CS 536: Machine Learning

Nonparametric Density Estimation
Unsupervised Learning - Clustering

Fall 2005

Ahmed Elgammal
Dept of Computer Science
Rutgers University

Outlines

- Density estimation
- Nonparametric kernel density estimation
- Mixture Densities
- Unsupervised Learning - Clustering:
  - Hierarchical Clustering
  - K-means Clustering
  - Mean Shift Clustering
  - Spectral Clustering – Graph Cuts
  - Application to Image Segmentation

Density Estimation

- Parametric: Assume a single model for $p\ (\boldsymbol{x} \mid C_i)$ (Chapter 4 and 5)
- Semiparametric: $p\ (\boldsymbol{x} \mid C_i)$ is a mixture of densities

  Multiple possible explanations/prototypes:

  Different handwriting styles, accents in speech
- Nonparametric: No model; data speaks for itself (Chapter 8)

Nonparametric Density Estimation

Density Estimation: Given a sample $S=\{x_i\}_{i=1..N}$ from a distribution obtain an estimate of the density function $\hat{f}(\cdot)$ at any point.

<u>Parametric</u> : Assume a parametric density family $f\,(.\,|\theta)$, (ex. $N(\mu,\sigma^2)$) and obtain the best estimator $\hat{\theta}$ of θ

Advantages:

- Efficient
- Robust to noise: robust estimators can be used

Problem with parametric methods

- An incorrectly specified parametric model has a bias that cannot be removed even by large number of samples.

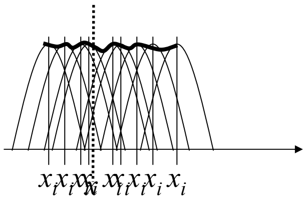
<u>Nonparametric</u> : directly obtain a good estimate $\hat{f}(\cdot)$ of the entire density $f(\cdot)$ from the sample.

Most famous example: Histogram

Kernel Density Estimation

- 1950s + (Fix & Hodges 51, Rosenblatt 56, Parzen 62, Cencov 62)
- Given a set of samples $S=\{x_i\}_{i=1..N}$ we can obtain an estimate for the density at x as:

$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} K(\frac{x-x_i}{h}) = \frac{1}{N}\sum_{i=1}^{N} K_h(x-x_i)$$



$$\hat{f}(x) = \frac{1}{Nh}\sum_{i=1}^{N} K(\frac{x-x_i}{h}) = \frac{1}{N}\sum_{i=1}^{N} K_h(x-x_i)$$

where $K_h(t)=K(t/h)/h$ called kernel function (window function)

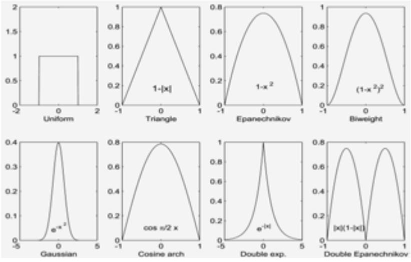
h : scale or bandwidth

K satisfies certain conditions, e.g.:

$$\int K_h(x)dx = 1$$

$$K_h(x) \geq 0$$

Kernel Estimation



- A variety of kernel shapes with different properties.
- Gaussian kernel is typically used for its continuity and differentiability.

- Multivariate case: Kernel Product

  Use same kernel function with different bandwidth *h* for each dimension.
- General form: avoid to store all the samples

$$\widehat{f}(x) = \frac{1}{N}\sum_{i=1}^{N}\prod_{j=1}^{d} K_{h_j}(x^j - x_i^{\,j})$$

$$\widehat{f}(x) = \sum_{i=1}^{N} \alpha_i K_h(x - x_i)$$

Kernel Density Estimation

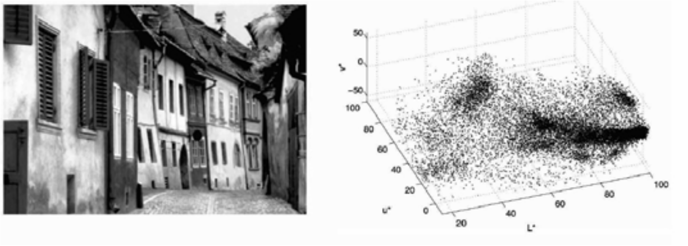
Advantages:

- Converge to any density shape with sufficient samples.

  asymptotically the estimate converges to any density.
- No need for model specification.
- Unlike histograms, density estimates are smooth, continuous and differentiable.
- Easily generalize to higher dimensions.
- All other parametric/nonparametric density estimation methods, e.g., histograms, are asymptotically kernel methods.
- In many applications, the densities are multivariate and multimodal with irregular cluster shapes.

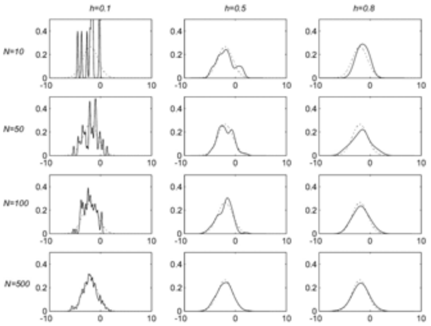
Example: color clusters

- Cluster shapes are irregular
- Cluster boundaries are not well defined.

