Lecture 14: Unsupervised learning I

- **Supervised Vs. unsupervised learning**
- **Flat clustering algorithms**
  - k-means
  - ISODATA
- **Hierarchical clustering algorithms**
  - Divisive
  - Agglomerative

Supervised Vs. unsupervised learning

- **The pattern recognition systems covered until now assume the definition of pattern we proposed in the first lecture**
  - Pattern is a pair of variables $\{x,\omega\}$ where
    - **x** is a collection of observations or features (feature vector)
    - ω is the concept behind the observation (label)
  - Such a pattern recognition system is called supervised, since the system is given BOTH the feature vector and the label (correct answer)
- **Here we will investigate a number of techniques which use unlabeled data: a collection of feature vectors $X=\{x^{(1}, x^{(2}, \ldots, x^{(N}\}$ without the class labels ω_i**
  - These methods are called unsupervised since they are not provided the correct answer
  - Although unsupervised learning methods may appear to have limited capabilities, there are several reasons that make them extremely useful
    - Labeling large data sets can be a costly procedure (i.e., speech recognition)
    - Class labels may not be known beforehand (i.e., data mining)
    - Large datasets can be compressed by finding a small set of prototypes (i.e., k-NNR)
- **There are two major approaches for unsupervised learning**
  - Parametric (mixture modeling)
    - Functional forms for the underlying class-conditional densities are assumed, and we must estimate the parameters

$$P(x \mid \theta) = \sum_{i=1}^{C} P(x_i \mid \omega_i, \theta_i) P(\omega_i)$$

      - This problem will not be covered in the course (For details refer to [Duda and Hart, 1973])
  - Non-parametric (clustering)
    - No assumptions are made about the underlying densities, instead we seek a partition of the data into clusters
      - These methods are typically referred to as clustering, and will be the subject of these lectures

Similarity measures and criterion function

- **Non-parametric unsupervised learning is concerned with finding natural groupings (clusters) in a dataset, which involves three steps**
  - Defining a measure of similarity between examples
  - Defining a criterion function for clustering
  - Defining an algorithm to minimize (or maximize) the criterion function
- **Similarity measures**
  - The most straightforward similarity measure is distance: Euclidean, city block or Mahalanobis
    - Euclidean and city block measures are very sensitive to scaling of the axis, so caution must be exercised
  - In general, we can define a non-metric similarity function s(x,y) whose value will be large when x and y are "similar", for example:
    - The Tanimoto distance is commonly used for binary-valued features $s(x,y)=\frac{x^T y}{x^T x + y^T y - x^T y}$
- **Criterion function**
  - Once a similarity measure has been determined, we need to define a criterion function to be optimized
  - The most widely used criterion function for clustering is the sum-of-square-error criterion

$$J_{MSE}=\sum_{i=1}^{C}\sum_{x\approx\omega_i}\left|x-\mu_i\right|^2 \qquad \text{where } \mu_i=\frac{1}{N_i}\sum_{x\approx\omega_i}x$$

    - This criterion measures how well the data set $X=\{x^{(1}, x^{(2}, \ldots, x^{(N}\}$ is represented by the cluster centers $\mu=\{\mu^{(1}, \mu^{(2}, \ldots, \mu^{(N}\}$
    - Clustering methods that use this criterion are called minimum variance methods
  - Other criterion functions exist, based on the scatter matrices used in Linear Discriminant Analysis

Iterative optimization

- **Once a criterion function has been defined, we must find a partition of the data set that minimizes the criterion**
  - Exhaustive enumeration of all partitions, which guarantees the optimal solution, is unfeasible
  - For example, a problem with 5 clusters and 100 examples yields 10^{67} partitionings
- **The common approach is to proceed in an iterative fashion**

  (1) find some reasonable initial partition and then

  (2) move samples from one cluster to another in order to reduce the criterion function
- **These iterative methods produce sub-optimal solution but are computationally tractable**
- **We will consider two groups of iterative methods**
  - Flat clustering algorithms
    - These algorithms produce a set of disjoint clusters
    - Two algorithms are widely used: k-means and ISODATA
  - Hierarchical clustering algorithms:
    - The result is a hierarchy of nested clusters
    - These algorithms can be broadly divided into agglomerative and divisive approaches

