EE 583
PATTERN RECOGNITION

Bayes Decision Theory

Supervised Learning

Linear Discriminant Functions

Unsupervised Learning

- Parametric methods
- Nonparametric methods
  - Iterative approaches
  - Hierarchical approaches
  - Graph-theoretical approaches

Introduction

- Classification procedures that use unlabeled samples are called *unsupervised*.
- Reasons for working such an "unpromised" approach :
  - computational cost of labeling a huge sample (training) set
  - for systems whose class-specific pattern generation change in time, it is necessary to adopt continuously
  - training with huge unlabeled data first, labeling the clusters afterwards, is more preferable
  - some useful features can be found during unsupervised classification

- Two main approaches :
  - Parametric strategy : assume forms are known, combined classification and parameter estimation
  - Nonparametric strategy : Partitioning

Parametric Unsupervised Learning (1/2)

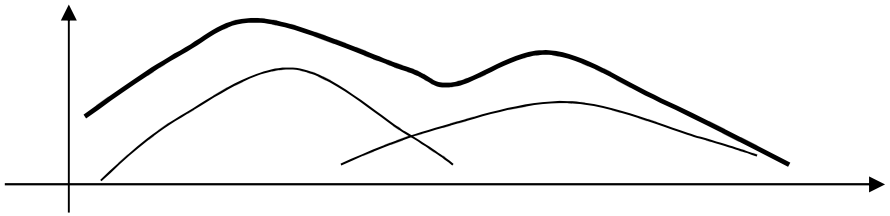
- There are initial assumptions :
  - Samples are from c number of classes
  - A priori class probabilities, $p(w_k)$, are known
  - Form of conditional prob, $p(x|w_k,\Theta_k)$, is known
  - Deterministic parameter vector is unknown
  - Class labels are unknown

- Samples are assumed to be obtained by
  - selecting a state of nature w_k with $p(w_k)$, then
  - selecting an x according to $p(x|w_k, \Theta_k)$

Parametric Unsupervised Learning (2/2)

- Mixture density can be written as :

$$p(x|\Theta) = \sum_{j=1}^{c} \underbrace{p(x|\Theta_j, w_j)}_{\text{component density}} p(w_j) \quad \textit{mixture density}$$



- Goal : First estimate unknown parameter, then decompose mixture into its components to make a classification
  - Note that different values of unknown parameter may lead to the same mixture density → "identifiable" densities are assumed (usually, a problem in discrete densities)

Parametric Unsupervised Learning
Maximum Likelihood (ML) Estimate (1/2)

- Let $X=\{x_1,\ldots,x_n\}$ be n unlabelled samples drawn independently from $p(x\mid\Theta)=\sum_{j=1}^{c} p(x\mid w_j,\Theta_j)\,p(w_j)$
- ML estimate : $\hat{\Theta}_{ML}=\arg\max_{\Theta} p(X\mid\Theta)=\arg\max_{\Theta}\prod_{k=1}^{n} p(x_k\mid\Theta)$
- Let $l\equiv\log p(X\mid\Theta)=\sum_{k=1}^{n}\log p(x_k\mid\Theta)$
- In order to maximize l

$$\nabla_{\Theta_i} l=\sum_{k=1}^{n}\nabla_{\Theta_i}\left(\log p(x_k\mid\Theta)\right)=\sum_{k=1}^{n}\frac{1}{p(x_k\mid\Theta)}\nabla_{\Theta_i}\left(\sum_{j=1}^{c} p(x_k\mid w_j,\Theta_j)\,p(w_j)\right)=0$$

Note that $\dfrac{\partial\log(u(x))}{\partial x}=\dfrac{1}{u(x)}\dfrac{\partial u(x)}{\partial x}$

Parametric Unsupervised Learning
Maximum Likelihood (ML) Estimate (2/2)

- Since parameters for different classes are functionally independent

$$\nabla_{\Theta_i} l = \sum_{k=1}^{n} \frac{1}{p(x_k \mid \Theta)} \nabla_{\Theta_i} \left(p(x_k \mid w_i, \Theta_i) p(w_i) \right)$$

$$= \sum_{j=1}^{c} \underbrace{\frac{p(w_i)}{p(x_k \mid \Theta)}}_{\frac{p(w_i \mid x_k, \Theta)}{p(x_k \mid w_i, \Theta_i)}} \nabla_{\Theta_i} \left(p(x_k \mid w_i, \Theta_i) \right)$$

- Considering the relation between derivative and log, we obtain the equation below to be solved for all i.

$$\nabla_{\Theta_i} l = \sum_{k=1}^{n} p(w_i \mid x_k, \Theta) \nabla_{\Theta_i} \left(\log p(x_k \mid w_i, \Theta_i) \right) = 0 \quad i = 1, \ldots, c$$

$$where \quad p(w_i \mid x_k, \Theta) = \frac{p(x_k \mid w_i, \Theta_i) p(w_i)}{\sum_{j=1}^{c} p(x_k \mid w_j, \Theta_j) p(w_j)}$$

- Since it is nonlinear, solution can be obtained iteratively.

- If $P(w_j)$'s are also unknown, ML estimate can be obtained by using a similar formulation

Parametric Unsupervised Learning
ML Estimate for Multivariate Normal (1/2)

- Assuming <u>only mean</u> is unknown, ML estimate :

$$\log p(x \mid w_i, \mu_i) = -\log\left((2\pi)^{d/2} \mid \Sigma_i^{-1} \mid\right) - \frac{1}{2}(x-\mu_i)^t \Sigma_i^{-1}(x-\mu_i)$$

$$\Rightarrow \quad \nabla_\mu \log p(x \mid w_i, \mu_i) = \Sigma_i^{-1}(x-\mu_i)$$

$$\Rightarrow \quad \sum_{k=1}^{n} p(w_i \mid x_k, \hat{\mu}) \Sigma_i^{-1}(x_k - \hat{\mu}_i) = 0 \qquad \textit{where } \hat{\mu} \textit{ is ML estimate}$$

$$\Rightarrow \quad \hat{\mu}_i = \frac{\overbrace{\sum_{k=1}^{n} p(w_i \mid x_k, \hat{\mu}) x_k}^{\textit{weighted average of } x_k}}{\sum_{k=1}^{n} p(w_i \mid x_k, \hat{\mu})} \qquad \textit{where } p(w_i \mid x_k, \hat{\mu}) = \frac{p(x_k \mid w_i, \mu_i)\, p(w_i)}{\sum_{j=1}^{c} p(x_k \mid w_j, \mu_j)\, p(w_j)}$$

