9.913 Pattern Recognition for Vision
Class VII, Part I – Techniques for Clustering
Yuri Ivanov
• Similarity metric
• K-means and IsoData algorithms
• EM algorithm
• Some hierarchical clustering schemes
Clustering

• Clustering is a process of partitioning the data into groups based on similarity

• Clusters are groups of measurements that are similar

• In Classification groups of similar data form classes
  – Labels are given
  – Similarity is deduced from labels

• In Clustering groups of similar data form clusters
  – Similarity measure is given
  – Labels are deduced from similarity
Clustering

*Labels given*

Classification

\[ y \]

\[ x \]

\[ \beta y \]

\[ \alpha x \]

Scaling

Clustering

\[ y \]

\[ x \]

\[ \beta y \]

\[ \alpha x \]

*Labels deduced*
Questions

- What is “similar”?
- What is a “good” partitioning?
Distances

Most obvious: distance between samples
- Compute distances between the samples
- Compare distances to a threshold

We need a metric to define distances and thresholds
Metric and Invariance

• We can choose it from a family:

\[ d(x, x') = \left( \sum_{k=1}^{d} \left| x_k - x'_k \right|^q \right)^{1/q} \] - Minkowski metric

\[ q = 1 \Rightarrow \text{Manhattan/city block/taxicab distance} \]
\[ q = 2 \Rightarrow \text{Euclidean distance} \]

\[ d(x, x') \text{ is invariant to rotation and translation only for } q = 2 \]
Minkovski Metrics

Points a distance 1 from origin
Other choices for invariant metric:

- We can use data-driven metric:
  \[ d(x, x') = \sqrt{(x - x')^T \Sigma^{-1} (x - x')} \] - Mahalanobis distance

- We can normalize data (whiten)
  \[ x' = \left( \Lambda^{-1/2} \Phi^T \right) x \]
  And then use the Euclidean metric
Metric

Euclidean metric
• Good for isotropic spaces
• Bad for linear transformations (except rotation and translation)

Mahalanobis metric:
• Good if there is enough data

Whitening:
• Good if the spread is due to random processes
• Bad if it is due to subclasses
Similarity

We need a symmetric function that is large for “similar” $x$

E.g.:

$$s(x, x') = \frac{x^T x'}{\|x^T\| \|x'\|}$$
- “angular” similarity

Vocabulary:

\{Two, three, little, star, monkeys, jumping, twinkle, bed\}

a) Three little monkeys jumping on the bed \((0, 1, 1, 0, 1, 1, 0, 1)\)
b) Two little monkeys jumping on the bed \((1, 0, 1, 0, 1, 1, 0, 1)\)
c) Twinkle twinkle little star \((0, 0, 1, 1, 0, 0, 2, 0)\)

Similarity matrix:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1.0</td>
<td>0.8</td>
<td>0.18</td>
</tr>
<tr>
<td>b</td>
<td>0.8</td>
<td>1.0</td>
<td>0.18</td>
</tr>
<tr>
<td>c</td>
<td>0.18</td>
<td>0.18</td>
<td>1.0</td>
</tr>
</tbody>
</table>
## Similarity

It doesn’t have to be metric:

<table>
<thead>
<tr>
<th>E.g.:</th>
<th>Has fur</th>
<th>Has 4 legs</th>
<th>Can type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monkey</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Platypus</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[
s(x, x') = \frac{x^t x'}{d}
\]

\[
s(x, x') = \frac{x^t x'}{x^t x + x'^t x' - x^t x'}
\]

*Tanimoto coefficient*
Partitioning Evaluation

$J$ – objective function, s.t. clustering is assumed optimal when $J$ is minimized or maximized

$$J = \sum_{k=1}^{K} \sum_{n=1}^{N_k} \left\| x_{n}^{(k)} - \mu_k \right\|^2$$ - Sum of squared error criterion (min)

Using the definition of the mean:

$$J = \frac{1}{2} \sum_{k=1}^{K} N_k \left[ \frac{1}{N_k^2} \sum_{n=1}^{N_k} \sum_{m=1}^{N_k} \left\| x_{n}^{(k)} - x_{m}^{(k)} \right\|^2 \right]$$

Dissimilarity measure
You can replace it with your favorite
Partitioning Evaluation

Other possibilities:

For within- and between- cluster scatter matrices (recall LDA)

\[ J = \left| S_W \right| = \left| \sum_{k=1}^{K} S_k \right| \]

- Scatter determinant criterion (min)

\[ J = tr \left| S_W^{-1} S_B \right| = \sum_{i=1}^{d} \lambda_i \]

- Scatter ratio criterion (max)

Careful with the ranks!
Which to choose?