The k-means algorithm

- **The k-means algorithm is a simple clustering procedure that attempts to minimize the criterion function J_{MSE} in an iterative fashion**

$$J_{MSE} = \sum_{i=1}^{C} \sum_{x \approx \omega_i} \left| x - \mu_i \right|^2 \qquad \text{where } \mu_i = \frac{1}{N_i} \sum_{x \approx \omega_i} x$$

1. Define the number of clusters
2. Initialize clusters by
   - an arbitrary assignment of examples to clusters or
   - an arbitrary set of cluster centers (examples assigned to nearest centers)
3. Compute the sample mean of each cluster
4. Reassign each example to the cluster with the nearest mean
5. If the classification of all samples has not changed, stop, else go to step 3

- **The k-means algorithm is widely used in the fields of signal processing and communication for Vector Quantization**
  - Unidimensional signal values are usually quantized into a number of levels (typically a power of 2 so the signal can be transmitted or stored in binary)
  - The same idea can be extended for multiple channels
    - However, rather than quantizing each separate channel, we can obtain a more efficient signal coding if we quantize the overall multidimensional vector by finding a number of multidimensional prototypes (cluster centers)
  - The set of cluster centers is called a “codebook”, and the problem of finding this codebook is normally solved using the k-means algorithm

(Supervised) Learning Vector Quantization

- **Rather than updating the codebook after presentation of all the examples, an on-line updating rule could be adopted**

$$\mu_c = \mu_c + \alpha(t)\left[x^{(i} - \mu_c\right] \quad \text{if } \mu_c \text{ is closest to } x^{(i}$$
$$\mu_i = \mu_i \quad \text{otherwise}$$

  - where $\alpha(t)$ is a learning rate ($0<\alpha(t)<1$) that decreases monotonically with each iteration to ensure convergence
- **This update rule is the basis of a collection of <u>supervised</u> methods, made popular by Teuvo Kohonen, known as Learning Vector Quantization (LVQ)**
  - The original algorithm, known as LVQ1, is summarized as follows

1. For each example $x^{(i}$
   - Find the closest cluster center. Denote this center by μ_c
   - Update cluster centers as follows

$$\mu_c = \begin{cases} \mu_c + \alpha(t)\left[x^{(i} - \mu_c\right] & \text{if } x \text{ classified correctly} \\ \mu_c - \alpha(t)\left[x^{(i} - \mu_c\right] & \text{if } x \text{ classified incorrectly} \end{cases}$$

$$\mu_j = \mu_j \quad \text{for } j \neq c$$

2. Update $\alpha(t)$ (linear decrease in LVQ1)
3. Go to step 1

  - This update rule is rather intuitive
    - Correct classification of example $x^{(i}$ moves the cluster center μ_c in the direction of $x^{(i}$, whereas incorrect classification moves μ_c in the opposite direction
    - Cluster centers μ_i not close to $x^{(i}$ are not altered

ISODATA (1)

- **ISODATA, which stands for Iterative Self-Organizing Data Analysis Technique (Algorithm) is an extension to the k-means algorithm with some heuristics to automatically select the number of clusters**
- **ISODATA requires the user to select a number of parameters**
  - N_{MIN_EX} minimum number of examples per cluster
  - N_D desired (approximate) number of clusters
  - σ_S^2 maximum spread parameter for splitting
  - D_{MERGE} maximum distance separation for merging
  - N_{MERGE} maximum number of clusters that can be merged
- **The algorithm works in an iterative fashion**
  - (1) Perform k-means clustering
  - (2) Split any clusters whose samples are sufficiently dissimilar
  - (3) Merge any two clusters sufficiently close
  - (4) Go to (1)
- **A more detailed description of the algorithm follows**

ISODATA (2)

