilsson, N., Introduction to Machine Learning, 1996)

Overview

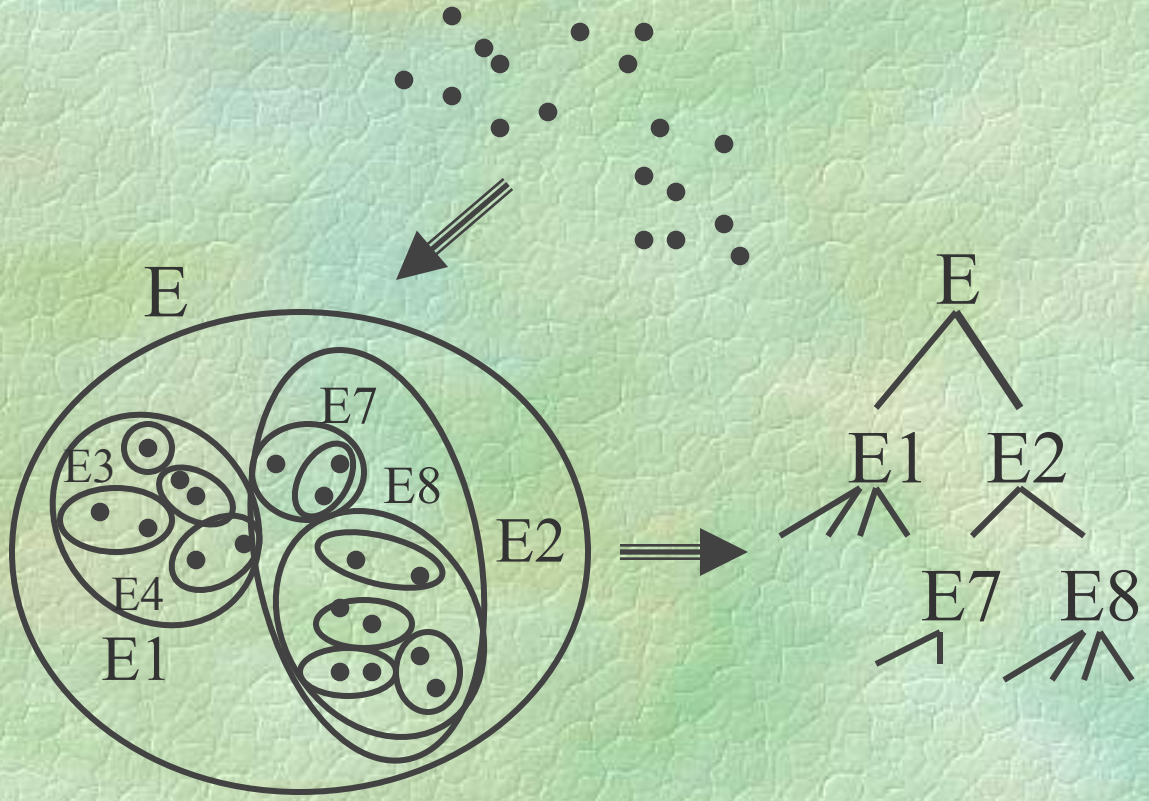
- So far, in all the learning techniques we considered, a training example consisted of a set of attributes (or features) and either a class (in the case of classification) or a real number (in the case of regression) attached to it.
- Unsupervised Learning takes as training examples the set of attributes/features alone.
- The purpose of unsupervised learning is to attempt to find natural partitions in the training set.
- Two general strategies for Unsupervised learning include: ***Clustering*** and ***Hierarchical Clustering***.

Clustering and Hierarchical Clustering

Clustering



Hierarchical Clustering



What is Unsupervised Learning Useful for?(I)

- Collecting and labeling a large set of sample patterns can be very expensive. By designing a basic classifier with a small set of labeled samples, and then tuning the classifier up by allowing it to run without supervision on a large, unlabeled set, much time and trouble can be saved.
- Training with large amounts of often less expensive, unlabeled data, and then using supervision to label the groupings found. This may be used for large "data mining" applications where the contents of a large database are not known beforehand.

What is Unsupervised Learning Useful for?(II)

- Unsupervised methods can also be used to find features which can be useful for categorization. There are unsupervised methods that represent a form of data-dependent "smart pre-processing" or "smart feature extraction."
- Lastly, it can be of interest to gain insight into the nature or structure of the data. The discovery of similarities among patterns or of major departures from expected characteristics may suggest a significantly different approach to designing the classifier.