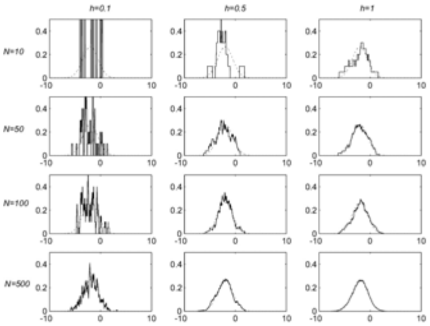


□✕ ⫽◡□□□□□ ⫽◡✓ ● •⎾ • ▾ ✓ ●_┘□□□□□⎾ ⎾ ✕□□□□⎾ ✓ ●□? • • _┘▴□□□□✕ ⫽✓ ▾ ? ▴□
✓ ⫽⫽✕ ⫽✓⎾ •□▴ ⫽◂ ✓ ✕_┘□_┘⎾ ✓ ▴ ▾ ✕⎾ □? ⫽✓⎾ ⎾ □✓ ● ✓ ◠⑊ ? • ?□□

Conversion - KDE

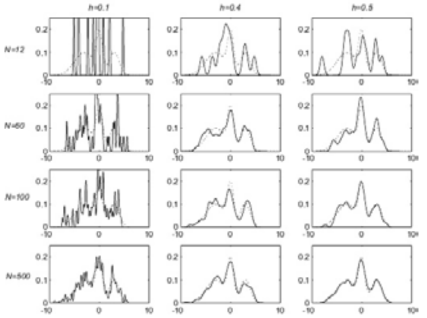


Estimation using Gaussian Kernel

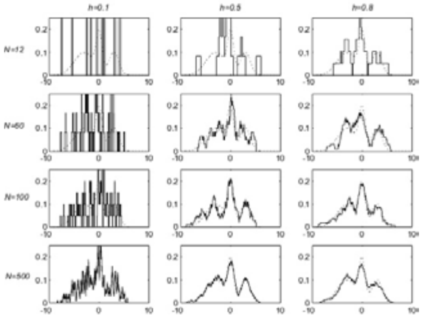


Estimation using Uniform Kernel

Conversion - KDE



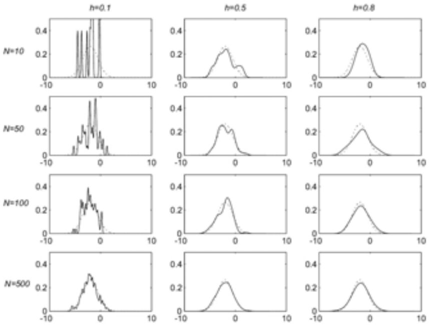
Estimation using Gaussian Kernel



Estimation using Uniform Kernel

Scale selection

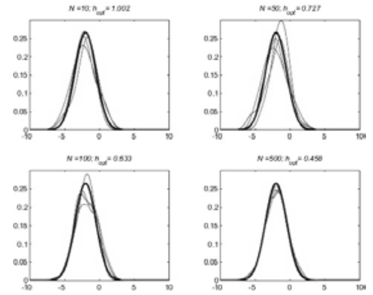
- Important problem. Large literature.
- Small *h* results in ragged densities.
- Large *h* results in over smoothing.
- Best choice for h depends on the number of samples:
  - small *n*, wide kernels
  - large *n*, Narrow kernels
  - $\lim_{n \to \infty} h(n) = 0$



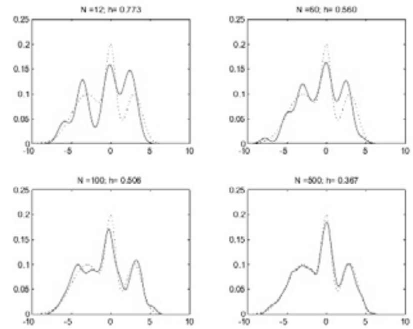
Optimal scale

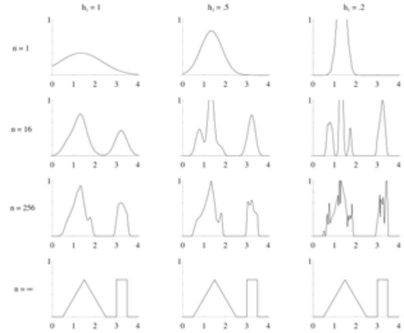
- Optimal kernel and optimal scale can be achieved by minimizing the mean integrated square error – if we know the density !
- Normal reference rule:

$$h^{opt} = (4/3)^{1/5} \sigma \cdot n^{-1/5} \approx 1.06\hat{\sigma} \cdot n^{-1/5}$$



Scale selection





From R. O. Duda, P. E. Hart, and D. G. Stork. “*Pattern Classification*” Wiley, New York, 2nd edition, 2000

Density Estimation

- Parametric: Assume a single model for $p\ (\boldsymbol{x} \mid \mathrm{C}_i)$ (Chapter 4 and 5)
- Semiparametric: $p\ (\boldsymbol{x} \mid \mathrm{C}_i)$ is a mixture of densities
  Multiple possible explanations/prototypes:
  Different handwriting styles, accents in speech
- Nonparametric: No model; data speaks for itself (Chapter 8)

Mixture Densities

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} p(\boldsymbol{x} \mid \mathrm{G}_i) P(\mathrm{G}_i)$$

where G_i the components/groups/clusters,

$P(\mathrm{G}_i)$ mixture proportions (priors),

$p(\boldsymbol{x} \mid \mathrm{G}_i)$ component densities

Gaussian mixture where $p(\boldsymbol{x}|\mathrm{G}_i) \sim \mathrm{N}(\boldsymbol{\mu}_i, \sum_i)$

parameters $\Phi = \{P(\mathrm{G}_i), \boldsymbol{\mu}_i, \sum_i\}_{i=1}^{k}$

unlabeled sample $\mathrm{X}=\{\boldsymbol{x}^t\}_t$ (unsupervised learning)

Classes vs. Clusters

- Supervised: $\mathrm{X} = \{\boldsymbol{x}^t, \boldsymbol{r}^t\}_t$
- Classes C_i i=1,...,K

$$p(\boldsymbol{x}) = \sum_{i=1}^{K} p(\boldsymbol{x} \mid \mathrm{C}_i) P(\mathrm{C}_i)$$

where $p(\boldsymbol{x} \mid \mathrm{C}_i) \sim \mathrm{N}(\boldsymbol{\mu}_i, \sum_i)$