- No methodological answer

- SSE criterion (minimum variance)
  - simple
  - good for well separated clusters in dense groups
  - affected by outliers, scale variant

- Scatter criteria
  - Invariant to general linear transformations
  - Poor on small amounts of data as related to dimensionality

- You should chose the metric and the objective that are invariant to the transformations natural to your problem
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathbf{x}$</td>
<td>input data</td>
</tr>
<tr>
<td>$K$</td>
<td>number of clusters (assumed known)</td>
</tr>
<tr>
<td>$N_k$</td>
<td>number points in cluster $k$</td>
</tr>
<tr>
<td>$N$</td>
<td>total number of data points</td>
</tr>
<tr>
<td>$t_k$</td>
<td>prototype (template) vector of $k$-th cluster</td>
</tr>
<tr>
<td>$J$</td>
<td>objective function, s.t. clustering is assumed optimal when $J$ is extremized</td>
</tr>
</tbody>
</table>
Clustering is usually an iterative procedure:

- Choose initial configuration
- Adjust configuration s.t. $J$ is optimized
- Check for convergence

$J$ is often only *partially* minimized.
Let’s choose the following model:

- **Known number of clusters**
- **Each cluster is represented by a single prototype**
- **Similarity is defined in the nearest neighbor sense**

**Sum-Squared-Error objective:**

\[
J = \sum_{k=1}^{K} \sum_{n=1}^{N_k} \left\| x_n^{(k)} - t_k \right\|^2 - \text{total in-cluster distance for all clusters}
\]

\[
\frac{dJ}{dt_k} = \sum_{c=1}^{K} \sum_{n=1}^{N_k} \frac{d}{dt_k} \left( \left\| x_n^{(k)} - t_k \right\|^2 \right) = -2 \sum_{n=1}^{N_k} (x_n^{(k)} - t_k) = 0 \Rightarrow
\]

\[
t_k = \frac{1}{N_k} \sum_{n=1}^{N_k} x_n^{(k)}
\]
K-Means Algorithm

Using the iterative procedure:

1. Choose M random positions for the prototypes
2. Classify all samples by the nearest $t_k$
3. Compute new prototype positions
4. If not converged (no cluster assignments changed from previous iteration), go to step 2

This is the *K-Means* (a.k.a. Lloyd’s, a.k.a. LBG) algorithm.

What to do with empty clusters? Some heuristics are involved.
K-Means Algorithm Example

$K = 10$
Cluster Heuristics

Sometimes clusters end up empty. We can:
- Remove them
- Randomly reinitialize them
- Split the largest ones

Sometimes we have too many clusters. We can:
- Remove the smallest ones
- Relocate the smallest ones
- Merge the smallest ones together if they are neighbors
IsoData Algorithm

In *K-Means* we assume that we know the number of clusters

IsoData tries to estimate them – ultimate *K-Means* hack

IsoData iterates between 3 stages:
- Center estimation
- Cluster splitting
- Cluster merging

The user specifies:

- \( T \) – min number of samples in a cluster
- \( N_D \) – *desired* number of clusters
- \( D_m \) – max distance for merging
- \( \sigma_S^2 \) – maximum cluster variance
- \( N_{max} \) – max number of merges
IsoData

Stage I – Cluster assignment:

1. Assign a label to each data point such that:

   \[ \omega^n = \arg \min_j \| x^n - t_j \| \]

2. Discard clusters with \( N_k < T \), reduce \( N_c \)

3. Update means of remaining clusters:

   \[ t_j = \frac{1}{N_j} \sum_{i=1}^{N_j} x_i^{(j)} \]