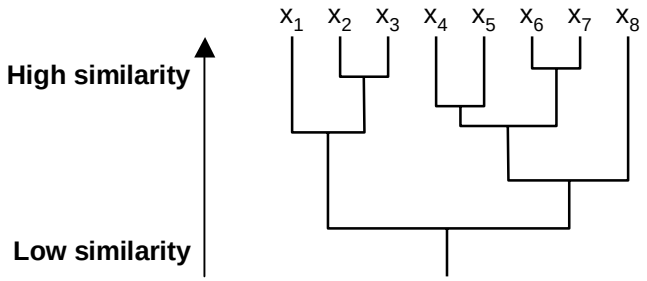
1. Select an initial number of clusters N_C and use the first N_C examples as cluster centers μ_k, k=1..N_C
2. Assign each example to a cluster according to each one's minimum distance to a cluster center
   a. Exit the algorithm if the classification of examples has not changed
3. Eliminate clusters that contain less than N_{MIN_EX} examples and
   a. Assign those examples to the other clusters using minimum distance to cluster centers
   b. Decrease N_C accordingly
4. For each cluster k,
   a. Compute the center μ_k as the sample mean of all the examples assigned to that cluster
   b. Compute the average distance between examples and cluster centers $d_{AVG} = \frac{1}{N}\sum_{k=1}^{N_C} N_k d_k$ and $d_k = \frac{1}{N_k}\sum_{x \approx \omega_k} |x - \mu_k|$
   c. Compute the variance of each axis and find the axis n* with maximum variance $\sigma_k^2(n^*)$
6. For each cluster k with $\sigma_k^2(n^*) > \sigma_S^2$, if $\{d_k > d_{AVG}$ and $N_k > 2N_{MIN\_EX}+1\}$ or $\{N_C < N_D/2\}$
   a. Split that cluster into two clusters where the two centers μ_{k1} and μ_{k2} differ only in the coordinate n*
      i. $\mu_{k1}(n^*) = \mu_k(n^*) + \sigma_k(n^*)/2$ (all other coordinates remain the same)
      ii. $\mu_{k2}(n^*) = \mu_k(n^*) - \sigma_k(n^*)/2$ (all other coordinates remain the same)
   b. Increment N_C accordingly
   c. Reassign the cluster's examples to one of the two new clusters based on minimum distance to cluster centers
7. If $N_C > 2N_D$ then
   a. Compute all distances $D_{ij} = d(\mu_i, \mu_j)$
   b. Sort D_{ij} in decreasing order
   b. For each pair of clusters sorted by D_{ij}, if (1) neither cluster has been already merged, (2) the distance D_{ij} satisfies $D_{ij} < D_{MERGE}$ and (3) not more than N_{MERGE} pairs of clusters have been merged in this loop, then
      i. Merge i^{th} and j^{th} clusters
      ii. Compute the cluster center $\mu' = \frac{N_i\mu_i + N_j\mu_j}{N_i + N_j}$
      iii. Decrement N_C accordingly
8. Go to step 1

ISODATA (3)

- **ISODATA has been shown to be an extremely powerful heuristic**
- **Some of its advantages are**
  - Self-organizing capabilities
  - Flexibility in eliminating clusters that have very few examples
  - Ability to divide clusters that are too dissimilar
  - Ability to merge clusters that are sufficiently similar
- **However, it suffers from the following limitations**
  - Data must be linearly separable (long narrow or curved clusters are not handled properly)
  - It is difficult to know a priori the “optimal” parameters
  - Performance is highly dependent on these parameters
  - For large datasets and large number of clusters, ISODATA is less efficient than other linear methods
  - Convergence is unknown, although it appears to work well for non-overlapping clusters
- **In practice, ISODATA is run multiple times with different values of the parameters and the clustering with minimum sum-squared error would be selected**

