

CS4770

Pattern Recognition

Unsupervised Learning & Clustering

P. J. Narayanan
Monsoon 2006

Why?

- Creating labelled training data is time-consuming and expensive.
- Unorganized data is out there. Can we find some quick structure?
- This structure can be the basis of tuned, supervised classifiers.
- **Data mining:** When the nature and type of the data in a large database is not known, use unsupervised methods to get quick structure.

Assumptions

- The number c of classes is known.
- Prior probabilities $P(\omega_i)$ are known.
- The parametric forms of $p(\mathbf{x}|\omega_i, \theta_i)$ are known.
- Parameter vectors $\theta_1, \dots, \theta_c$ are not known.
- Set of samples $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ without labels are given.

Probability Mixture Model

- We can write:

$$p(\mathbf{x}|\theta) = \sum_{i=0}^c p(\mathbf{x}|\omega_i, \theta_i)P(\omega_i)$$

- Gives a mixture model or mixture density.
 $p(\mathbf{x}|\omega_i, \theta_i)$ are the **component densities** and $P(\omega_i)$ are the **mixing parameters**.
- **Identifiability:** A density $p(\mathbf{x}|\theta)$ is identifiable if $\theta \neq \theta'$ implies $p(\mathbf{x}|\theta) \neq p(\mathbf{x}|\theta')$ for some \mathbf{x} .
- Identifiability is the property of the model.

- Identifiability is a problem for discrete density functions.
- We concentrate only on identifiable mixtures.

Maximum-Likelihood Estimation

- Samples are drawn independently: $p(D|\theta) = \prod_{i=1}^n p(\mathbf{x}_i|\theta)$.
- Take the logarithm of likelihood: $l(\theta) = \sum_{i=1}^n \ln p(\mathbf{x}_i|\theta)$.
- Set gradients with respect to θ_i to zero and solve for maximum likelihood estimates $\hat{\theta}_i$ values.
- Results in:

$$\sum_{k=1}^n P(\omega_i|\mathbf{x}_k, \hat{\theta}) \nabla_{\theta_i} \ln p(\mathbf{x}_k|\omega_i, \hat{\theta}_i) = 0, \quad i = 1 \cdots c$$

- Priors $P(\omega_i)$ can also be estimated as:

$$\hat{P}(\omega_i) = \frac{1}{n} \sum_{k=1}^n \hat{P}(\omega_i | \mathbf{x}_k, \hat{\theta})$$

and

$$\sum_{k=1}^n \hat{P}(\omega_i | \mathbf{x}_k, \hat{\theta}) \nabla_{\theta_i} \ln p(\mathbf{x}_k | \omega_i, \hat{\theta}_i) = 0$$

with

$$\hat{P}(\omega_i | \mathbf{x}_k, \hat{\theta}) = \frac{p(\mathbf{x}_k | \omega_i, \hat{\theta}_i) \hat{P}(\omega_i)}{\sum_j p(\mathbf{x}_k | \omega_j, \hat{\theta}_j) \hat{P}(\omega_j)}$$

- Maximize probability densities with respect to θ_i .

For Normal Densities

- When μ_i is unknown but Σ_i are known:

$$\nabla_{\mu_i} \ln p(\mathbf{x}_k | \omega_i, \mu_i) = \Sigma_i^{-1} (\mathbf{x}_k - \mu_i)$$

- We get:

$$\hat{\mu}_i = \frac{\sum_{k=1}^n P(\omega_i | \mathbf{x}_k, \hat{\mu}) \mathbf{x}_k}{\sum_{k=1}^n P(\omega_i | \mathbf{x}_k, \hat{\mu})}$$

- ML estimate of μ_i is the weighted average of \mathbf{x}_k with its likelihood in ω_i as the weight.
- Iterative procedure to estimate μ_i using a gradient ascent procedure.
- Similarly when both μ_i and Σ_i are unknown.

- We get:

$$\hat{\mu}_i = \frac{\sum_{k=1}^n P(\omega_i | \mathbf{x}_k, \hat{\theta}) \mathbf{x}_k}{\sum_{k=1}^n P(\omega_i | \mathbf{x}_k, \hat{\theta})}$$

k -Means Clustering

- Compute Euclidean distances $\|\mathbf{x}_k - \mu_i\|$ for each i . Assign \mathbf{x}_k to ω_m for which the distance is minimum.
- Amounts to: $\hat{P}(\omega_i | \mathbf{x}_k, \hat{\theta}) = 1$ if $i = m$ and 0 otherwise.
- Iterative algorithm to find μ_i :
 - Initialize μ_1, \dots, μ_c
 - do**
 - Classify n samples to nearest μ_i
 - Recompute μ_i based on new labels
 - until** no change in μ_i .