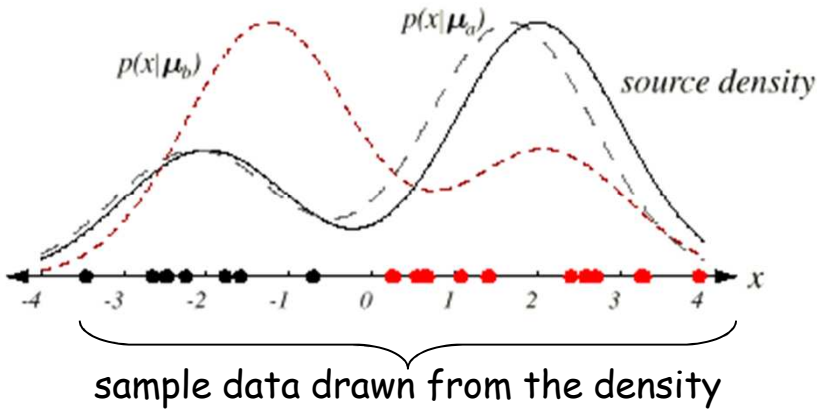
- No explicit solution; mean can be obtained iteratively with no guarantee for global minimum

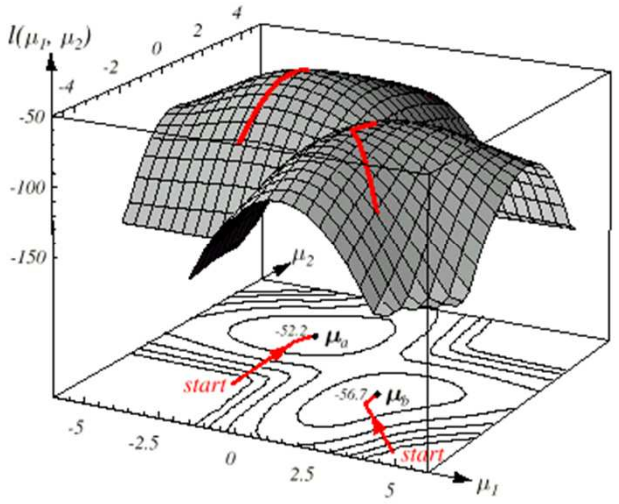
$$\hat{\mu}_i(j+1) = \sum_{k=1}^{n} p(w_i \mid x_k, \hat{\mu}(j)) x_k \Big/ \sum_{k=1}^{n} p(w_i \mid x_k, \hat{\mu}(j))$$

Parametric Unsupervised Learning
ML Estimate for Multivariate Normal (2/2)

- Consider the following example : $\mu^a = (\mu_1, \mu_2) = (-2, 2)$:

$$p(x \mid \mu_1, \mu_2) = \frac{1}{3\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\mu_1)^2\right] + \frac{2}{3\sqrt{2\pi}} \exp\left[-\frac{1}{2}(x-\mu_2)^2\right]$$





- Note that starting from two different starting points, the iterative algorithm reaches two different solutions = μ^a & μ^b

→ unidentifiable

Parametric Unsupervised Learning
K-means algorithm (1/2)

- If we find the nearest mean μ_m to x_k ➔ approximate as

$$p(w_i \mid x_k, \Theta) \approx \begin{cases} 1 & \text{if i = m} \\ 0 & \text{otherw.} \end{cases}$$

- Then, the iterative ML estimation becomes K-means

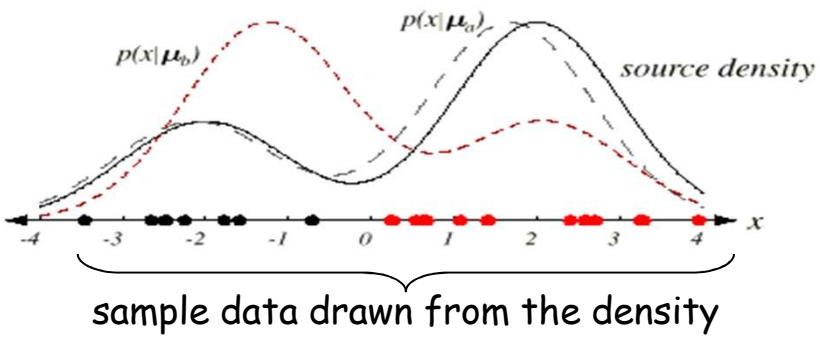
$$\hat{\mu}_i(j+1) = \sum_{k=1}^{n} p(w_i \mid x_k, \hat{\mu}(j)) x_k \Big/ \sum_{k=1}^{n} p(w_i \mid x_k, \hat{\mu}(j)) \Rightarrow \hat{\mu}_i(j+1) = \sum_{x_k \in X_{\mu_i(j)}} x_k / n_{|X_i|}$$

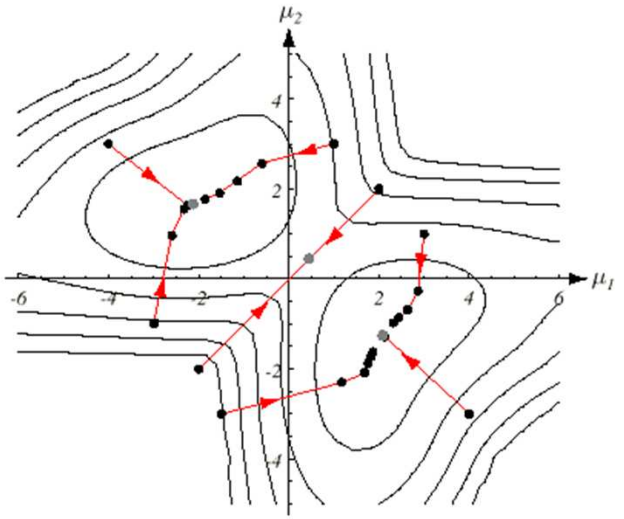
- <u>K-means or ISODATA or c-means algorithm</u> :