This is basically a step of \( K\)-Means algorithm
IsoData

Stage II – Cluster splitting:

1. If this is the last iteration, set $D_m=0$ and go to Stage III
2. If $N_c <= N_D/2$, go to splitting (step 4)
3. If iteration is even or if $N_c >= 2N_D$ go to Stage III
4. Compute:

$$d_k = \frac{1}{N_k} \sum_{i=1}^{N_k} \left\| x_i^{(k)} - t_k \right\|$$  - avg. distance from the center

$$\sigma_k^2 = \max_j \frac{1}{N_k} \sum_{i=1}^{N_k} \left( x_{i,j}^{(k)} - t_{k,j} \right)^2$$  - max variance along a single dimension

$$d = \frac{1}{N} \sum_{k=1}^{N_c} N_k d_k$$  - overall avg. distance from centers
5. For clusters with $\sigma_k^2 > \sigma_S^2$:

If $(d_k > d \text{ AND } N_k > 2(T+1)) \text{ OR } N_c < N_D / 2$

Split the cluster by creating a new mean:

$$t'_{k,j} = t_{k,j} + 0.5\sigma_k^2$$

And moving the old one to:

$$t_{k,j} = t_{k,j} - 0.5\sigma_k^2$$
IsoData

Stage III – Cluster merging:

If no split has been made:

1. Compute the matrix of distances between cluster centers

\[ D_{i,j} = \| t_i - t_j \| \]

2. Make the list of pairs where \( D_{i,j} < D_m \)

3. Sort them in ascending order

4. Merge up to \( N_{\text{max}} \) unique pairs starting from the top by removing \( t_j \) and replacing \( t_i \) with:

\[ t_i = \frac{1}{N_i + N_j} \left( N_i t_i + N_j t_j \right) \]
IsoData Example

\[ N_D = 10 \]
\[ T = 10 \]
\[ \sigma_S^2 = 3 \]
\[ D_m = 2 \]
\[ N_{max} = 3 \]
Mixture Density Model

*Mixture model* – a linear combination of parametric densities

\[
p(x) = \sum_{j=1}^{M} p(x \mid j)P(j)
\]

Number of components

Component density

Component weight

\[
P(j) \geq 0, \quad \forall j \quad \text{and} \quad \sum_{j=1}^{M} P(j) = 1
\]

Recall Kernel density estimation

Kernels are parametric densities, subject to estimation
Example

\[ p(x) = \sum_{j=1}^{M} p(x | j) P(j) \]
Mixture Density

Using ML principle, the objective function is the *log-likelihood*:

\[
l(\theta) \equiv \ln \left\{ \prod_{n=1}^{N} p(x^n) \right\} = \sum_{n=1}^{N} \ln \left\{ \sum_{j=1}^{M} p(x^n | j) P(j) \right\}
\]

Differentiate w.r.t. parameters:

\[
\nabla_{\theta_j} l(\theta) = \sum_{n=1}^{N} \frac{\partial}{\partial \theta_j} \ln \left\{ \sum_{k=1}^{M} p(x^n | k) P(k) \right\}
\]

\[
= \sum_{n=1}^{N} \frac{1}{\sum_{k=1}^{M} p(x^n | k) P(k)} \frac{\partial}{\partial \theta_j} p(x^n | j) P(j)
\]

*Here because of the log*
Mixture Density

For distributions $p(x|j)$ in the exponential family:

$$\frac{\partial}{\partial \theta} \left[ A(\theta) e^{B(\theta,x)} \right] = A(\theta) e^{B(\theta,x)} \frac{\partial}{\partial \theta} \left[ B(\theta,x) \right] + \frac{\partial}{\partial \theta} \left[ A(\theta) \right] e^{B(\theta,x)}$$

$$\Rightarrow \frac{\partial l(\theta)}{\partial \theta} = \sum_{n=1}^{N} P(j|x^n) \times (Stuff \ + More \ Stuff)$$

