Distance and similarity

Hongxin Zhang
zhx@cad.zju.edu.cn

State Key Lab of CAD&CG, ZJU
2013-03-14
In the last lesson

- Data driven decomposition:
  - Data driven curve fitting
  - Singular value decomposition (the power of orthogonal basis)
  - A sort of spectral analysis

- PCA and its related techniques are very useful
In the last lesson

- Mathematical concepts and techniques
  - Least squares (LSQ)
  - Curve fitting
  - Norm (范数) and inner product (内积)
  - Singular value decomposition
  - Eigen vectors and eigen-values
  - Low rank matrix approximation and decomposition
Today’s Talk

- Pre-processing: Distance / metric learning
  - ISO-map
  - LLE

- What is similarity? How to Clustering
  - Spectral based
  - E-M based
    - MOG and K-means
  - Mean shift
Clustering

- Given set of data points, group them, find the overall structure
- Unsupervised learning
- Learn the similarity. Which patient are similar? (or customers, faces, earthquakes, …)
Distance

- Given $n$-dimensional vector $x, y$
- Euclidian ($L^2$ distance)
  
  $$
  dist(x, y; 2) = \left( \sum_{i=1}^{n} (x_i - y_i)^2 \right)^{1/2}
  $$

- $L^1$ distance
  
  $$
  dist(x, y; 1) = \sum_{i=1}^{n} |x_i - y_i|
  $$

- $L^p$ distance (Minkowsky)
  
  $$
  dist(x, y; p) = \left( \sum_{i=1}^{n} (x_i - y_i)^p \right)^{1/p}
  $$

  $$
  dist(x, y; \infty) = \max_i |x_i - y_i|
  $$

切比雪夫
Distance, Norm and inner product

- **Distance**
  
  $$dist(x, y; 2) = \left( \sum_{i=1}^{n} (x_i - y_i)^2 \right)^{1/2}$$

- **Norm**
  
  $$\text{norm}(x; 2) := \|x\|_2 = \left( \sum_{i=1}^{n} x_i^2 \right)^{1/2}$$

- **Inner product**
  
  $$x \cdot y = \sum_{i=1}^{n} x_i y_i$$

  $$\text{norm}(x - y; 2) = \text{dist}(x, y; 2) \quad \text{dist}(x, y; 2) = \left( x \cdot x + y \cdot y - 2 x \cdot y \right)^{1/2}$$

  $$= \left( \|x\|_2^2 + \|y\|_2^2 - 2 x \cdot y \right)^{1/2}$$
Dimensional aware distances

- Along dimension $j$:

$$x_j = \frac{1}{N} \sum_{i=1}^{N} x_{i,j}$$

$$R_j = \max_{i} x_{i,j} - \min_{i} x_{i,j}$$

$$S_j = \left( \frac{1}{N-1} \sum_{i=1}^{N} (x_{i,j} - \bar{x}_j)^2 \right)^{1/2}$$

- Normalized data:

$$x'_{i,j} = \frac{x_{i,j} - \bar{x}_j}{R_j}$$

$$x'_{i,j} = \frac{x_{i,j} - \bar{x}_j}{S_j}$$
M-distance

- Consider the dependency of different dimensions

\[ \text{dist}(x, y; M) = (x - y)^T M^{-1} (x - y) \]

- M is the covariance matrix of data
- Transform invariance
More complex method for distance computing

- PCA …
- structure aware
  - Main idea:
    - Find a suitable mapping
    - Compute distance in mapped space
  - Available techniques
    - MDS + global geodesic distance: ISO-MAP
    - Local distance approximation: LLE
    - …
Classical Multi-dimensional Scaling

- MDS: 多维标度法
- Main idea:
  - Compute (match) distance between samples
  - Use SVD to find similarity
Isomap: (Science 2001) Isometric feature mapping

- Preserve the **intrinsic geometry** of the data.
- Use the **geodesic distances** on manifold between all pairs.