Hierarchical clustering

- **k-means and ISODATA create disjoint clusters, resulting in a "flat" data representation**
- **However, sometimes it is desirable to obtain a hierarchical representation of data, with clusters and sub-clusters arranged in a tree-structured fashion**
  - Hierarchical representations are commonly used in the sciences (i.e., biological taxonomy)
- **Hierarchical clustering methods can be grouped in two general classes**
  - Agglomerative (bottom-up, merging):
    - Starting with N singleton clusters, successively merge clusters until one cluster is left
  - Divisive (top-down, splitting)
    - Starting with a unique cluster, successively split the clusters until N singleton examples are left
- **The preferred representation for hierarchical clusters is the dendrogram**
  - The dendrogram is a binary tree that shows the structure of the clusters
    - In addition to the binary tree, the dendrogram provides the similarity measure between clusters (the vertical axis)
  - An alternative representation is based on sets: $\{\{x_1, \{x_2, x_3\}\}, \{\{\{x_4, x_5\}, \{x_6, x_7\}\}, x_8\}\}$ but, unlike the dendrogram, sets cannot express quantitative information



Divisive clustering

- **Outline**
  - Define
    - N_C: Number of clusters
    - N_{EX}: Number of examples

1. Start with one large cluster
2. Find “worst” cluster
3. Split it
4. If $N_C<N_{EX}$ go to 1

- **How to choose the “worst” cluster**
  - Largest number of examples
  - Largest variance
  - Largest sum-squared-error
  - ...
- **How to split clusters**
  - Mean-median in one feature direction
  - Perpendicular to the direction of largest variance
  - …
- **The computations required by divisive clustering are more intensive than for agglomerative clustering methods and, thus, agglomerative approaches are more common**

Agglomerative clustering (1)

- **Outline**
  - Define
    - N_C: Number of clusters
    - N_{EX}: Number of examples

1. Start with N_{EX} singleton clusters
2. Find nearest clusters
3. Merge them
4. If N_C>1 go to 1

- **How to find the "nearest" pair of clusters**
  - Minimum distance $d_{min}(\omega_i,\omega_j)=\min_{\substack{x\in\omega_i\\y\in\omega_j}}\|x-y\|$
  - Maximum distance $d_{max}(\omega_i,\omega_j)=\max_{\substack{x\in\omega_i\\y\in\omega_j}}\|x-y\|$
  - Average distance $d_{avg}(\omega_i,\omega_j)=\frac{1}{N_iN_j}\sum_{x\in\omega_i}\sum_{y\in\omega_j}\|x-y\|$
  - Mean distance $d_{mean}(\omega_i,\omega_j)=\|\mu_i-\mu_j\|$

Agglomerative clustering (2)

- **Minimum distance**
  - When d_{min} is used to measure distance between clusters, the algorithm is called the nearest-neighbor or single-linkage clustering algorithm
  - If the algorithm is allowed to run until only one cluster remains, the result is a minimum spanning tree (MST)
  - This algorithm favors elongated classes
- **Maximum distance**
  - When d_{max} is used to measure distance between clusters, the algorithm is called the farthest-neighbor or complete-linkage clustering algorithm
  - From a graph-theoretic point of view, each cluster constitutes a complete sub-graph
  - This algorithm favors compact classes
- **Average and mean distance**
  - The minimum and maximum distance are extremely sensitive to outliers since their measurement of between-cluster distance involves minima or maxima
  - The average and mean distance approaches are more robust to outliers
  - Of the two, the mean distance is computationally more attractive
    - Notice that the average distance approach involves the computation of N_iN_j distances for each pair of clusters

Agglomerative clustering, minimum Vs. maximum distance

- Consider the problem of clustering nine major cities in the United States

	BOS	NY	DC	MIA	CHI	SEA	SF	LA	DEN
BOS	0	206	429	1504	963	2976	3095	2979	1949
NY	206	0	233	1308	802	2815	2934	2786	1771
DC	429	233	0	1075	671	2684	2799	2631	1616
MIA	1504	1308	1075	0	1329	3273	3053	2687	2037
CHI	963	802	671	1329	0	2013	2142	2054	996
SEA	2976	2815	2684	3273	2013	0	808	1131	1307
SF	3095	2934	2799	3053	2142	808	0	379	1235
LA	2979	2786	2631	2687	2054	1131	379	0	1059
DEN	1949	1771	1616	2037	996	1307	1235	1059	0