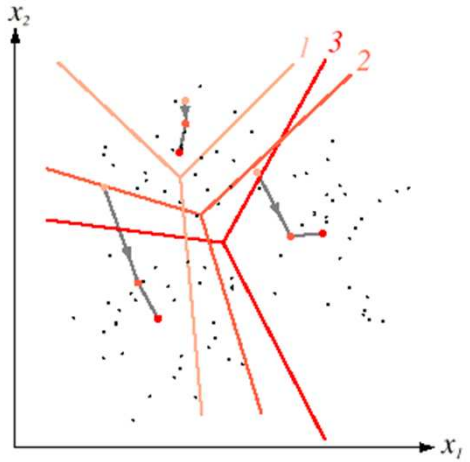
1) Choose initial mean values for k (or c) classes
2) Classify n samples by assigning them to "closest" mean
3) Recompute the means as the average of samples in their (new) classes
4) Continue till there is no change in mean values

Parametric Unsupervised Learning
K-means algorithm (2/2)

- The trajectory of the mean values during iterations for different initial values







- For 2-D unsupervised data, Voronoi regions change during iterations

Nonparametric Unsupervised Learning

- For non-trivial distributions parametric approaches usually fail
- An obvious alternative approach is using a priori information to design a classifier, then classify unlabelled data using this initial classifier (decision directed strategy)

- Some problems related such a strategy :
  - Initial classification is critical
  - An unfortunate sequence of samples will create more errors compared to estimating likelihood
  - Even if the initial classification is optimal, samples from the tails of other overlapping distributions will create biased estimates (i.e. less probable samples from one class will be included in the other class)

Nonparametric Unsupervised Learning

- Clustering : A procedure yields a data description in terms of groups of data points that posses strong internal "similarities" (criterion fnct.)

- Partitioned samples in one cluster should be more similar to samples in the same cluster (wrt to the samples in other clusters), but
  - how to measure similarity between samples?
  - how one should evaluate a partitioning?

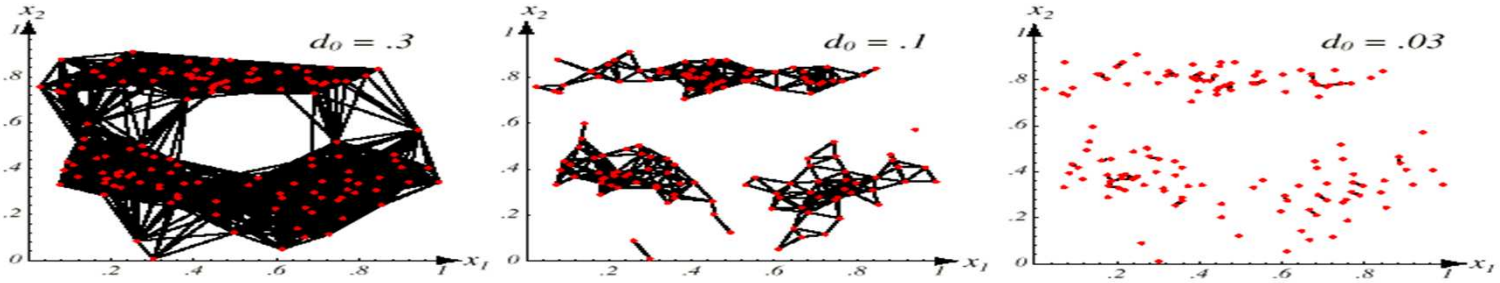
Nonparametric Unsupervised Learning Similarity Measures (1/2)

- A good similarity measure is some 'distance' (e.g. Euclidian) between two samples
- It is expected to have shorter distance between the samples in one cluster, compared to the samples in different clusters
- Using a threshold, clusters can be classified, as "similar" or "dissimilar"
- In order to be scale independent, either
  - use normalization (can be problematic)
  - use non metric similarity functions, such as normalized inner product :

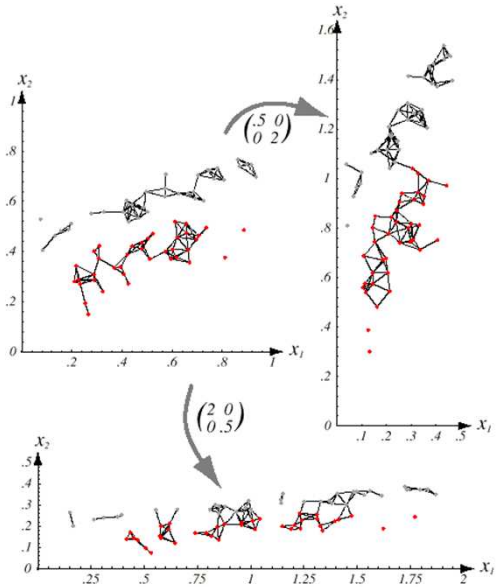
$$s(\vec{x}_1, \vec{x}_2) = \frac{\vec{x}_1^t \vec{x}_2}{\|\vec{x}_1\| \|\vec{x}_2\|}$$

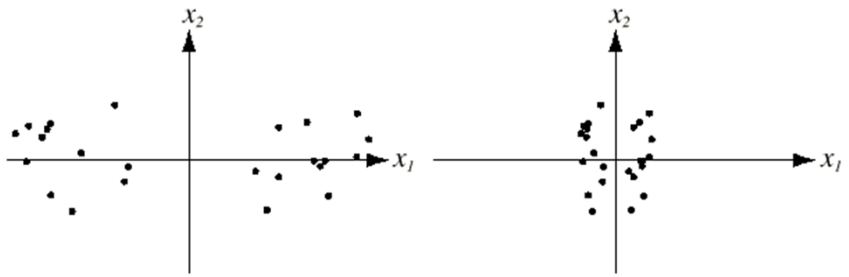
Nonparametric Unsupervised Learning Similarity Measures (2/2)