Three steps algorithm
Isomap: Construct Neighborhood Graph

- Determine which points are neighbors, based on the distances $d(i,j)$.
  - K nearest neighbors
  - $\varepsilon$-radius

- Create a graph $G$, with edges between neighbors and distance weights.
Isomap: Compute Shortest Paths

- Estimate the geodesic distances.
- Compute all-pairs shortest paths in G.
- Can be done using Floyd’s algorithm, $O(N^2 \ln N)$.

\[
\begin{align*}
  d_G(i, j) &= d(i, j) \text{ neighbor in } G \quad i, j \\
  d_G(i, j) &= \infty \quad \text{otherwise}
\end{align*}
\]

For $k = 1, 2, \ldots, N$

\[
d_G(i, j) = \min\{ d_G(i, j), d_G(i, k) + d_G(k, j) \}\]
Isomap:
Construct d-dimensional Embedding

Classical MDS with \( d_G(i,j) \), minimize the cost function:

\[
E = \left\| \tau(D_G) - \tau(D_Y) \right\|_{L^2}
\]

where \( D_Y(i, j) = \left\| y_i - y_j \right\| \)

\( D_G(i, j) = d_G(i, j) \)

and

\[
\tau(D) = \frac{-1}{2} (I - \frac{1}{N}) D^{-2} (I - \frac{1}{N})
\]

Solution: take top d eigenvectors of the matrix \( \tau(D_G) \)
Isomap:
Classical **Multi-dimensional Scaling**

\[
X'X = -\frac{1}{2} JE J \quad \quad E: \text{Euclidian distance matrix}
\]

\[
B = -\frac{1}{2} JM J \quad \quad M: \text{Manifold distance matrix}
\]

\[
L(\hat{X}) = \left\| -\frac{1}{2} J (E - M) J \right\| = \left\| \hat{X} \hat{X}' - B \right\|.
\]

\[
B = Q \Lambda Q' \quad \quad \hat{X} = Q+ \Lambda+^{\frac{1}{2}}
\]

\[
c_i = \sum_{a=1}^{m} x_{ia}^2
\]

\[
d_{ij}^2 = \sum_{a=1}^{m} (x_{ia} - x_{ja})^2
\]

\[
E = c1' + 1c' - 2XX'
\]

\[
J = I - \frac{1}{n} 11'
\]

\[
B = -\frac{1}{2} J(c1' + 1c' - 2XX')J = \frac{1}{2} Jc0' - \frac{1}{2} 0c' J + JXX'J = XX'.
\]

**Eigen-structure analysis, SVD again**
Isomap: Classical Multi-dimensional Scaling (2D)

\[ J = \text{eye}(n) - \text{ones}(n)./n; \]
\[ B = -0.5 \times J \times M \times J; \]
% Find largest eigenvalues and their eigenvectors:
\[ [Q, L] = \text{eigs}(B, 2, 'LM'); \]
% Extract the coordinates:
\[ \text{newy} = \sqrt{L(1, 1)} \times Q(:, 1); \]
\[ \text{newx} = \sqrt{L(2, 2)} \times Q(:, 2); \]
Isomap: application texture mapping

Fig. 3. An example of a face flattening. (a) A 3D reconstruction of a face. (b) The flattened texture image of the face.
Isomap: Examples

- $N=2000$ images
  64x64 pixels $K=6$
Isomap:
More Results

Input: 698 images of 64x64
K=7, d=2

Outputs:
Isomap:
More Results

- Same inputs, but this time with $d=3$

698 images of 64x64 $K=7$
Locally Linear Embedding (LLE)

- Recovers global nonlinear structure from locally linear fits.
- Each data point and it’s neighbors is expected to lie on or close to a locally linear patch.
- Each data point is constructed by it’s neighbors:
  \[
  \vec{\hat{X}}_i = \sum_j W_{ij} \vec{X}_j
  \]
  
  \[
  W_{ij} = 0 \text{ if } \vec{X}_j \text{ is not a neighbor of } \vec{X}_i
  \]
LLE: Getting the Reconstruction Weights