For a Gaussian:

$$\frac{\partial l(\theta)}{\partial \mu_j} = \sum_{n=1}^{N} P(j|x^n) \left[ \Sigma_j^{-1} (x^n - \mu_j) \right]$$

$$\frac{\partial l(\theta)}{\partial \hat{S}_j} = \sum_{n=1}^{N} P(j|x^n) \left[ \hat{S}_j^{-1} - \hat{S}_j^{-1} (x^n - \mu_j)(x^n - \mu_j)^T \hat{S}_j^{-1} \right]$$
Mixture Density

At the extremum of the objective:

\[ P(j) = \frac{1}{N} \sum_{n=1}^{N} P(j \mid x^n) \]

\[ \mu_j = \frac{\sum_{n=1}^{N} P(j \mid x^n) x^n}{\sum_{n=1}^{N} P(j \mid x^n)} \]

\[ \hat{S}_j = \frac{\sum_{n=1}^{N} P(j \mid x^n) (x^n - \mu_j)(x^n - \mu_j)^T}{\sum_{n=1}^{N} P(j \mid x^n)} \]

BUT:

\[ P(j \mid x^n) = \frac{p(x^n \mid j) P(j)}{\sum_{k=1}^{M} p(x^n \mid k) P(k)} \]

- parameters are tied

Solution – EM algorithm.
Suppose we pick an initial configuration (just like in K-Means)

Recall the objective (change of sign):

\[ E \equiv -l(\theta) = -\ln \left\{ \prod_{n=1}^{N} p(x^n) \right\} = -\sum_{n=1}^{N} \ln \left\{ p(x^n) \right\} \]

After a single step of optimization:

\[ E^{new} - E^{old} = -\sum_{n=1}^{N} \ln \left\{ \frac{p^{new}(x^n)}{p^{old}(x^n)} \right\} \]

\[ = -\sum_{n=1}^{N} \ln \left\{ \sum_{j=1}^{M} \frac{P^{new}(j)p^{new}(x^n | j)}{p^{old}(x^n)} \right\} \]
EM Algorithm

After optimization step:

\[
E^{\text{new}} - E^{\text{old}} = - \sum_{n=1}^{N} \ln \left\{ \sum_{j=1}^{M} \frac{P^{\text{new}}(j) p_{j}^{\text{new}}(x^{n} | j)}{p^{\text{old}}(x^{n})} \right\}
\]

\[
= - \sum_{n=1}^{N} \ln \left\{ \sum_{j=1}^{M} \frac{P^{\text{new}}(j) p_{j}^{\text{new}}(x^{n} | j)}{p^{\text{old}}(x^{n})} \right\}
\]

\[
= - \sum_{n=1}^{N} \ln \left\{ \sum_{j=1}^{M} P^{\text{old}}(j | x^{n}) \frac{P^{\text{new}}(j) p_{j}^{\text{new}}(x^{n} | j)}{p^{\text{old}}(x^{n}) P^{\text{old}}(j | x^{n})} \right\}
\]

\[
= - \sum_{n=1}^{N} \ln \left\{ \sum_{j=1}^{M} \frac{P^{\text{old}}(j | x^{n})}{P^{\text{old}}(j | x^{n})} \right\}
\]

\[
= - \sum_{n=1}^{N} \ln \left\{ \sum_{j=1}^{M} \lambda_{j} y_{j} \right\}
\]

Sums to 1 over \( j \)
Definition: Function $f$ is convex on $[a, b]$ iff for any $x_1, x_2$ in $[a, b]$ and any $\lambda$ in $[0, 1]$:

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2)$$
Digression - Jensen’s Inequality

If \( f \) is a convex function:

\[
f \left( \sum_{j=1}^{M} \lambda_j x_j \right) \leq \sum_{j=1}^{M} \lambda_j f(x_j)
\]

\( \forall \lambda : 0 \leq \lambda_j \leq 1, \sum_j \lambda_j = 1 \)

Equivalently:

\[
f \left( E[x] \right) \leq E \left[ f(x) \right]
\]

Or:

\[
f \left( \frac{1}{M} \sum_{j=1}^{M} x_j \right) \leq \frac{1}{M} \sum_{j=1}^{M} f(x_j)
\]