- Threshold selection is critical



- Normalization can be problematic





Nonparametric Unsupervised Learning
Criterion Function for Clustering

- Criterion function measures the clustering quality of any partition of data

1) Sum-of-squared-error criterion (minimum variance) :

$$J_e = \sum_{i=1}^{c} \sum_{x \in X_i} \left\| x - m_i \right\|^2 \quad where\ m_i = \frac{1}{n_i} \sum_{x \in X_i} x$$

2) Related minimum variance criteria :

$$J_e = \frac{1}{2} \sum_{i=1}^{c} n_i \bar{s}_i \quad where \quad \bar{s}_i = \frac{1}{n_i^2} \sum_{x' \in X_i} \sum_{x \in X_i} \left\| x - x' \right\|^2 \quad or$$

$$\text{More generally, } \bar{s}_i = \frac{1}{n_i^2} \sum_{x' \in X_i} \sum_{x \in X_i} s(x, x') \ : \text{average}$$

$$or \quad \bar{s}_i = \min_{x, x' \in X_i} s(x, x') \quad or \quad \bar{s}_i = \underset{x, x' \in X_i}{median}\ s(x, x')$$

Nonparametric Unsupervised Learning Iterative Optimization (1/4)

- Clustering becomes a well-defined problem as as soon as criterion function is selected
- One option for a solution is exhaustive search

  e.g. 5 clusters, 100 samples --> 10^{67} partitioning !
- Iterative optimization is another option

  Begin from a reasonable initial partition, then "move" samples from one group to another if that move is feasible
- Iterative optimization is suboptimal since it depends on initial partitioning

Nonparametric Unsupervised Learning Iterative Optimization (2/4)

- Assume sum-of-squared-error criterion

$$J_e = \sum_{i=1}^{c} J_i \quad J_i = \sum_{x \in X_i} \|x - m_i\|^2 \quad m_i = \frac{1}{n_i} \sum_{x \in X_i} x$$

- Let x move from cluster X_i to X_j

$$m_j^* = m_j + \frac{\hat{x} - m_j}{n_j + 1}, \quad J_j^* = J_j + \frac{n_j}{n_j + 1} \|\hat{x} - m_j\|^2$$

$$m_i^* = m_i - \frac{\hat{x} - m_i}{n_i - 1} \quad J_i^* = J_i - \frac{n_i}{n_i - 1} \|\hat{x} - m_i\|^2$$

- Since total criterion, J_e, is tried to be minimized, transfer from X_i to X_j is accepted if decrease in J_i is greater than increase in J_j

$\frac{n_j}{n_j + 1} \|\hat{x} - m_j\|^2 < \frac{n_i}{n_i - 1} \|\hat{x} - m_i\|^2$ (usually happens if x is closer to m_j than m_i)

Nonparametric Unsupervised Learning
Iterative Optimization (3/4)

- Minimum squared algorithm :

  1) Select an initial partition for n samples; compute Je and mean

  2) Select next candidate, x, (let x belong to cluster i)

  3) Compute

$$a_j = \begin{cases} \dfrac{n_j}{n_j+1}\left\|\hat{x}-m_j\right\|^2 & j \neq i \\ \dfrac{n_i}{n_i-1}\left\|\hat{x}-m_i\right\|^2 & j = i \end{cases}$$

  4) Transfer x to X_k if $a_k < a_j$ for all j

  5) Update J_e, m_i and m_k

  6) If J_e does not change after n attempts, STOP

Note that MSE is the sequential version of K-means

Nonparametric Unsupervised Learning Iterative Optimization (4/4)

- Experimentally, it is shown that
  - MSE is more susceptible to being trapped by a local minima compared to ISODATA
  - MSE depends on the order of choosing samples
  - MSE is step-wise optimal (if the problem is sequential, it is advantageous)
- A common problem to iterative optimization methods is selection of starting points
- A solution is to use (c-1)-cluster problem as an initial estimate c-cluster problem. Beginning from 1-cluster, use the sample furthest from all samples to obtain the next cluster mean →basis for *hierarchical clustering*

Nonparametric Unsupervised Learning Hierarchical Clustering (1/4)

- Two main strategies for hierarchical clustering
  - Agglomerative (bottom-up) :

    Start with n singleton clusters and reach to c clusters by merging "similar" clusters by decreasing cluster number
  - Divisive (top-down) :

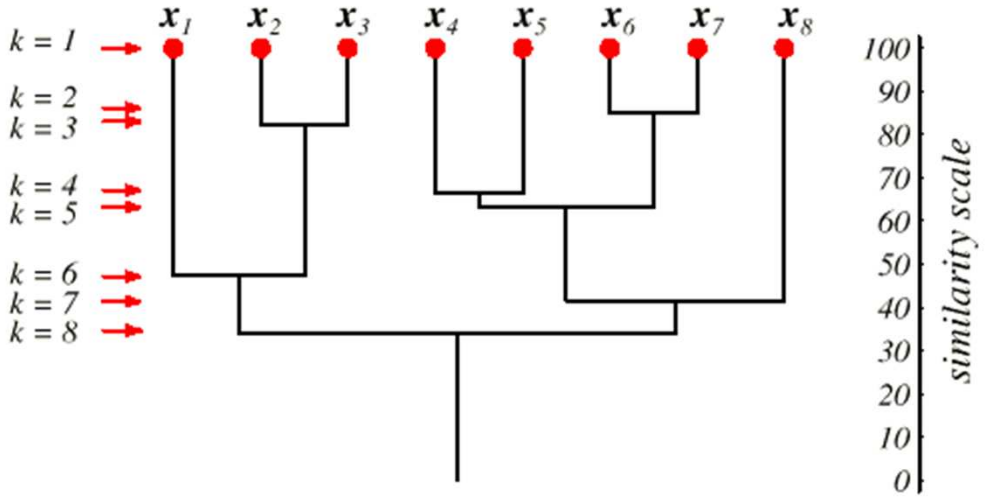
    Start with 1 cluster which contains n samples and reach to c clusters successively split clusters