- We want to minimize the error function:

$$\varepsilon(W) = \sum_i \left| \bar{X}_i - \sum_j W_{ij} \bar{X}_j \right|^2$$

- With the constrains:

$$W_{ij} = 0 \quad \text{if} \quad \bar{X}_j \text{ is not a neighbor of } \bar{X}_i$$

$$\sum_j W_{ij} = 1$$

- Solution (using Lagrange multipliers):

$$W_j = \sum_k C^{-1}_{jk} (\bar{X} \cdot \vec{\eta}_k + \lambda)$$

$$\lambda = 1 - \sum_{jk} C^{-1}_{jk} (\bar{X} \cdot \vec{\eta}_k) / \sum_{jk} C^{-1}_{jk}$$
LLE: Find Embedded Coordinates

- Choose d-dimensional coordinates, \( Y \), to minimize:

\[
\phi(Y) = \sum_i \left| \vec{Y}_i - \sum_j W_{ij} \vec{Y}_j \right|^2
\]

Under: \( \sum Y_i = 0 \), \( \frac{1}{N} \sum Y Y^T = I \)

Quadratic form:

\[
\phi(Y) = \sum_{ij} M_{ij} (\vec{Y}_i \vec{Y}_j)
\]

where:

\[
M = (I - W)^T (I - W)
\]

- Solution: compute bottom d+1 eigenvectors of M. (discard the last one)
LLE:
Summary

- Input: N data items in D dimension (X).
- Output: d < D dimensional embedding coordinates (Y) for the input points.
LLE:
Algorithm Pseudocode (I)

Find neighbors in X space
For i=1:N
    compute the distance from Xi to every other point Xj
    find the K smallest distances
    assign the corresponding points to be neighbors of Xi
end

http://www.cs.toronto.edu/~roweis/lle/algorithm.html
Solve for reconstruction weights $W$.  
for $i=1:N$  
  create matrix $Z$ consisting of all neighbors of $X_i$  
  subtract $X_i$ from every column of $Z$  
  compute the local covariance $C=Z'*Z$  
  solve linear system $C*w = 1$ for $w$  
  set $W_{ij}=0$ if $j$ is not a neighbor of $i$  
  set the remaining elements in the $i$th row of $W$ equal to $w/\text{sum}(w)$;  
end
LLE:
Algorithm Pseudocode (III)

Compute embedding coordinates $Y$ using weights $W$.

create sparse matrix $M = (I-W)'*(I-W)$

find bottom $d+1$ eigenvectors of $M$ (corresponding to the $d+1$ smallest eigenvalues)

set the $q$-th ROW of $Y$ to be the $q+1$ smallest eigenvector (discard the bottom eigenvector $[1,1,1,1...]$ with eigenvalue zero)
LLE: Example

- $N=8588$ (RGB) images of lips of size $108 \times 84$. $D=27216$
- Num of neighbors $K=16$
Spectral clustering

- The spectral clustering method we define relies on a random walk representation over the points. We construct this in three steps:
  1. a nearest neighbor graph
  2. similarity weights on the edges:
     \[ W_{ij} = \exp\{-\beta \|x_i - x_j\|\} \]
     where \( W_{ii} = 1 \) and the weight is zero for non-edges.
  3. transition probability matrix
     \[ P_{ij} = W_{ij} / \sum_{j'} W_{ij'} \]
Properties of the random walk