- Think of this as estimating the ML estimates of the means. It also minimizes a criterion function based on squared errors.

Fuzzy k-Means Clustering

- If a point belongs to multiple clusters to different degrees.
- The degree of belonging is precisely: $\hat{P}(\omega_i | \mathbf{x}_k, \hat{\theta})$.

- Minimize (such that $\sum_{i=1}^c \hat{P}(\omega_i | x_j, \hat{\theta}) = 1$)

$$J = \sum_{i=1}^c \sum_{j=1}^n |\hat{P}(\omega_i | \mathbf{x}_j, \hat{\theta})|^b \|\mathbf{x}_j - \mu_i\|^2$$

- Solution: (for $d_{ij} = \|\mathbf{x}_j - \mu_i\|^2$)

$$\mu_i = \frac{\sum_{j=1}^n |\hat{P}(\omega_i | \mathbf{x}_j, \hat{\theta})|^b \mathbf{x}_j}{\sum_{j=1}^n |\hat{P}(\omega_i | \mathbf{x}_j, \hat{\theta})|^b} \quad \text{and} \quad \hat{P}(\omega_i | \mathbf{x}_j) = \frac{(1/d_{ij})^{1/(b-1)}}{\sum_{k=1}^c (1/d_{kj})^{1/(b-1)}}$$

- Reduces to k -Means when $\hat{P}(\omega_i | x_j) = 1$ for nearest i .

Data Description & Clustering

- Discover structure in the data.
- For normally distributed points, mean and variance suffice. Mean gives the first order properties and variance gives the second order properties.
- Meaningless if assumption doesn't hold. Different organizations of points with same μ and Σ could be totally different.
- For data with c normal clusters, estimate mixture probabilities.

- Or use Parzen windows to estimate distributions without knowing the parametric form.
- Clustering: Detect classes/subclasses in the data automatically.
- Divide data into natural groups with strong internal similarities and strong inter-group differences.
- Outline: Define a criterion function in terms of the assumed clusters and optimize it.

Clustering: Similarity Measure

- Assign points to the same group if the distance between them is below a threshold. (Or similarity above a threshold)
- High threshold will group all points into one cluster; low threshold will favour n clusters!
- Problem: partition $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ to subsets D_1, D_2, \dots, D_c . Samples in D_i are similar to each other. Samples in different D_i s are dissimilar.
- The similarity function itself also is very important.

- Invariances: Rotation, translation, scaling, and other transformations could be present. Along some axes.
- Rotation of all points: Apply PCA and describe using the eigenvector directions.
- Length in cm or mm : Scaling along the axis!
- Solution: Normalize to have zero mean and unit variance. Assumes min and max of each axis has same “meaning”.

Distance Measures

- Minkowski: $d_k(\mathbf{x}, \mathbf{y}) = \left[\sum_{i=1}^d |x_i - y_i|^k \right]^{1/k} = L_k(\mathbf{x}, \mathbf{y})$
- Manhattan distance, Euclidean distance, etc.
- Similarity: Normalized inner product $s(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$
- For binary features: $s(\mathbf{x}, \mathbf{y}) = \mathbf{x}^\top \mathbf{y} / d$.
- Tanimoto similarity: $s(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}^\top \mathbf{y}}{\mathbf{x}^\top \mathbf{x} + \mathbf{y}^\top \mathbf{y} - \mathbf{x}^\top \mathbf{y}}$.
- Feature vectors define subsets with binary features.
Tanimoto similarity between sets: $s(\mathbf{x}, \mathbf{y}) = \frac{|A \cap B|}{|A \cup B|}$.
Tanimoto distance between sets: $d(A, B) = 1 - \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$.