- $\Phi = \{P(\mathrm{C}_i), \boldsymbol{\mu}_i, \sum_i\}_{i=1}^{K}$

$$\hat{P}(C_i) = \frac{\sum_t r_i^t}{N} \quad \boldsymbol{m}_i = \frac{\sum_t r_i^t \boldsymbol{x}^t}{\sum_t r_i^t}$$

$$\mathbf{S}_i = \frac{\sum_t r_i^t (\boldsymbol{x}^t - \boldsymbol{m}_i)(\boldsymbol{x}^t - \boldsymbol{m}_i)^T}{\sum_t r_i^t}$$

- Unsupervised : $\mathrm{X} = \{\boldsymbol{x}^t\}_t$
- Clusters G_i i=1,...,k

$$p(\boldsymbol{x}) = \sum_{i=1}^{k} p(\boldsymbol{x} \mid \mathrm{G}_i) P(\mathrm{G}_i)$$

where $p(\boldsymbol{x} \mid \mathrm{G}_i) \sim \mathrm{N}(\boldsymbol{\mu}_i, \sum_i)$

- $\Phi = \{P(\mathrm{G}_i), \boldsymbol{\mu}_i, \sum_i\}_{i=1}^{k}$

Labels, $\boldsymbol{r}_i^t$?

k-Means Clustering

- Find *k* reference vectors (prototypes/codebook vectors/codewords) which best represent data
- Reference vectors, $\boldsymbol{m}_j, j = 1,...,k$
- Use nearest (most similar) reference:

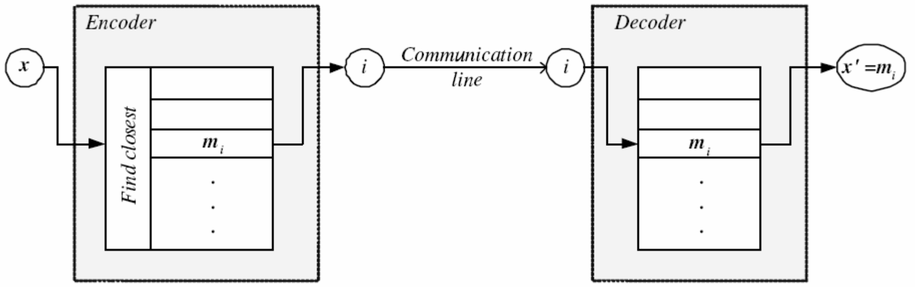
$$\left\|\boldsymbol{x}^t - \boldsymbol{m}_i\right\| = \min_j \left\|\boldsymbol{x}^t - \boldsymbol{m}_j\right\|$$

- Reconstruction error

$$E\left(\{\boldsymbol{m}_i\}_{i=1}^k \middle| \mathcal{X}\right) = \sum_t \sum_i b_i^t \left\|\boldsymbol{x}^t - \boldsymbol{m}_i\right\|$$

$$b_i^t = \begin{cases} 1 & \text{if } \left\|\boldsymbol{x}^t - \boldsymbol{m}_i\right\| = \min_j \left\|\boldsymbol{x}^t - \boldsymbol{m}_j\right\| \\ 0 & \text{otherwise} \end{cases}$$

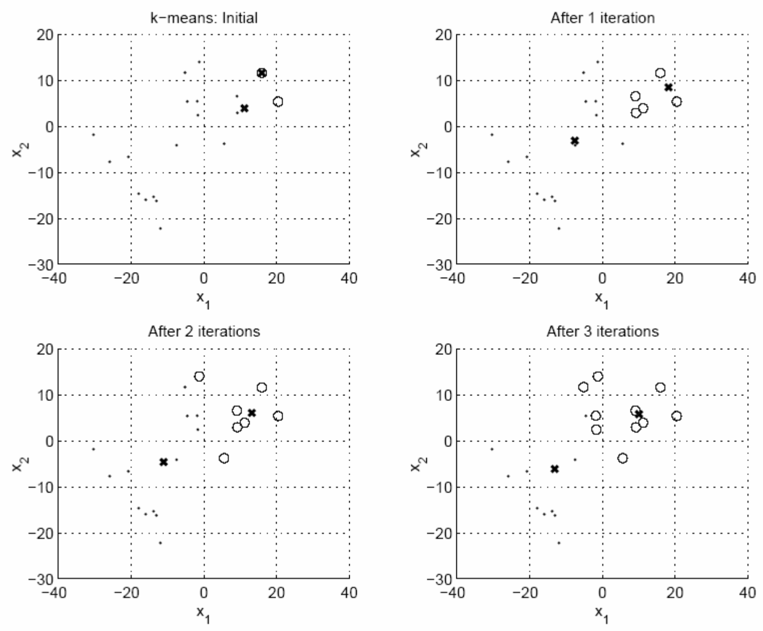
Encoding/Decoding

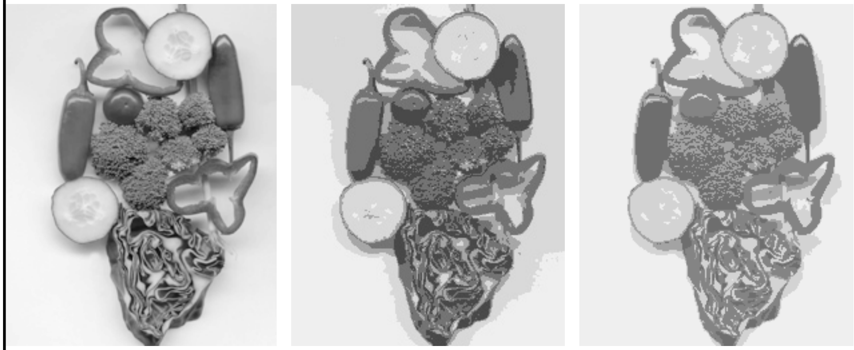


$$b_i^t = \begin{cases} 1 & \text{if } \left\|\boldsymbol{x}^t - \boldsymbol{m}_i\right\| = \min_j \left\|\boldsymbol{x}^t - \boldsymbol{m}_j\right\| \\ 0 & \text{otherwise} \end{cases}$$

k-means Clustering

```
Initialize $\boldsymbol{m}_i, i = 1, \ldots, k$, for example, to $k$ random $\boldsymbol{x}^t$
Repeat
    For all $\boldsymbol{x}^t \in \mathcal{X}$
        $b_i^t \leftarrow \begin{cases} 1 & \text{if } \|\boldsymbol{x}^t - \boldsymbol{m}_i\| = \min_j \|\boldsymbol{x}^t - \boldsymbol{m}_j\| \\ 0 & \text{otherwise} \end{cases}$
    For all $\boldsymbol{m}_i, i = 1, \ldots, k$
        $\boldsymbol{m}_i \leftarrow \sum_t b_i^t \boldsymbol{x}^t / \sum_t b_i^t$
Until $\boldsymbol{m}_i$ converge
```