- If we start from \( i_0 \), the distribution of points \( i_t \) that we end up in after \( t \) steps is given by

\[
i_1 \sim P_{i_0 i_1}, \quad P_{ij} = \frac{W_{ij}}{W_i}, \quad \text{where } W_i = \sum_j W_{ij}
\]

\[
i_2 \sim \sum_{i_1} P_{i_0, i_1} P_{i_1 i_2} = [P^2]_{i_0 i_2},
\]

\[
i_3 \sim \sum_{i_1} \sum_{i_2} P_{i_0, i_1} P_{i_1 i_2} P_{i_2 i_3} = [P^3]_{i_0 i_3},
\]

\[
\vdots
\]

\[
i_t \sim [P^t]_{i_0 i_t}
\]

where \( P^t = PP \ldots P \) (\( t \) matrix products) and \([ \cdot ]_{ij}\) denotes the \( i, j \) component of the matrix.
The distributions of points we end up in after $t$ steps converge as $t$ increases. If the graph is connected, the resulting distribution is independent of the starting point.

Even for large $t$, the transition probabilities $[P^t]_{i,j}$ have a slightly higher probability of transitioning within “clusters” than across; we want to recover this effect from eigenvalues/vectors.
Eigenvalues/vectors and spectral clustering

- Let \( W \) be the matrix with components \( W_{ij} \) and \( D \) a diagonal matrix such that \( D_{ii} = \sum_j W_{ij} \). Then

\[
P = D^{-1}W
\]

- To find out how \( P^t \) behaves for large \( t \) it is useful to examine the eigen-decomposition of the following symmetric matrix

\[
D^{-\frac{1}{2}}WD^{-\frac{1}{2}} = \lambda_1 z_1 z_1^T + \lambda_2 z_2 z_2^T + \ldots + \lambda_n z_n z_n^T
\]

where the ordering is such that \(|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|\).
Eigenvalues/vectors cont’d

The symmetric matrix is related to $P^t$ since

$$(D^{-\frac{1}{2}}WD^{-\frac{1}{2}}) \cdots (D^{-\frac{1}{2}}WD^{-\frac{1}{2}}) = D^{\frac{1}{2}} (P \cdots P) D^{-\frac{1}{2}}$$

This allows us to write the $t$ step transition probability matrix in terms of the eigenvalues/vectors of the symmetric matrix

$$P^t = D^{-\frac{1}{2}} \left( D^{-\frac{1}{2}}WD^{-\frac{1}{2}} \right)^t D^{\frac{1}{2}}$$

$$= D^{-\frac{1}{2}} \left( \lambda_1^t z_1 z_1^T + \lambda_2^t z_2 z_2^T + \ldots + \lambda_n^t z_n z_n^T \right) D^{\frac{1}{2}}$$

where $\lambda_1 = 1$ and

$$P^\infty = D^{-\frac{1}{2}} \left( z_1 z_1^T \right) D^{\frac{1}{2}}$$
Eigenvalues/vectors and spectral clustering

- We are interested in the largest correction to the asymptotic limit

\[ P^t \approx P^\infty + D^{-\frac{1}{2}} \left( \lambda_2^t z_2 z_2^T \right) D^{\frac{1}{2}} \]

Note: \([z_2 z_2^T]_{ij} = z_{2i} z_{2j}\) and thus the largest correction term increases the probability of transitions between points that share the same sign of \(z_{2i}\) and decreases transitions across points with different signs.

- Binary spectral clustering: we divide the points into clusters based on the sign of the elements of \(z_2\)

\[ z_{2j} > 0 \Rightarrow \text{cluster 1}, \text{otherwise cluster 0} \]
Spectral clustering: example

Components of the eigenvector corresponding to the second largest eigenvalue
Reference papers of SC


And more about image segmentations …
- Graph cut
- Mean-shift
Classical methods on cluster distance

- A *linkage* method: we have to be able to measure distances between clusters of examples $C_k$ and $C_l$
  
  a) Single linkage: Nearest neighbor

  \[ d_{kl} = \min_{i \in C_k, j \in C_l} d(x_i, x_j) \]

  b) Average linkage:

  \[ d_{kl} = \frac{1}{|C_l| |C_k|} \sum_{i \in C_k, j \in C_l} d(x_i, x_j) \]

  c) Centroid linkage:

  \[ d_{kl} = d(\bar{x}_k, \bar{x}_l), \quad \bar{x}_l = \frac{1}{|C_l|} \sum_{i \in C_l} x_i \]
Hierarchical (bottom-up) clustering