- For n (large number) samples to be classified into c (large number) classes, agglomerative approaches are better from computational point of view (or vice versa) : n → c ← 1

Nonparametric Unsupervised Learning
Hierarchical Clustering (2/4)

- Agglomerative Hierarchical Clustering :

1) Let c'=n, $X_i=\{x_i\}$ i=1,…,n
2) If c'<c STOP (c=# of regions)
3) Find "nearest" pair of distinct clusters, X_i & X_j
4) Merge X_i & X_j ; delete X_j; decrement c'; GOTO 2



Similarity scale can be used to determine whether groupings are natural or forced

Nonparametric Unsupervised Learning
Hierarchical Clustering (3/4)

- In this hierarchy, at any level, distance between clusters gives a similarity value for finding the nearest clusters

- Following distance measures can be utilized

$$d_{\min}(X_i, X_j) = \min_{x \in X_i; x' \in X_j} \|x - x'\|$$ Nearest neighbour measure

$$d_{\max}(X_i, X_j) = \max_{x \in X_i; x' \in X_j} \|x - x'\|$$ Furthest neighbour measure

$$d_{avg}(X_i, X_j) = \sum_{x \in X_i} \sum_{x' \in X_j} \|x - x'\|$$

$$d_{mean}(X_i, X_j) = \|m_i - m_j\|$$

Compromise between two

Nonparametric Unsupervised Learning
Hierarchical Clustering (4/4)

- **Stepwise Optimal Hierarchical Clustering** :
  - Change one step of "Agglomerative Clustering" algorithm so that at every step a stepwise optimal procedure (wrt a criterion function) is extremized (Ward's Algorithm):

    Find pair of distinct clusters Xi & Xj whose merger would increase (decrease) the criterion function, minimum

$$J_e = \sum_{r=1}^{c} \sum_{x \in C_r} \left\| x - m_r \right\|^2 \rightarrow \nabla J_e = \sum_{x \in C_{ij}} \left\| x - m_{ij} \right\|^2 - \sum_{x \in C_i} \left\| x - m_i \right\|^2 - \sum_{x \in C_j} \left\| x - m_j \right\|^2$$

  - If we choose distance function as below, then the increase in sum-of squared error criterion is minimized at every step

$$d_e(X_i, X_j) = \nabla J_e = \sqrt{\frac{n_j n_i}{n_j + n_i}} \left\| m_i - m_j \right\|$$

    In practice, d_e usually favors merging small regions to large regions rather than merging two mid-size region

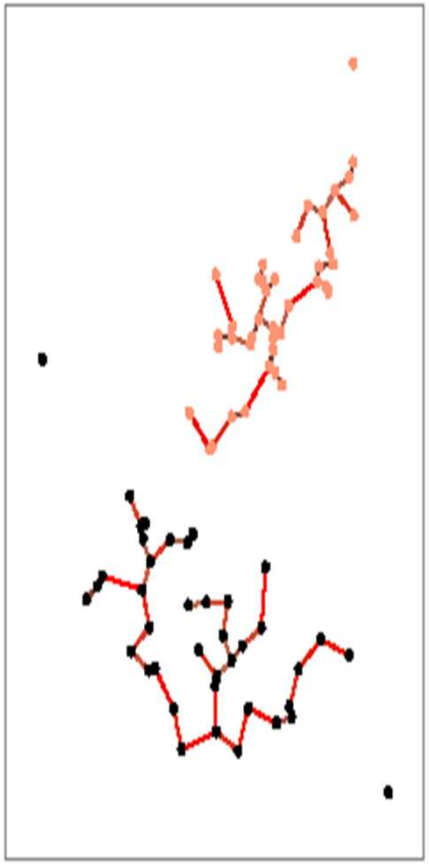
Nonparametric Unsupervised Learning Graph Theoretic Approaches (1/3)

- Graph Theory can be applied to nonparametric unsupervised learning
- The samples to be clustered can be assumed to be *nodes* of a graph
- If two nodes are found out to be similar (belonging to the same cluster) ➔ an edge is placed between these nodes
- All nodes, which are connected to each via chain of edges belong to the same cluster
- At any state, how to merge two clusters in such a graph?

Nonparametric Unsupervised Learning
Graph Theoretic Approaches (2/3)

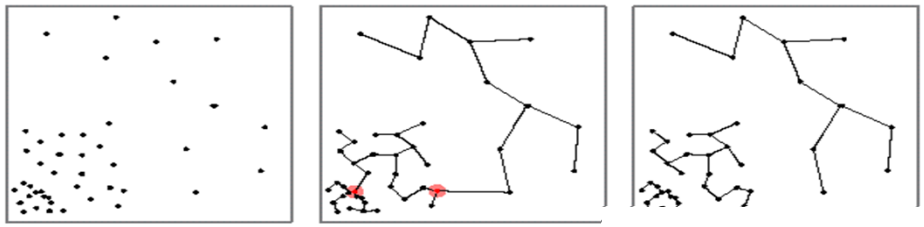
- Assume merging two clusters corresponds to adding an *edge* between these two nodes
- Since edges linking for clusters never make loops, in graph theory, such edge groups are called *trees*
- If continue edge linking till all clusters are merged, then a *spanning tree* is obtained which reaches all nodes
- If $d_{min}(Xi,Xj)$ measure is used during merger

  ➔ resulting graph is a *minimal spanning tree*,
  - i.e nearest neighbor clustering algorithm can be viewed as an algorithm to obtain minimal spanning tree