Flip the inequality if \( f \) is concave
Proof by induction:

a) JE is trivially true for any 2 points (definition of convexity)

b) Assuming it is true for any $k-1$ points:

\[
\sum_{i=1}^{k} \lambda_i f(x_i) = \lambda_k f(x_k) + (1 - \lambda_k) \sum_{i=1}^{k-1} \lambda^*_i f(x_i)
\]

\[
\geq \lambda_k f(x_k) + (1 - \lambda_k) f\left(\sum_{i=1}^{k-1} \lambda^*_i x_i\right)
\]

\[
\geq f\left(\lambda_k x_k + (1 - \lambda_k) \sum_{i=1}^{k-1} \lambda^*_i x_i\right) = f\left(\sum_{i=1}^{k} \lambda_i x_i\right)
\]

End of digression
Change in the error:

\[ E^{new} - E^{old} = \]

\[ = -\sum_{n=1}^{N} \ln \left\{ \sum_{j=1}^{M} \left[ P^{old}(j \mid x^n) \frac{P^{new}(j)p^{new}(x^n \mid j)}{p^{old}(x^n)P^{old}(j \mid x^n)} \right] \right\} \]

by Jensen’s inequality:

\[ \leq -\sum_{n=1}^{N} \sum_{j=1}^{M} P^{old}(j \mid x^n) \ln \left\{ \frac{P^{new}(j)p^{new}(x^n \mid j)}{p^{old}(x^n)P^{old}(j \mid x^n)} \right\} \sum_{j=1}^{M} \lambda_j \ln \{ y_j \} \]
Change in the error:

\[ E^{new} - E^{old} = \]

\[ = -\sum_{n=1}^{N} \ln \left\{ \sum_{j=1}^{M} \left[ P^{old} (j \mid x^n) \frac{P^{new} (j) p^{new} (x^n \mid j)}{p^{old} (x^n) P^{old} (j \mid x^n)} \right] \right\} \]

by Jensen’s inequality:

\[ \leq -\sum_{n=1}^{N} \sum_{j=1}^{M} P^{old} (j \mid x^n) \ln \left\{ \frac{P^{new} (j) p^{new} (x^n \mid j)}{p^{old} (x^n) P^{old} (j \mid x^n)} \right\} \]

call this “Q”
EM as Upper Bound Minimization

Then: \[ E^{new} \leq E^{old} + Q \] - upper bound on \( E^{new}(\theta^{new}) \)

Some observations:
• \( Q \) is convex
• \( Q \) is a function of new parameters \( \theta^{new} \)
• So is \( E^{new} \)
• If \( \theta^{new} = \theta^{old} \) then \( E^{new} = E^{old} + Q \)

Step downhill in \( Q \) leads downhill in \( E^{new} \) !!!

\[ E(\theta^{new}) \]

\[ E^{old} \]

\[ E^{old} + Q(\theta^{new}) \]

\[ E^{new}(\theta^{new}) \]

\( \theta^{old} \)

\( \theta^{new} \)
EM Iteration

Given initial $\theta$ minimize $Q$

Compute new $E^{old} + Q$

Fall 2004 Pattern Recognition for Vision
EM (cont.)

\[ Q = -\sum_{n=1}^{N} \sum_{j=1}^{M} P^{old}(j \mid x^n) \ln \left\{ \frac{P^{new}(j)p^{new}(x^n \mid j)}{p^{old}(x^n)P^{old}(j \mid x^n)} \right\} \]

Can drop these

\[ \tilde{Q} = -\sum_{n=1}^{N} \sum_{j=1}^{M} P^{old}(j \mid x^n) \ln \left\{ P^{new}(j)p^{new}(x^n \mid j) \right\} \]

for a Gaussian mixture:

\[ = -\sum_{n=1}^{N} \sum_{j=1}^{M} P^{old}(j \mid x^n) \left\{ \ln P^{new}(j) - \ln \left(G_j(x^n)\right) \right\} \]

As before – differentiate, set to 0, solve for parameter.
EM (cont.)