- Hierarchical agglomerative clustering: we sequentially merge the pair of “closest” points/clusters

- The procedure
  1. Find two closest points (clusters) and merge them
  2. Proceed until we have a single cluster (all the points)

- Two prerequisites:
  1. distance measure $d(x_i, x_j)$ between two points
  2. distance measure between clusters (cluster linkage)
Hierarchical (bottom-up) clustering

- A dendrogram representation of hierarchical clustering

The height of each pair represents the distance between the merged clusters; the specific linear ordering of points is chosen for clarity.
Clustering and image segmentation

Mean-shift segmentation
Regression revisit: Polynomial Curve Fitting

\[ y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j \]

\[ y(x, \mathbf{w}) = w_0 h_0(x) + w_1 h_1(x) + w_2 h_2(x) + \ldots + w_M h_M(x) = \sum_{j=0}^{M} w_j h_j(x) \]

The normal equation is:

\[ \mathbf{w} = (\mathbf{H}^T \mathbf{H})^{-1} \mathbf{H}^T \mathbf{y} \]
Regression revisit: alternative approach

\[ P(x) = \sum_{z=1}^{K} P(Z = z \mid \pi) N(x \mid \mu_z, \Sigma_z) \]
Mixtures of Gaussians

- Mixture distribution:
  - Assume $P(x)$ is a mixture of $K$ different Gaussians
  - Assume each data point, $x$ is generated by 2-step process
    - Choose one of the $K$ Gaussians as label $z$
    - Generate $x$ according to the Gaussian $N(\mu_z, \Sigma_z)$

\[
P(x) = \sum_{z=1}^{K} P(Z = z | \pi) N(x | \mu_z, \Sigma_z)
\]

- What object function shall we optimize?
  - Maximize data likelihood
Mixtures of Gaussians (cont.)

- Multivariate Gaussian model

\[ p(x|\mu, \Sigma) = \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu) \right\} \]

- How to generate it?

\[ F_{\mu,\sigma^2}(x) = \int_{-\infty}^{x} p(z|\mu, \sigma^2)dz \]

\[ u \sim \text{Uniform}(0, 1) \implies x = F_{\mu,\sigma^2}^{-1}(u) \sim p(x|\mu, \sigma^2) \]

\[ z_i \sim p(z_i|\mu = 0, \sigma^2 = 1), \quad z = [z_1, \ldots, z_d]^T \]

\[ x = \Sigma^{1/2}z + \mu \]
Multi-variate density estimation

- A mixture of Gaussians model

\[ p(\mathbf{x}|\theta) = \sum_{i=1}^{k} p_j \ p(\mathbf{x}|\mu_j, \Sigma_j) \]

where \( \theta = \{p_1, \ldots, p_k, \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k\} \) contains all the parameters of the mixture model. \( \{p_j\} \) are known as mixing proportions or coefficients.
Mixtures of Gaussians: Wishart distribution

- A mixture of Gaussian Model:
  \[ p(x|\theta) = \sum_{i=1}^{k} p_j p(x|\mu_j, \Sigma_j) \]
  \[ \theta = \{ p_1, \ldots, p_k, \mu_1, \ldots, \mu_k, \Sigma_1, \ldots, \Sigma_k \} \]

- Wishart prior
  \[ P(\Sigma|S, n') \propto \frac{1}{|\Sigma|^{n'/2}} \exp \left( -\frac{n'}{2} \text{Trace}(\Sigma^{-1} S) \right) \]
  \[ S = \text{“prior” covariance matrix} \]
  \[ n' = \text{equivalent sample size} \]
Mixture density