Clustering Methods I: A Method Based on Euclidean Distance

Suppose we have R randomly chosen ***cluster seekers*** $C_1, \ldots, C_R$. (During the training process, these points will move towards the center of one of the clusters of patterns)

- For each pattern X_i, presented,
  - Find the cluster seeker C_j that is closest to X_i
  - Move C_j closer to X_i as follows:

$$C_j \leftarrow (1-\alpha_j)C_j + \alpha_j X_i$$

where α_j is a learning rate parameter for the ***j-th*** cluster seeker which determines how far C_j is moved towards X_i

Clustering Methods I: A Method Based on Euclidean Distance

- It might be useful to make the cluster seekers move less far as training proceeds.
- To do so, let each cluster seeker have a mass, m_j, equal to the number of times it has moved.
- If we set $\alpha_j = 1/(1+m_j)$, it can be seen that a cluster seeker is always at the center of gravity (sample mean) of the set of patterns towards which it has so far moved.
- Once the cluster seekers have converged, the implied classifier can be based on a ***Voronoi*** partitioning of the space (based on the distances to the various cluster seekers)
- Note that it is important to ***normalize*** the pattern features.

Clustering Methods I: A Method Based on Euclidean Distance

- The number of cluster seekers can be chosen ***adaptively*** as a function of the ***distance*** between them and the ***sample variance*** of each cluster.

Examples:

- If the distance d_{ij} between two cluster seekers C_i and C_j ever falls below some threshold ε, then merge the two clusters.
- If the sample variance of some cluster is larger than some amount δ, then split the clusters into two separate clusters by adding an extra cluster seeker.

Clustering Methods II: A Method Based on Probabilities

Given a set of unlabeled patterns, an empty list, L, of clusters, and a measure of similarity between an instance X and a cluster C_i, $S(X,C_i)=p(x_1|C_i) \ldots p(x_n|C_i)p(C_i)$, $X=<x_1,\ldots x_n>$

- For each pattern X (one at a time) {
  - Compute $S(X,C_i)$ for each cluster, C_i. Let $S(X, C_{max})$ be the largest.
    - If $S(X,C_{max}) > \delta$, assign X to C_{max} and update the sample statistics
    - Otherwise, create a new cluster $C_{new}=\{X\}$ and add C_{new} to L
  - Merge any existing clusters C_i and C_j if $(M_i-M_j)^2 <\varepsilon$ [Mi, Mj are the sample means]. Compute the new sample statistics for the new cluster C_{merge}
  - If the sample statistics did not change, then return L. }

Hierarchical Clustering I: A Method Based on Euclidean Distance

Agglomerative Clustering

1. Compute the Euclidean Distance between every pair of patterns. Let the smallest distance be between patterns Xi and Xj.
2. Create a cluster C composed of Xi and Xj. Replace Xi and Xj by cluster vector C in the training set (the average of Xi and Xj).
3. Go back to 1, treating clusters as points, though with an appropriate weight, until no point is left.

Hierarchical Clustering II: A Method Based on Probabilities

A probabilistic quality measure for partitions

- Given a partitioning of the training set into R classes $C_1,..., C_R$ and given that each component of a pattern $X=<x_1,..,x_n>$ can take values v_{ij} (where i is the component number and j, the different values that that component can take),
- If we use the following probability measure for ***guessing the i^{th} component correctly given that it is in class k***:

$$\Sigma_j[p_i(v_{ij}|C_k)]^2$$

where $p_i(v_{ij}|C_k)$=*probability*$(x_i=v_{ij}|C_k)$, **then**

- The ***average number of components whose values are guessed correctly*** is: $\Sigma_i\Sigma_j[p_i(v_{ij}|C_k)]^2$
- The ***goodnes measure of this partitioning*** is

$$\Sigma k\ p(Ck)\ \Sigma_i\Sigma_j[p_i(v_{ij}|C_k)]^2$$

- The ***final measure of goodness*** is:

$$(1/R)\Sigma k\ p(Ck)\ \Sigma_i\Sigma_j[p_i(v_{ij}|C_k)]^2$$

Hierarchical Clustering II: A Method Based on Probabilities

COBWEB

1. Let us start with a tree whose root node contains all the training patterns and has a single empty successor.

2. Select a pattern X_i (if there are no more pattern to select, then terminate).

3. Set μ to the root node.

4. For each of the successors of μ, calculate the best host for X_i, as a function of the Z value of the different potential partitions.

5. If the best host is an empty node η, then place X_i in η and generate an empty successor and sibling of η. Go to 2.

6. If the best host η is a non-empty singleton, then place X_i in η, generate successors $\{X_i\}$ and previous η to the new η, add empty successors everywhere, and go to 2.

7. If the best host is a non-empty, non-singleton node, η, place X_i in η, set μ to η, and go to 4.

Hierarchical Clustering II: A Method Based on Probabilities

Making COBWEB less order dependent

- **Node Merging**:
  - Attempt to merge the two best hosts
  - If merging improves the Z value, then do so.
- **Node Splitting:**
  - Attempt to replace the best host among a group of sibling by that hosts's successors.
  - If splitting improves the Z value, then do so.

Other Unsupervised Methods:

- There are a lot of other Unsupervised Learning Methods.
- **Examples:**
  - k-means
  - The EM Algorithm
  - Competitive Learning
  - Kohonen's Neural Networks: Self-Organizing Maps
  - Principal Component Analysis, Autoassociation