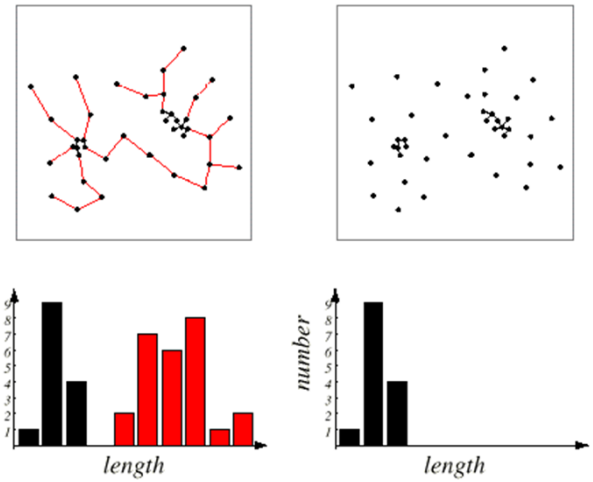


Nonparametric Unsupervised Learning
Graph Theoretic Approaches (3/3)

- Conversely, given a minimal spanning tree, clusters obtained by the nearest neighbor algorithm can be found by first dividing tree into two by removing the longest edge, then …
- Instead of removing the longest edge, one option is to remove the most 'inconsistent' edge incident to the same node

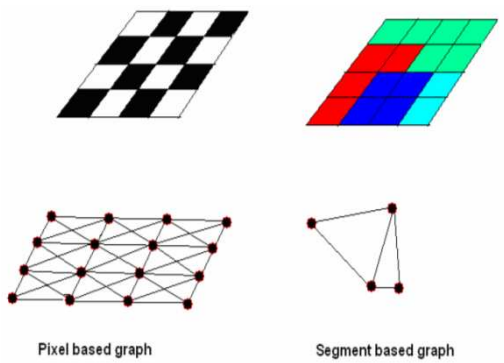


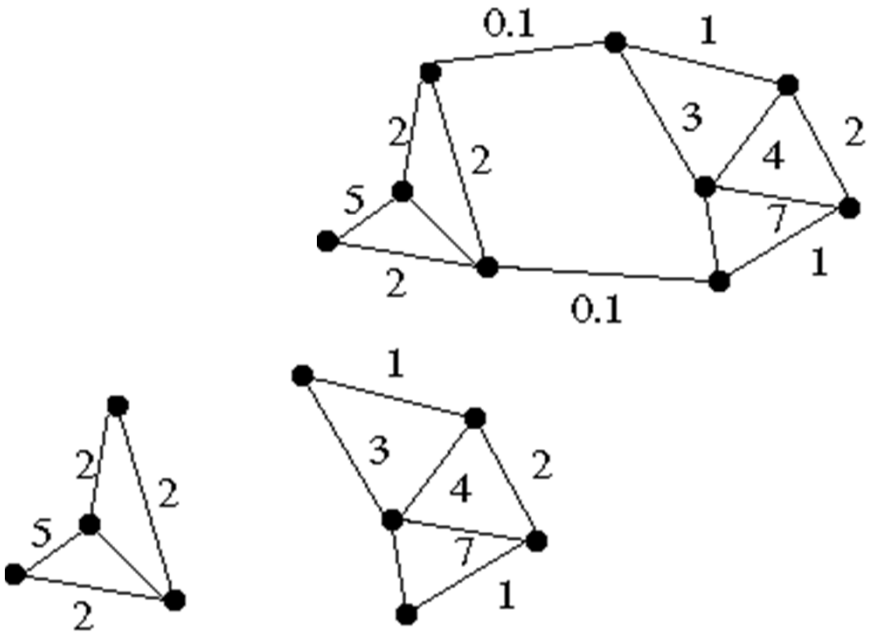
- Another option is to determine the edge length distribution and segment different edge length groups accordingly

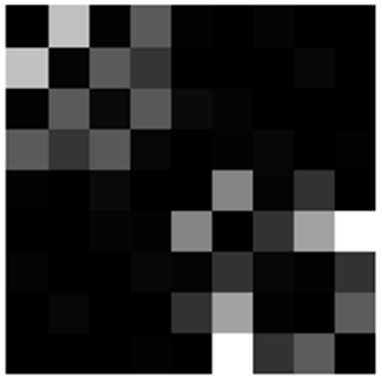


Graph theoretic Clustering

- A top-down approach
- Tokens (nodes) are represented by using a weighted graph.
  - affinity matrix, *A* (similar nodes have higher entries)
- Cut up this graph to get subgraphs with strong links