- Data generation process:

\[ p(x|\theta) = \sum_{j=1,2} P(y=j) \cdot p(x|y=j) \quad \text{(generic mixture)} \]

\[ = \sum_{j=1,2} p_j \cdot p(x|\mu_j, \Sigma_j) \quad \text{(mixture of Gaussians)} \]

- Any data point \( x \) could have been generated in two ways
Mixture density

- If we are given just $x$ we don’t know which mixture component this example came from

$$
p(x|\theta) = \sum_{j=1,2} p_j p(x|\mu_j, \Sigma_j)
$$

- We can evaluate the posterior probability that an observed $x$ was generated from the first mixture component

$$
P(y = 1|x, \theta) = \frac{P(y = 1) \cdot p(x|y = 1)}{\sum_{j=1,2} P(y = j) \cdot p(x|y = j)} = \frac{p_1 p(x|\mu_1, \Sigma_1)}{\sum_{j=1,2} p_j p(x|\mu_j, \Sigma_j)}
$$

- This solves a *credit assignment* problem
Mixture density: posterior sampling

- Consider sampling $x$ from the mixture density, then $y$ from the posterior over the components given $x$, and finally $x'$ from the component density indicated by $y$:

  $x \sim p(x|\theta)$
  
  $y \sim P(y|x, \theta)$
  
  $x' \sim p(x'|y, \theta)$

Is $y$ a fair sample from the prior distribution $P(y)$?
Is $x'$ a fair sample from the mixture density $p(x'|\theta)$?
Mixture density estimation

- Suppose we want to estimate a two component mixture of Gaussians model.
  \[ p(x|\theta) = p_1 p(x|\mu_1, \Sigma_1) + p_2 p(x|\mu_2, \Sigma_2) \]

- If each example \( x_i \) in the training set were labeled \( y_i = 1, 2 \) according to which mixture component (1 or 2) had generated it, then the estimation would be easy.

- Labeled examples \( \Rightarrow \) no credit assignment problem
Mixture density estimation

When examples are already assigned to mixture components (labeled), we can estimate each Gaussian independently:

- If $\hat{n}_j$ is the number of examples labeled $j$, then for each $j = 1, 2$ we set

\[
\hat{p}_j \leftarrow \frac{\hat{n}_j}{n}
\]
\[
\hat{\mu}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i:y_i=j} x_i
\]
\[
\hat{\Sigma}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i:y_i=j} (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^T
\]
Mixture density estimation: credit assignment

- Of course we don’t have such labels ... but we can guess what the labels might be based on our current mixture distribution.

- We get soft labels or posterior probabilities of which Gaussian generated which example:

\[ \hat{p}(j|i) \leftarrow P(y_i = j|x_i, \theta) \]

where \( \sum_{j=1,2} \hat{p}(j|i) = 1 \) for all \( i = 1, \ldots, n \).

- When the Gaussians are almost identical (as in the figure), \( \hat{p}(1|i) \approx \hat{p}(2|i) \) for almost any available point \( x_i \).

Even slight differences can help us determine how we should modify the Gaussians.
The Expectation-Maximization algorithm

**E-step:** softly assign examples to mixture components

\[
\hat{p}(j|i) \leftarrow P(y_i = j|x_i, \theta), \text{ for all } j = 1,2 \text{ and } i = 1,\ldots,n
\]

**M-step:** re-estimate the parameters (separately for the two Gaussians) based on the soft assignments.

\[
\hat{n}_j \leftarrow \sum_{i=1}^{n} \hat{p}(j|i) = \text{Soft \# of examples labeled } j
\]

\[
\hat{\mu}_j \leftarrow \frac{\hat{n}_j}{\hat{n}_j} \sum_{i=1}^{n} \hat{p}(j|i) x_i
\]

\[
\hat{\Sigma}_j \leftarrow \frac{1}{\hat{n}_j} \sum_{i=1}^{n} \hat{p}(j|i) (x_i - \hat{\mu}_j)(x_i - \hat{\mu}_j)^T
\]
The EM-algorithm