Image

Clusters on intensity

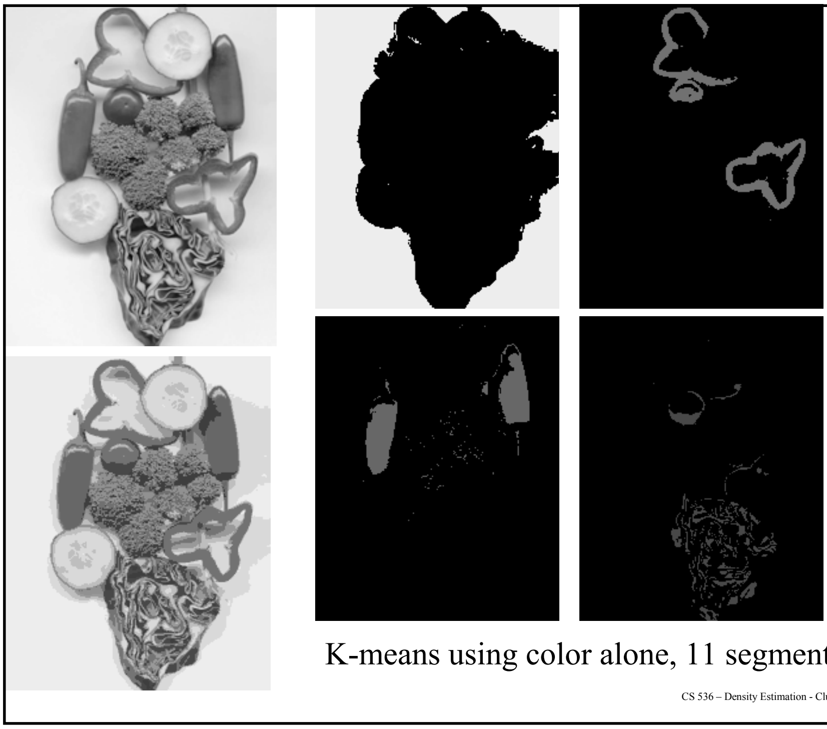
Clusters on color

K-means clustering using intensity alone and color alone
K=5 segmented image is labeled with cluster means

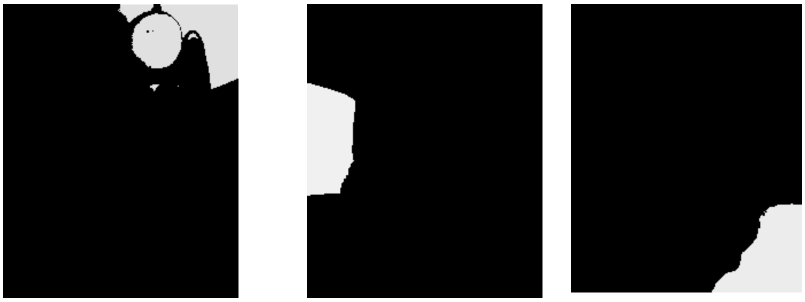
Image

Clusters on color

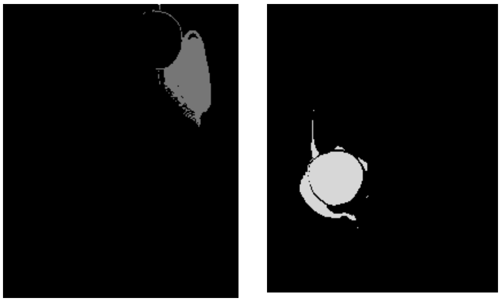
K-means using color alone, 11 segments



K-means using color alone, 11 segments.



K-means using color and position, 20 segments



Hierarchical Clustering

- Cluster based on similarities/distances
- Distance measure between instances $\boldsymbol{x}^r$ and $\boldsymbol{x}^s$

  Minkowski (L_p) (Euclidean for $p = 2$)

$$d_m\left(\mathbf{x}^r, \mathbf{x}^s\right) = \left[\sum_{j=1}^{d}\left(x_j^r - x_j^s\right)^p\right]^{1/p}$$

  City-block distance

$$d_{cb}\left(\mathbf{x}^r, \mathbf{x}^s\right) = \sum_{j=1}^{d}\left|x_j^r - x_j^s\right|$$

Hierarchical Clustering:

- Agglomerative clustering – clustering by merging – bottom-up
  - Each data point is assumed to be a cluster
  - Recursively merge clusters
  - Algorithm:
    - Make each point a separate cluster
    - Until the clustering is satisfactory
      - Merge the two clusters with the smallest *inter-cluster distance*
- Divisive clustering – clustering by splitting – top-down
  - The entire data set is regarded as a cluster
  - Recursively split clusters
  - Algorithm:
    - Construct a single cluster containing all points
    - Until the clustering is satisfactory
      - Split the cluster that yields the two components with the largest *inter-cluster distance*

Hierarchical Clustering:

- Two main issues:
- What is a good inter-cluster distance
  - single-link clustering: distance between the closest elements -> extended clusters
  - complete-link clustering: the maximum distance between elements –> rounded clusters
  - group-average clustering: Average distance between elements – rounded clusters
- How many clusters are there (model selection)
- Dendrograms
  - yield a picture of output as clustering process continues

Agglomerative Clustering

- Start with *N* groups each with one instance and merge two closest groups at each iteration
- Distance between two groups G_i and G_j:
  - Single-link:

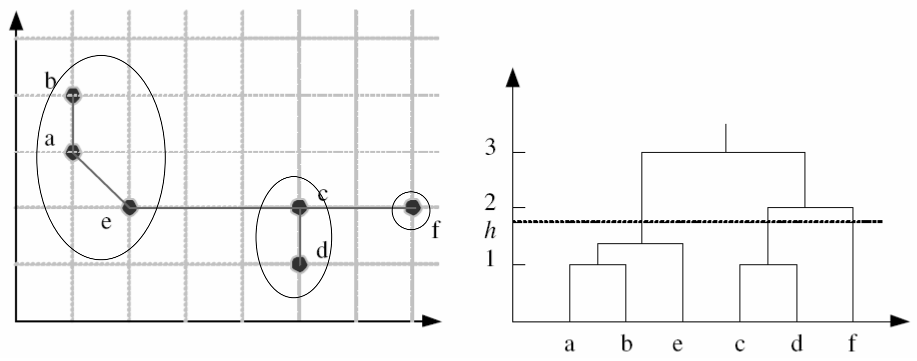
$$d(\mathsf{G}_i, \mathsf{G}_j) = \min_{\mathbf{x}^r \in \mathsf{G}_i, \mathbf{x}^s \in \mathsf{G}_j} d(\mathbf{x}^r, \mathbf{x}^s)$$

  - Complete-link:

$$d(\mathsf{G}_i, \mathsf{G}_j) = \max_{\mathbf{x}^r \in \mathsf{G}_i, \mathbf{x}^s \in \mathsf{G}_j} d(\mathbf{x}^r, \mathbf{x}^s)$$

  - Average-link, centroid

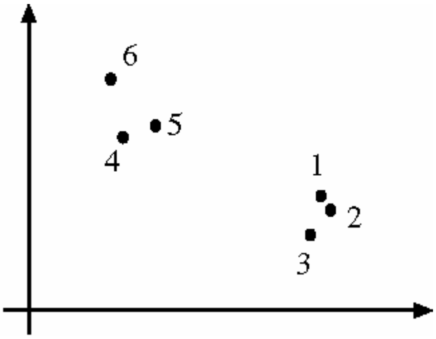
Example: Single-Link Clustering

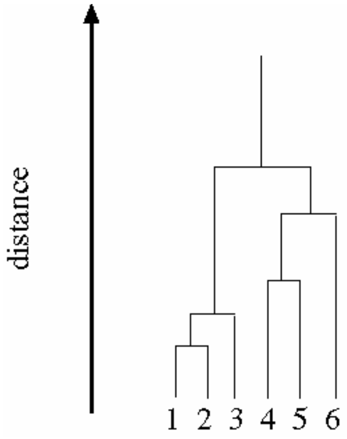


Dendrogram

Choosing k

- Defined by the application, e.g., image quantization
- Plot data (after PCA) and check for clusters
- Incremental (leader-cluster) algorithm: Add one at a time until "elbow" (reconstruction error/log likelihood/intergroup distances)
- Manual check for meaning





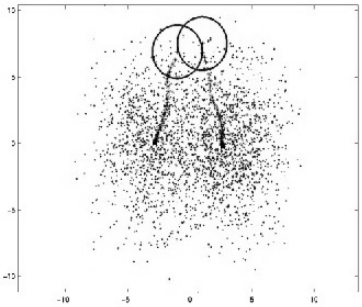
Mean Shift

- Given a sample $S=\{s_i : s_i \in R^n\}$ and a kernel *K,* the sample mean using *K* at point *x*:

$$m(x) = \frac{\sum_i s_i K(s_i - x)}{\sum_i K(s_i - x)}$$

- Iteration of the form $x \leftarrow m(x)$ will lead to the density local mode
- Let *x* is the center of the window

  Iterate until conversion.
  - Compute the sample mean *m(x)* from the samples inside the window.
  - Replace *x* with *m(x)*



Mean Shift

- Given a sample $S=\{s_i : s_i \in R^n\}$ and a kernel K, the sample mean using K at point x:

$$m(x) = \frac{\sum_i s_i K(s_i - x)}{\sum_i K(s_i - x)}$$

- Fukunaga and Hostler 1975 introduced the mean shift as the difference *m(x)-x* using a flat kernel.
- Iteration of the form $x \leftarrow m(x)$ will lead to the density mode
- Cheng 1995 generalized the definition using general kernels and weighted data

$$m(x) = \frac{\sum_i s_i K(s_i - x) w(s_i)}{\sum_i K(s_i - x) w(s_i)}$$

- Recently popularized by D. Comaniciu and P. Meer 99+
- Applications: Clustering[Cheng,Fu 85], image filtering, segmentation[Meer 99] and tracking [Meer 00].

Mean Shift

- Iterations of the form x ← m(x) are called mean shift algorithm.
- If K is a Gaussian (e.g.) and the density estimate using K is

$$\hat{P}(x) = C \sum_i K(x - s_i) w(s_i)$$

- Using Gaussian Kernel $K_\sigma(x)$, the derivative is $K'_\sigma(x) = -\frac{x}{\sigma^2} K_\sigma(x)$

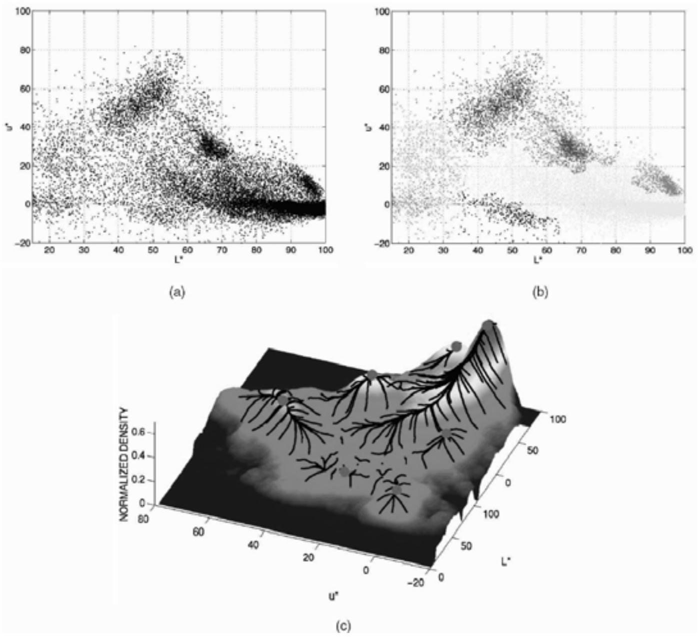
we can show that:

$$\frac{\nabla \hat{P}(x)}{\hat{P}(x)} = m(x) - x$$

- the mean shift is in the gradient direction of the density estimate.