- Hausdorff Distance from set A to set B:

$$h(A, B) = \max_{\mathbf{x} \in A} \min_{\mathbf{y} \in B} d(\mathbf{x}, \mathbf{y})$$

- Hausdorff Distance between sets A and B:

$$H(A, B) = \max[h(A, B), h(B, A)]$$

- Similarity and distance measures: $s(\mathbf{x}, \mathbf{y}) = -d(\mathbf{x}, \mathbf{y})$.

If in the range $[-1, 1]$, $s(\mathbf{x}, \mathbf{y}) = 1 - d(\mathbf{x}, \mathbf{y})$.

- Combining measures from feature components is tricky!
One could be colour and other shape!

Clustering: Criterion Function

- Partition $D = \{\mathbf{x}_1, \dots, \mathbf{x}_n\}$ to subsets D_1, D_2, \dots, D_c .
- Sum of squared error: $J_e = \sum_{i=1}^c \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|^2$
- Induces *minimum variance partitions* on D .
- MSE is sensitive to outliers. Prefers equal sized clusters.
- Rewrite as: $J_e = \sum_{i=1}^c n_i \bar{s}_i$ and $\bar{s}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \sum_{\mathbf{y} \in D_i} L_2(\mathbf{x}, \mathbf{y})$
- Can use other distance/similarity measures instead of L_2 .
Yields other minimum variance partitions.

Scatter Matrices

- Define: $S_i = \sum_{\mathbf{x} \in D_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^\top$, $S_W = \sum_{i=1}^c S_i$,
 $S_B = \sum_{i=1}^c n_i (\mathbf{m}_i - \mathbf{m})(\mathbf{m}_i - \mathbf{m})^\top$, $S_T = \sum_{i=0}^n (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^\top$
- $S_T = S_W + S_B$, independent of the partitioning. Hence minimize S_W or maximize S_B .
- Scalar measures are convenient. What are they?
- Trace: $\text{tr}[S_W] = \sum_{i=1}^c \text{tr}[S_i] = \sum_{i=1}^c \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|^2 = J_e$
- Minimize $\text{tr}[S_W]$ or maximize $\text{tr}[S_B] = \sum_{i=1}^c n_i \|\mathbf{m}_i - \mathbf{m}\|^2$

- Determinant: Minimize $J_d = |S_W|$
- Different invariances for each of these. J_e is sensitive to scaling. J_d is not.
- $S_W^{-1}S_B$ is invariant to a general non-singular linear transform of the vectors.
- Maximize $\text{tr}[S_W^{-1}S_B]$.
- Easier to minimize $\text{tr}[S_T^{-1}S_W]$. Or $|S_W|/|S_T|$.
- Invariant clusterings are more susceptible local extrema.

Iterative Optimization

- Minimize a criterion function such as:

$$J = \sum_{i=1}^c \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|^2$$

- For fixed n , the combinatorics of clustering is finite but exponential. Evaluating all is impractical.
- Define: $J_i = \sum_{\mathbf{x} \in D_i} \|\mathbf{x} - \mathbf{m}_i\|^2$ where $\mathbf{m}_i = \frac{1}{n_i} \sum_{\mathbf{x} \in D_i} \mathbf{x}$
- Assume a sample $\hat{\mathbf{x}}$ is moved from D_i to D_j .
- $\mathbf{m}'_j = \mathbf{m}_j + \frac{\hat{\mathbf{x}} - \mathbf{m}_j}{n_j + 1}$ and $\mathbf{m}'_i = \mathbf{m}_i - \frac{\hat{\mathbf{x}} - \mathbf{m}_i}{n_i - 1}$

- $$\begin{aligned}
 J'_j &= \|\hat{\mathbf{x}} - \mathbf{m}'_j\|^2 + \sum_{\mathbf{x} \in D_j} \|\mathbf{x} - \mathbf{m}'_j\|^2 \\
 &= \left\| \frac{n_j}{n_j+1} (\hat{\mathbf{x}} - \mathbf{m}'_j) \right\|^2 + \sum \left\| \mathbf{x} - \mathbf{m}_j - \frac{\hat{\mathbf{x}} - \mathbf{m}_j}{n_j+1} \right\|^2 \\
 &= \left\| \frac{n_j}{n_j+1} (\hat{\mathbf{x}} - \mathbf{m}'_j) \right\|^2 + \sum \|\mathbf{x} - \mathbf{m}_j\|^2 - 2K \sum (\mathbf{x} - \mathbf{m}_j) + \frac{n_j^2}{(n_j+1)^2} \|\hat{\mathbf{x}} - \mathbf{m}'_j\|^2
 \end{aligned}$$