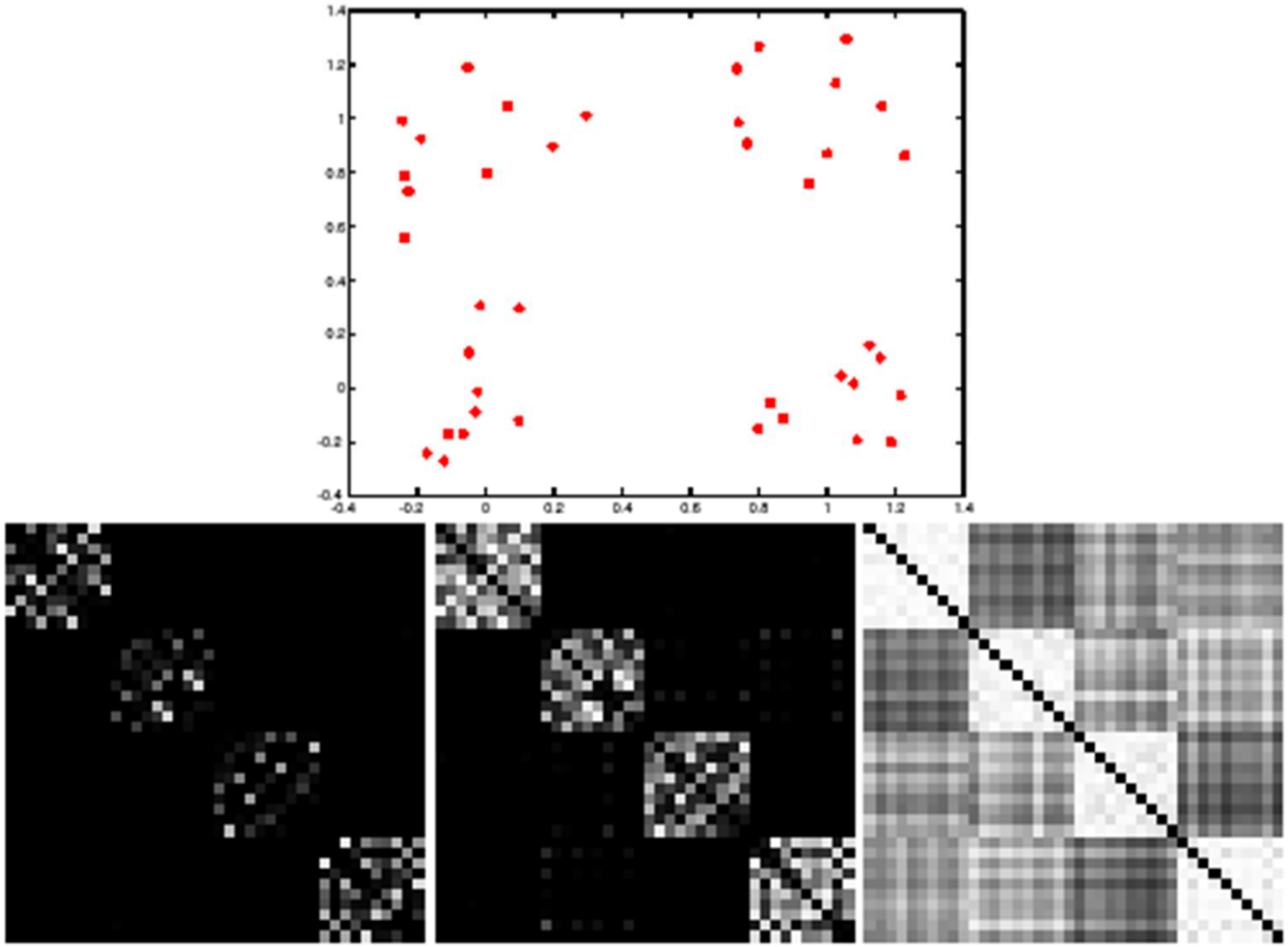




Affinity matrix

Affinity Matrix with scale (σ)

$$aff(x,y) = \exp\left\{-\left(1/2\sigma_d^2\right)\left(\|x-y\|^2\right)\right\}$$



Solution via Eigenvectors

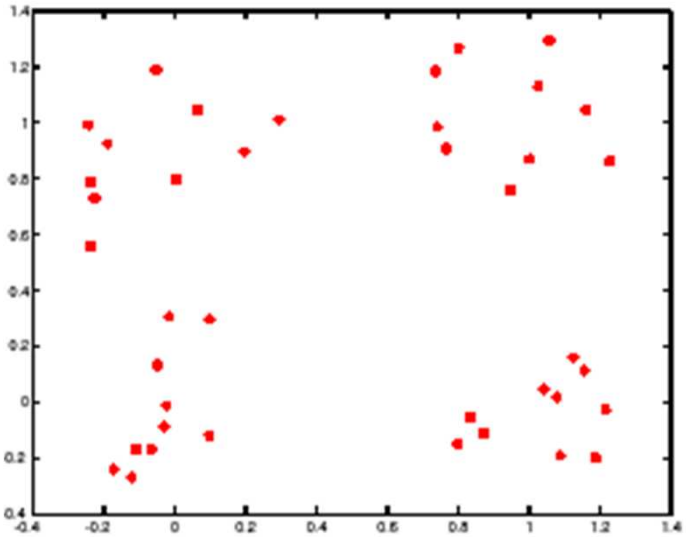
- Idea: Find vector, w, giving the association between each node and a cluster
  - Elements within a cluster should have strong affinity with each other
  - Maximize the following relation: $w^T A\, w$ (A : *affinity matrix*)

$$w^T A\, w = \sum_{i,j} w_i a_{i,j} w_j$$

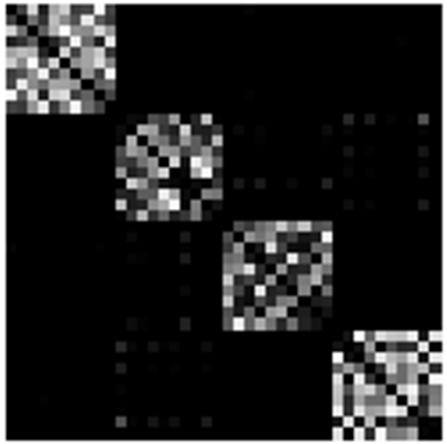
$$= \sum \Bigg(\underbrace{\text{assoc. of node}_i \text{ to cluster}}_{w_i} \Bigg) \Bigg(\underbrace{\text{similarity node}_i \text{ \& node}_j}_{a_{i,j}} \Bigg) \Bigg(\underbrace{\text{assoc. of node}_j \text{ to cluster}}_{w_j} \Bigg)$$

- Above relation maximizes, in case all 3 terms are non-zero (or not very small)
- There should be an extra constraint, as $w^T w = 1$
- Optimize by method of *Lagrange* multiplier : $\max \{ w^T A w + \lambda (w^T w - 1) \}$
- Solution is an eigenvalue problem
- Choose the eigenvector of A with the largest eigenvalue