Mean Shift

- The mean shift is in the gradient direction of the density estimate.
- Successive iterations would converge to a local maxima of the density, i.e., a stationary point: *m(x)=x* .
- Mean shift is a steepest-ascent like procedure with variable size steps that leads to fast convergence “well-adjusted steepest ascent”.



Mean shift and Image Filtering

Discontinuity preserving smoothing

- Recall, average or Gaussian filters blur images and do not preserve region boundaries.

Mean shift application:

- Represent each pixel x as spatial location x^s and range x^r (color, intensity)
- Look for modes in the joint spatial-range space
- Use a product of two kernels: a spatial kernel with bandwidth h_s and a range kernel with bandwidth h_r

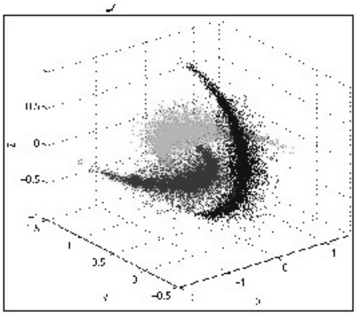
$$K_{h_s,h_r} = k_{h_s}(x^s)k_{h_r}(x^r)$$

- Algorithm:

  For each pixel $x_i=(x_i^s,x_i^r)$

  – apply mean shift until conversion. Let the conversion point be (y_i^s,y_i^r)

  – Assign $z_i = (x_i^s,y_i^r)$ as filter output
- Results: see the paper.

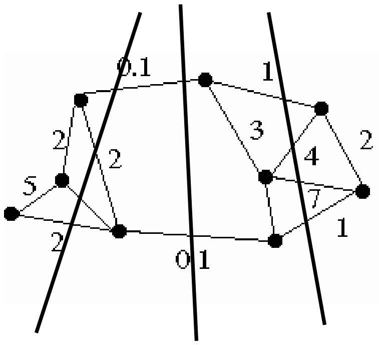


Graph Cut

What is a Graph Cut:

- We have undirected, weighted graph *G=(V,E)*
- Remove a subset of edges to partition the graph into two disjoint sets of vertices *A,B* (two sub graphs):

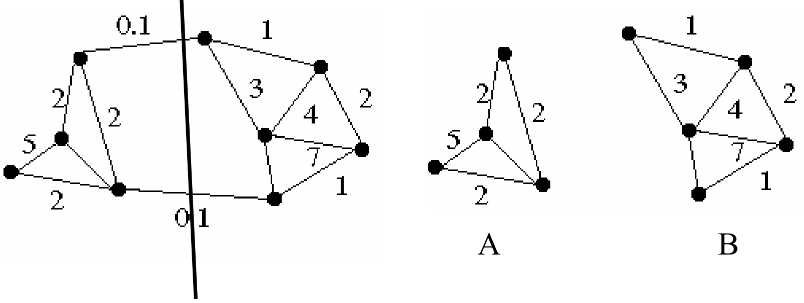
$A \cup B = V, A \cap B = \Phi$



Graph Cut

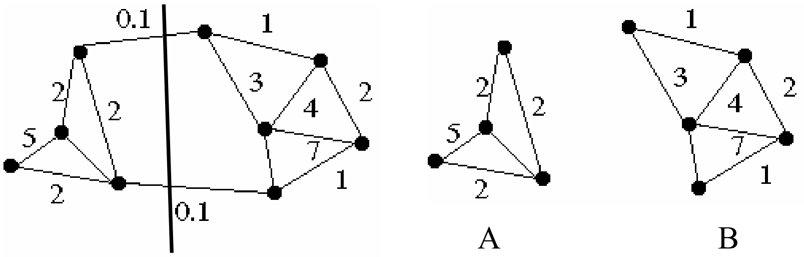
- Each cut corresponds to some cost (cut): sum of the weights for the edges that have been removed.

$$cut(A,B) = \sum_{u \in A, v \in B} w(u,v)$$



Graph Cut

- In many applications it is desired to find the cut with minimum cost: *minimum cut*
- Well studied problem in graph theory, with many applications
- There exists efficient algorithms for finding minimum cuts

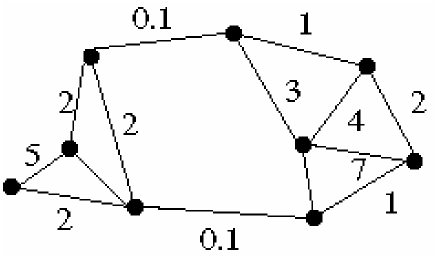


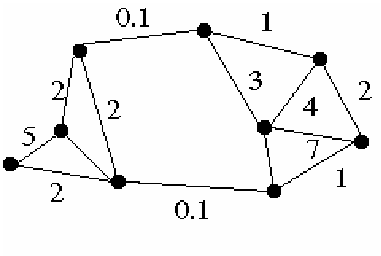
$$cut(A,B) = \sum_{u\in A, v\in B} w(u,v)$$

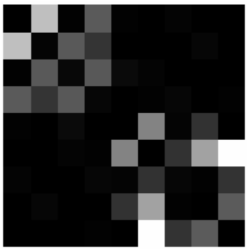
Graph theoretic clustering

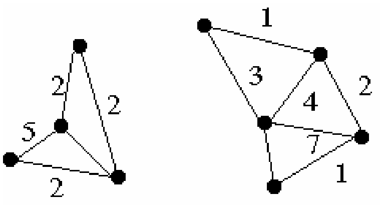
- Represent tokens using a weighted graph
  - Weights reflects similarity between tokens
  - *affinity* matrix
- Cut up this graph to get subgraphs such that:
  - Similarity within sets maximum.
  - Similarity between sets minimum.