- Each iteration of the EM-algorithm *monotonically* increases the (log-)likelihood of the $n$ training examples $x_1, \ldots, x_n$:

$$
\log p(\text{data} | \theta) = \sum_{i=1}^{n} \log \left( \frac{p(x_i | \theta)}{p(x_i | \mu_1, \Sigma_1) + p(x_i | \mu_2, \Sigma_2)} \right)
$$

where $\theta = \{p_1, p_2, \mu_1, \mu_2, \Sigma_1, \Sigma_2\}$ contains all the parameters of the mixture model.
The EM algorithm

- The EM-algorithm finds a local maximum of $l(\theta; D)$

**E-step:** evaluate the expected complete log-likelihood

$$J(\theta; \theta^{(t)}) = \sum_{i=1}^{n} E_{j \sim P(j|x_i, \theta^{(t)})} \log \left( p_j p(x_i | \mu_j, \Sigma_j) \right)$$

$$= \sum_{i=1}^{n} \sum_{j=1,2} P(j|x_i, \theta^{(t)}) \log \left( p_j p(x_i | \mu_j, \Sigma_j) \right)$$

**M-step:** find the new parameters by maximizing the expected complete log-likelihood

$$\theta^{(t+1)} \leftarrow \arg \max_{\theta} J(\theta; \theta^{(t)})$$
Regularized EM algorithm

- To maximize a penalized (regularized) log-likelihood

\[ l'(\theta; D) = \sum_{i=1}^{n} \log p(x_i|\theta) + \log p(\theta) \]

we only need to modify the M-step of the EM-algorithm. Specifically, in the M-step, we find \( \theta \) that maximize a penalized expected complete log-likelihood:

\[ J(\theta; \theta^{(t)}) = \sum_{i=1}^{n} \mathbb{E}_{j \sim P(j|x_i, \theta^{(t)})} \log \left( p_j p(x_i|\mu_j, \Sigma_j) \right) \]

\[ + \log p(p_1, p_2) + \log p(\Sigma_1) + \log p(\Sigma_1) \]

where, for example, \( p(p_1, p_2) \) could be a Dirichlet and each \( p(\Sigma_j) \) a Wishart prior.
Selecting the number of components

- As a simple strategy for selecting the appropriate number of mixture components, we can find $k$ that minimize the following asymptotic approximation to the description length:

$$DL \approx - \log p(\text{data}|\hat{\theta}_k) + \frac{d_k}{2} \log(n)$$

where $n$ is the number of training points, $\hat{\theta}_k$ is the maximum likelihood parameter estimate for the $k$-component mixture, and $d_k$ is the (effective) number of parameters in the $k$-mixture.
Mixture density estimation: example
K-means clustering

Given data \( <x_1 \ldots x_n> \), and \( K \), assign each \( x_i \) to one of \( K \) clusters, \( C_1 \ldots C_K \), minimizing

\[
J = \sum_{j=1}^{K} \sum_{x_i \in C_j} ||x_i - \mu_j||^2
\]

Where \( \mu_j \) is mean over all points in cluster \( C_j \)

K-Means Algorithm:

Initialize \( \mu_1 \ldots \mu_K \) randomly

Repeat until convergence:

1. Assign each point \( x_i \) to the cluster with the closest mean \( \mu_j \)
2. Calculate the new mean for each cluster

\[
\mu_j \leftarrow \frac{1}{|C_j|} \sum_{x_i \in C_j} x_i
\]
K-Means vs. Mixture of Gaussians

- Both are iterative algorithms to assign points to clusters

- Objective function
  - K Means: minimize
  - MoG: maximize likelihood

- MoG the more general formulation
  - Equivalent to K Means when $\Sigma_k = \sigma I$, and $\sigma \rightarrow 0$
Disadvantage of K-means and MOG

- The result is sensitive to the initial data
- How to determine the number of clusters
Mean shift