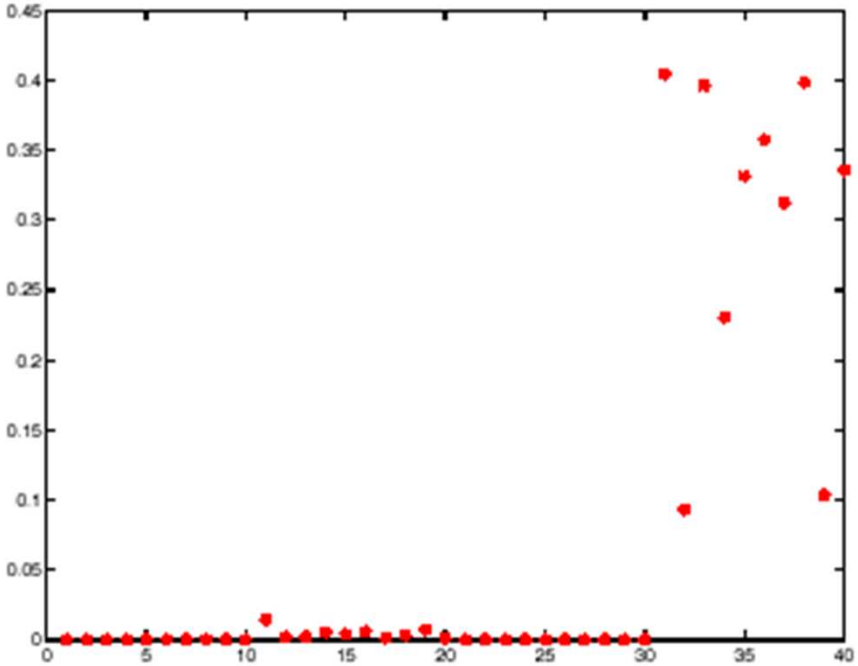
Solution via Eigenvectors



points



matrix

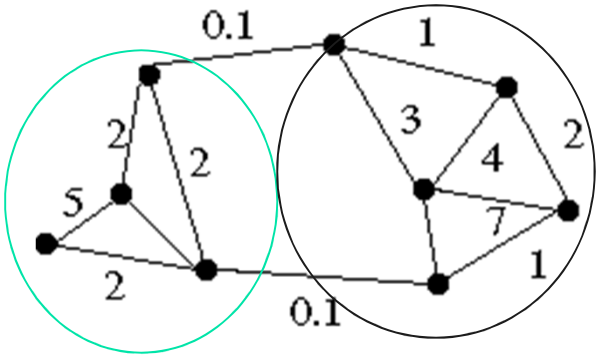


eigenvector

Slides modified from D.A. Forsyth, Computer Vision - A Modern Approach

Normalized cuts

- Previous criterion only evaluates within cluster similarity, but not across cluster difference
- Instead, one would like to maximize within cluster similarity compared to the across cluster difference
- Write graph V, one cluster as A and the other as B



- Minimize *Normalized Cut*

$$\left(\frac{cut(A,B)}{assoc(A,V)}\right)+\left(\frac{cut(A,B)}{assoc(B,V)}\right)$$

$cut(A,B)$: sum of edges between $A\&B$

$assoc(A,V)$: sum of edges only in A

- Construct A, B such that their within cluster similarity is high,
  - compared to their association with the rest of the graph

Slides modified from D.A. Forsyth, Computer Vision - A Modern Approach

Normalized cuts

$$NCut(A,B) = \left(\frac{cut(A,B)}{assoc(A,V)}\right) + \left(\frac{cut(A,B)}{assoc(B,V)}\right)$$

$cut(A,B)$: sum of edges between A&B
$assoc(A,V)$: sum of edges only in A

$$= \frac{(\bar{1}+x)^T (D-W)(\bar{1}+x)}{k\,\bar{1}^T D\bar{1}} + \frac{(\bar{1}-x)^T (D-W)(\bar{1}-x)}{k\,\bar{1}^T D\bar{1}}$$

x : vector of enteries ± 1, $x(i)=1 \Leftrightarrow i \in A$; $x(i)=-1 \Leftrightarrow i \in B$

W : affinity matrix; $D(i,i) = \sum_j W(i,j)$; $k = \frac{\sum_{x_i>0} D(i,i)}{\sum_i D(i,i)}$

$Let\ y \equiv (\bar{1}+x) - b(\bar{1}-x), \quad$ where $b = k/(1-k)$

Normalized cuts

- Defined vector y, has elements as
  - 1, if item is in A,
  - $-b$, if item is in B
- After derivations $\min NCut(A,B) = \min\left(\frac{cut(A,B)}{assoc(A,V)}\right) + \left(\frac{cut(A,B)}{assoc(B,V)}\right)$

  is shown to be equivalent to $\min_y\left(\frac{y^T(D-W)y}{y^T Dy}\right)$
- with the constraint $y^T D1 = 0$

  (Read proof in the distributed notes)
- This is so called *Rayleigh Quotient*

Normalized cuts

- Its solutions is the generalized eigenvalue problem

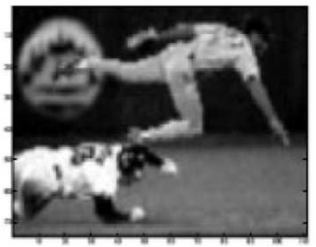
$$\min \left(y^T (D-W) y\right) \text{ subject to } \left(y^T D y = 1\right)$$

which gives

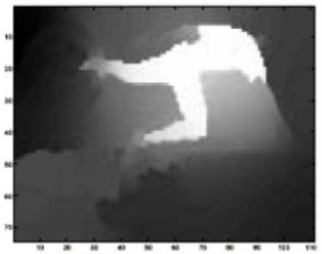
$$(D-W)y = \lambda D y$$

$$\Rightarrow D^{-1/2}(D-W)D^{-1/2}y = \lambda y$$

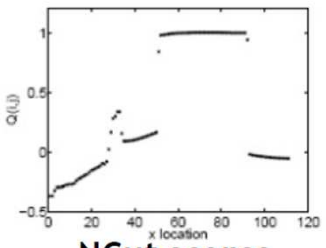
- Optimal solution is the eigenvector due to second smallest eigenvalue
- Now, look for a quantization threshold that maximizes the criterion --- i.e. all components of y above that threshold go to one, all below go to $-b$



Image



Eigenvector



NCut scores

Slides modified from D.A. Forsyth, Computer Vision - A Modern Approach