⇒ Minimum cut





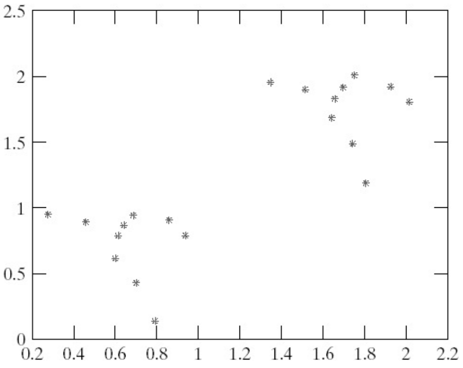


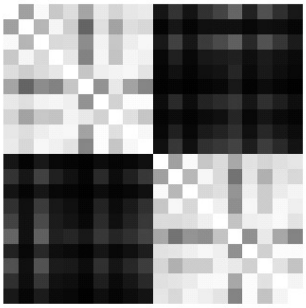


- Use exponential function for edge weights

$$w(x) = e^{-(d(x)/\sigma)^2}$$

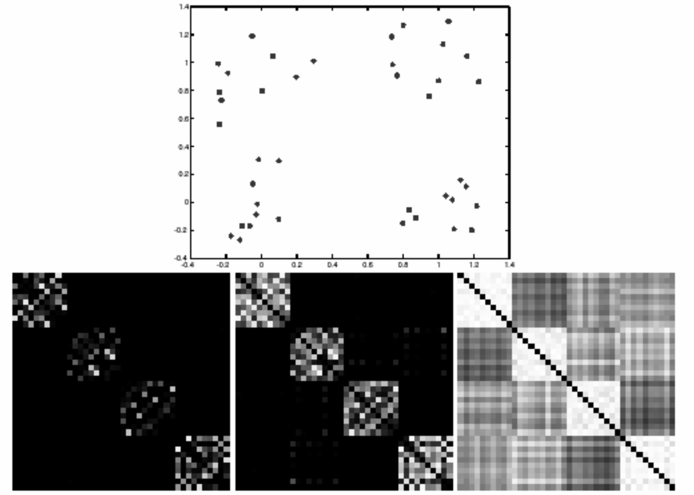
d(x) : feature distance





Scale affects affinity

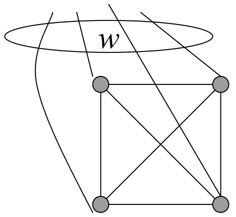
$$w(x) = e^{-(d(x)/\sigma)^2}$$



σ=0.1 σ=0.2 σ=1

Eigenvectors and clustering

- Simplest idea: we want a vector w giving the association between each element and a cluster
- We want elements within this cluster to, on the whole, have strong affinity with one another
- We could maximize



$$w_n^T A w_n$$

Sum of

Association of element i with cluster n ×

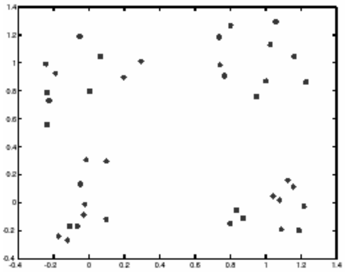
Affinity between i and j ×

Association of element j with cluster n

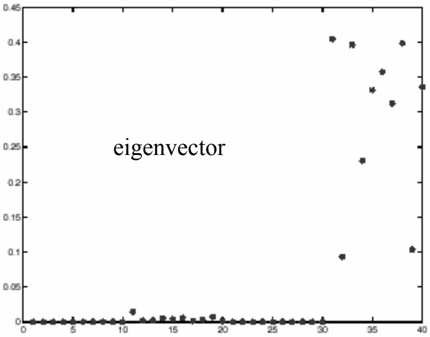
Eigenvectors and clustering

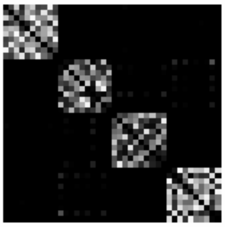
- We could maximize $$w_n^T A w_n$$
- But need the constraint $$w_n^T w_n = 1$$
- Using Lagrange multiplier λ $$w_n^T A w_n + \lambda(w_n^T w_n - 1)$$
- Differentiation $$A w_n = \lambda w_n$$
- This is an eigenvalue problem - choose the eigenvector of A with largest eigenvalue