- First proposed by Fukunaga in 1970’s
- Wildly used since 1998
  - In computer vision
  - And other areas

The following several slides is mainly from:
Density estimation

CD Rates

True density

Percent

7.4 7.6 7.8 8 8.2 8.4 8.6 8.8 9
Histogram representation
Histogram-based estimates

- You can use a variety of fitting techniques to produce a curve from a histogram
  - Lines, polynomials, splines, etc.
  - Also called regression/function approximation
  - Normalize to make this a density

- If you know quite a bit about the underlying density you can compute a good bin size
  - But that’s rarely realistic in vision
  - And defeats the whole purpose of the non-parametric approach
Nearest-neighbor estimate

- To estimate the density, count the number of nearby data points
  - Like histogramming with sliding bins
  - Avoid bin-placement artifacts

\[
p(x) = \frac{\# \{ x_i \mid \|x_i - x\| \leq \varepsilon \}}{N}
\]

- We can fix \( \varepsilon \) and compute this quantity, or we can fix the quantity and compute \( \varepsilon \)
Parzen estimation

- Each observed data increases our estimate of the probability nearby
  - Simplest case: raise the probability uniformly within a fixed radius
    - Place a fixed-height “box” at each data point, add them up to get the density estimate
  - This is nearest neighbor with fixed $\varepsilon$

- More generally, you can use some slowly decreasing function (such as a Gaussian)
  - Called Kernel function
Parzen example

from Hastie et al.
Importance of scale
Mean shift algorithm

- Non-parametric method to compute the nearest mode of a distribution
  - Density increases as we get near “center”
Image and histogram
Local modes
Kernel Density Estimation

- Multivariate kernel density estimation

\[
f(x) = \frac{1}{nh^d} \sum_{i=1}^{n} \frac{1}{h} K\left( \frac{x - x_i}{h} \right)
\]

- Kernels
  - Gaussian
    \[
    K_N = (2\pi)^{-d/2} \exp\left( -\frac{1}{2} \|x\|^2 \right)
    \]
  - Epanechnikov
    \[
    K_E = \begin{cases} 
    1/2c_d^{-1}(d + 2)(1 - \|x\|^2) & \text{if } \|x\| < 1 \\
    0 & \text{otherwise}
    \end{cases}
    \]
Finding Mean-Shift Vector

- Gradient computation
  - For symmetric kernel

\[ \hat{\nabla} f(x) = \frac{2}{nh^{d+2}} \sum_{i=1}^{n} K_N \left( \frac{x - x_i}{h} \right) \left[ \frac{\sum_{i=1}^{n} x_i K_N \left( \frac{x - x_i}{h} \right)}{\sum_{i=1}^{n} K_N \left( \frac{x - x_i}{h} \right)} - x \right] \]

- Always converges to the local maximum!
The mean shift procedure

- Give a point $x$
  1. Compute the mean shift vector
     \[
     \hat{\nabla} f(x) = \frac{2}{nh^{d+2}} \sum_{i=1}^{n} K_{N} \left( \frac{x - x_i}{h} \right) \left[ \sum_{i=1}^{n} x_i K_{N} \left( \frac{x - x_i}{h} \right) - x \right]
     \]
  2. Translate density estimation window:
     \[
     x^{(t+1)} \leftarrow x^{(t)} + \hat{\nabla} f(x^{(t)})
     \]
  3. Iterate steps 1. and 2. until convergence i.e., \( \hat{\nabla} f(x) \rightarrow 0 \)
Applications

- Pattern recognition
  - Clustering
- Image processing
  - Filtering
  - Segmentation
- Density estimation
  - Density approximation
  - Particle filter
- Mid-level application
  - Tracking
  - Background subtraction
Summary

- The distance computing plays an important role in data analysis to find out
  - the suitable similarity measurement
  - the intrinsic structure of data

- Further reading on metric learning
- In the next lesson, we will explore more complex data with structure