Straight-forward for means and covariances:

\[
\hat{\mu}_j = \frac{\sum_{n=1}^{N} P_{old}^j (j | x^n) x^n}{\sum_{n=1}^{N} P_{old}^j (j | x^n)}
\]
- convex sum, weighted w.r.t. previous estimate

\[
\hat{S}_j = \frac{\sum_{n=1}^{N} P_{old}^j (j | x^n) (x^n - \hat{\mu}_j)(x^n - \hat{\mu}_j)^T}{\sum_{n=1}^{N} P_{old}^j (j | x^n)}
\]
- convex sum, weighted w.r.t. previous estimate
EM (cont.)

Need to enforce sum-to-one constraint for $P(j)$:

$$J_P = \tilde{Q} + \lambda \left( \sum_{j=1}^{M} P_{new}^{j} - 1 \right)$$

$$\frac{\partial}{\partial P_{new}^{j}} J_P = - \sum_{n=1}^{N} \frac{P_{old}^{j|X^n}}{P_{new}^{j}} + \lambda = 0$$

$$\Rightarrow \lambda P_{new}^{j} = \sum_{n=1}^{N} P_{old}^{j|X^n}$$

$$\Rightarrow \lambda \sum_{j=1}^{M} P_{new}^{j} = \sum_{n=1}^{N} \sum_{j=1}^{M} P_{old}^{j|X^n}$$

$$\Rightarrow \lambda = N \Rightarrow P_{new}^{j} = \frac{1}{N} \sum_{n=1}^{N} P_{old}^{j|X^n}$$
EM Example

\[ N_c = 3 \]
You can manipulate $P(j|x)$. Eg: Partially labeled data

$P(j|x)$ tells how much the data point affects each cluster, unlike in K-means.
Furthermore, $P(j|x)$ can be replaced with:

$$
\tilde{P}(j|x) = \frac{P(j|x)e^{\gamma P(j|x)}}{\sum_k P(k|x)e^{\gamma P(k|x)}} \bigg|_{\gamma=0}
$$

if $\gamma = 0$, $\tilde{P}(j|x) = P(j|x)$

Now let’s relax $\gamma$:

$$
\lim_{\gamma \to \infty} \tilde{P}(j|x) = \delta(P(j|x), \max P(j|x))
$$

This is K-Means!!!
Hierarchical Clustering

Ex: Dendrogram

There are 2 ways to do it:
• Agglomerative (bottom-up)
• Divisive (top-down)

Different thresholds induce different cluster configurations.

Stopping criterion – either a number of clusters, or a distance threshold
Hierarchical Agglomerative Clustering

**General structure:**

**Initialize:** $K, \hat{K} \leftarrow N, D_n \leftarrow x_n, n = 1..N$

```
do
  \hat{K} \leftarrow \hat{K} - 1
  i, j = \text{argmin}_{l, m} d(D_l, D_m)
  \text{merge}(D_i, D_j)
until \hat{K} = K
```

**Ex:**

\[
\begin{align*}
  d &= d_{\text{mean}}(D_i, D_j) = \| \mu_i - \mu_j \| \\
  d &= d_{\text{min}}(D_i, D_j) = \min_{x_1 \in D_i, x_2 \in D_j} \| x_1 - x_2 \| \\
  d &= d_{\text{max}}(D_i, D_j) = \max_{x_1 \in D_i, x_2 \in D_j} \| x_1 - x_2 \|
\end{align*}
\]

Each induces different algorithm

Need to specify

Fall 2004 Pattern Recognition for Vision
Choosing $d = d_{\text{min}}$ results in a Nearest Neighbor Algorithm (a.k.a. single linkage algorithm, a.k.a. minimum algorithm)

Each cluster is a minimal spanning tree of the data in the cluster.

Identifies clusters that are well separated
Complete Linkage Algorithm

Choosing $d = d_{max}$ results in a Farthest Neighbor Algorithm (a.k.a. complete linkage algorithm, a.k.a. maximum algorithm)

Each cluster is a complete subgraph of the data.

Identifies clusters that are well localized

$N = 2$
Summary

- General concerns about choice of similarity metric
- K-means algorithm – simple but relies on Euclidean distances
- IsoData – old-school step towards model selection
- EM – “statistician’s K-means” – simple, general and convenient
- Some hierarchical clustering schemes