Giving $J'_j = J_j + \frac{n_j}{n_j+1} \|\hat{\mathbf{x}} - \mathbf{m}'_j\|^2$

- Similarly, $J'_i = J_i - \frac{n_i}{n_i-1} \|\hat{\mathbf{x}} - \mathbf{m}'_i\|^2$
- Select $\hat{\mathbf{x}}$ randomly in each step. Move it from the D_i it belongs to (nearest cluster) to a D_k where $k = \arg \min_j \frac{n_j}{n_j+1} \|\hat{\mathbf{x}} - \mathbf{m}_j\|^2$.
- This is an iterative version of the k-Means algorithm.

Hierarchical Clustering

- Clusters, subclusters, subsubclusters etc.
- At the top, all samples belong to the same class.
At the bottom, each is a separate class of its own.
- **Dendrogram:** hierarchical clustering tree showing the grouping of samples. Root of the tree represents the whole set. Intermediate nodes represent elements under each.
- Another representation: Like Venn diagram.

- **Agglomerative:** Bottom-up clustering of samples.
Divisive: Top-down clustering.

Initialize $c, k \leftarrow n, D_i$

do $k \leftarrow k - 1$

 Find nearest clusters D_i and D_j

 Merge D_i and D_j

until $k = c$

- Builds up the tree bottom-up
- What is $d(D_i, D_j)$? Several measures are possible.

Minimum Spanning Tree

- $d(D_i, D_j) = \min_{\mathbf{x}, \mathbf{y}} \|\mathbf{x} - \mathbf{y}\|, \mathbf{x} \in D_i, \mathbf{y} \in D_j$
- Join the pair of points which are closest to each other.
- Repeat till the distance between the nearest clusters exceeds a threshold. Points will be joined as a tree.
- Algorithm is called the single linkage algorithm.
- Results in the **minimum spanning tree** of the clusters.

Farthest-Neighbour

- $d(D_i, D_j) = \max_{\mathbf{x}, \mathbf{y}} \|\mathbf{x} - \mathbf{y}\|, \mathbf{x} \in D_i, \mathbf{y} \in D_j$
- Complete linkage algorithm. When two clusters are joined, add links between every pair of nodes of each.
- Each cluster is a complete subgraph or a **clique** of the graph with the samples as its nodes.
- Diameter of a partition is increased as little as possible.
- This algorithm prefers compact clusters.
- Average distance, distance of means, Hausdorff-distance, etc., can be used also.

Stepwise Optimal Hierarchical Clustering

- Generalize the notion of the nearest pair of clusters using the criterion function.
- Merge the pair that increases criterion function the least.

Initialize $c, k \leftarrow n, D_i$

do $k \leftarrow k - 1$

 Find clusters D_i, D_j whose merger changes J least

 Merge D_i and D_j

until $k = c$

- Optimal merger at every step. May not give global optimum.

Graph-Theoretic Methods

- Let $s(\mathbf{x}, \mathbf{y})$ be a measure of similarity between points \mathbf{x} and \mathbf{y} .
- Define: $s_{ij} = 1$ if $s(\mathbf{x}_i, \mathbf{x}_j) > \tau$ for some τ and 0 otherwise.
- Defines a **similarity graph** of the points.
- Connected components of this graph define clusters using single linkage algorithm.
- Maximal cliques define clusters using complete linkage algorithm.

Hierarchical Graph Clustering

- Set τ such that all points form a single (or the required c) cluster.
- Find the minimal spanning tree of this graph.
- Remove the longest edge to get 2 clusters. Remove the next longest to get 3 clusters, etc.
- Gives a top-down or divisive hierarchical clustering procedure.
- Alternately, remove an edge if it is much longer (say twice) than the average length of edges incident on a node.

PCA, NLCA, ICA

- PCA gives a procedure to describe the given data by finding the directions of maximum spread, etc.
- PCA can be computed by an auto-encoder ANN with n input nodes, k intermediate nodes, and n output nodes. Output is same as input and adjust weights.
- Non-Linear Component Analysis: Have a 5 layer auto-encoder ANN. k middle layers with linear perceptrons. The intermediate layers are non-linear.
- Independent Components: blind-source separation.