Example eigenvector



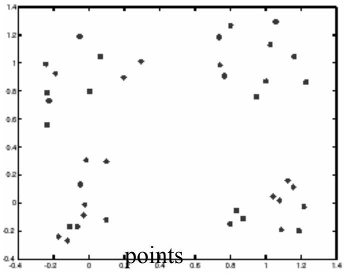
points



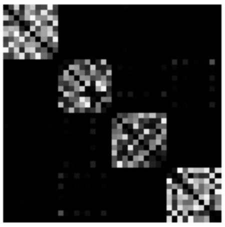


matrix

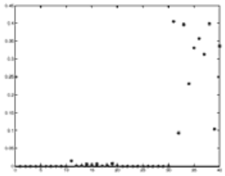
Example eigenvector



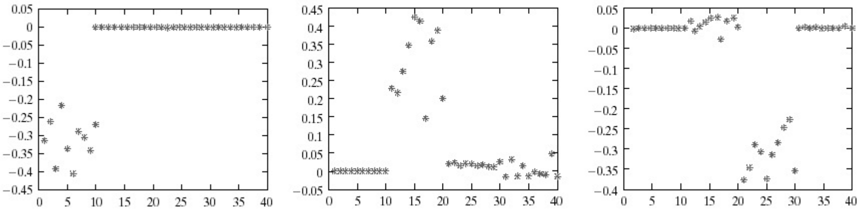
points



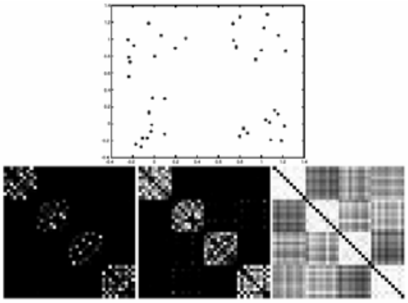
matrix

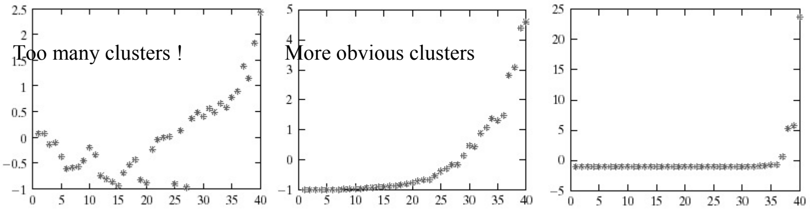


First eigenvectors



The three eigenvectors corresponding to the next three eigenvalues of the affinity matrix





Too many clusters !

More obvious clusters

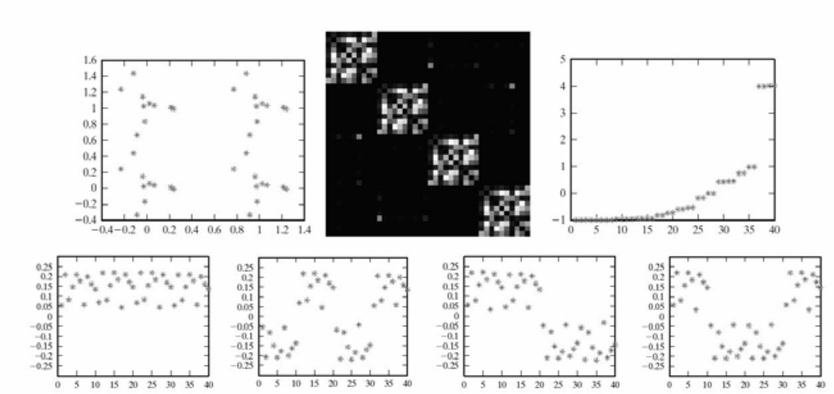
eigenvalues for three different scales for the affinity matrix

More than two segments

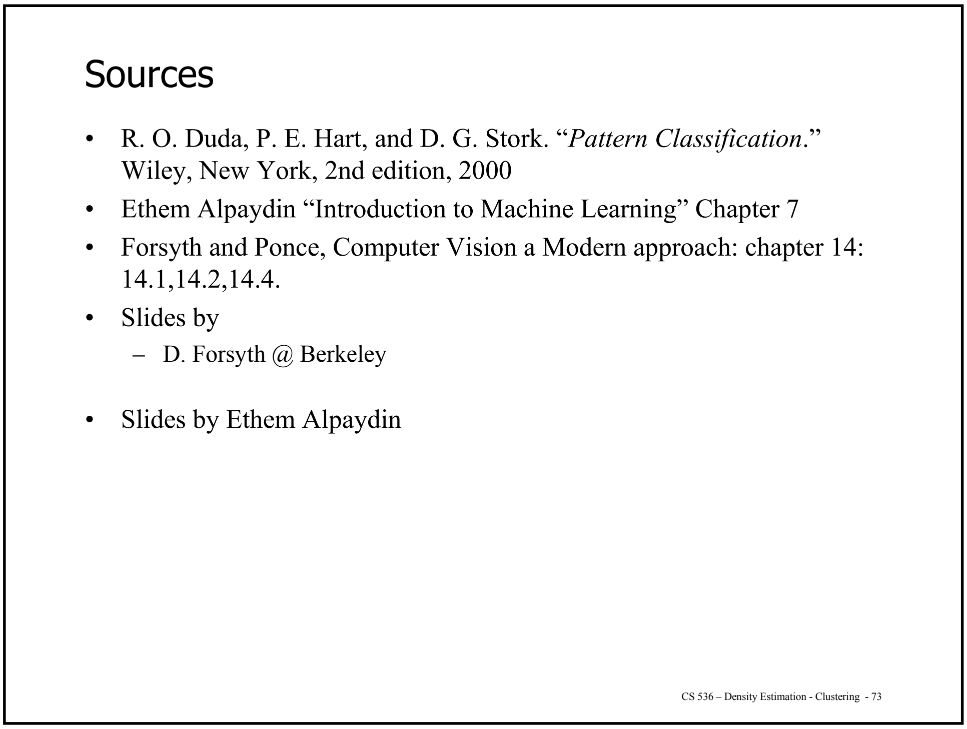
- Two options
  - Recursively split each side to get a tree, continuing till the eigenvalues are too small
  - Use the other eigenvectors

Algorithm

- Construct an Affinity matrix A
- Computer the eigenvalues and eigenvectors of A
- Until there are sufficient clusters
  - Take the eigenvector corresponding to the largest unprocessed eigenvalue; zero all components for elements already clustered, and threshold the remaining components to determine which element belongs to this cluster, (you can choose a threshold by clustering the components, or use a fixed threshold.)
  - If all elements are accounted for, there are sufficient clusters



We can end up with eigenvectors that do not split clusters because any linear combination of eigenvectors with the same eigenvalue is also an eigenvector.



Sources

- R. O. Duda, P. E. Hart, and D. G. Stork. “*Pattern Classification*.” Wiley, New York, 2nd edition, 2000
- Ethem Alpaydin “Introduction to Machine Learning” Chapter 7
- Forsyth and Ponce, Computer Vision a Modern approach: chapter 14: 14.1,14.2,14.4.
- Slides by
  - D. Forsyth @ Berkeley
- Slides by Ethem Alpaydin