**Pattern Recognition & Classification**

**Classification**

- Supervised
  - parallelepiped
  - minimum distance
  - maximum likelihood (Bayes Rule)
    - non-parametric
    - parametric
- **Unsupervised (clustering)**
  - K-Means
  - ISODATA

---

**Unsupervised Classification**

A classification procedure is *unsupervised* if

- no training data are required
- the user only needs to specify information that does not describe individual class characteristics:
  - number of classes
  - a threshold distance
  - …
Unsupervised Classification: Clustering

Clustering: Pattern Classification by Distance Functions

Premise: pixels which are close to each other in feature space are likely to belong to the same class.
- The "distance" between pixels in feature space (n-D histogram) is the measure of similarity.
- All dimensions should be in comparable units.
- Distance may be scaled in pixels, radiance, reflectance, ….
- Most effective if the clusters are disjoint.
- Requires the least amount of prior information to operate.

Clustering

Distance in feature space is the primary measure of similarity in all clustering algorithms.

Figure: Two "patterns" in a two-dimensional measurement space. The patterns are identifiable because the points group or cluster in this measurement space.

Pixels that are "close" in feature space will be grouped in the same class.

a) The relative distances may change when data are
  – calibrated (digital counts \( \rightarrow \) radiance)
  – atmospherically corrected
  – rescaled in ways that treat different features differently..

b) If two features have different units, they must be scaled to provide comparable variance. Otherwise the "distance" will be biased.
**Minimum distance classification**

**Single prototypes:** Each class (pattern) is represented by a single prototype vector, $z$.

- Each class is modeled as a circular (or spherical) distribution in measurement (or feature) space.
- A circular pattern suggests that
  - there is no correlation among the measurements $(x_1, x_2, \ldots)$, and
  - the variance within each class is the same for all features.

---

**Minimum distance classification**

**Single prototypes:** Each class (pattern) is represented by a single prototype vector, $z$.

- Assume that there are $m$ classes and that these classes are represented by the prototype vectors, $z_1, z_2, z_3, \ldots, z_m$.
- The Euclidean distance, $D_i(x)$, of a measurement vector, $x$, from the prototype vector, $z_i$:
  \[
  D_i(x) = |x - z_i| = [(x - z_i)'(x - z_i)]^{1/2}
  \]

The discriminant function is usually defined as the negative of the separation distance: $d_i(x) = -D_i(x)$

- The larger (less negative) $d_i(x)$, the closer the measurement vector $x$ lies relative to the prototype vector $z_i$.
- The maximum value of $d_i(x)$ is zero and occurs when $x$ matches the prototype vector exactly.
**Alternate definitions of the discriminant function**

*Distance* in feature space is the primary measure of similarity in all clustering algorithms.

1) Euclidean distance:
\[ d_i(x) = -D_i(x) = -[(x-z_i)'(x-z_i)]^{1/2} \]

2) Square of the Euclidean distance:
\[ d_i(x) = -D_i^2(x) = -[(x-z_i)'(x-z_i)] \]

3) Square of Euclidean distance after eliminating components that are independent of class:
\[ d_i(x) = -[x'z_i + \frac{1}{2}(z_i'z_i)] \]

4) Taxicab distance
\[ d_i(x) = -D_i(x) = -\sum_{k=1}^{N_h} |x_k - z_{ki}| \]

---

**Clustering: single prototype decision boundary**

The *decision boundary* for the single prototype, simple distance discriminant function is the set of planar surfaces perpendicular to and bisecting the lines connecting pairs of prototypes.

This is a *minimum-distance* classifier.

If the prototype is the mean value of the training pixels for each class, it is called a *minimum-distance-to-mean* classifier.
**Cluster seeking**

The results of clustering will depend strongly on the choice of the prototype.

Alternatives for prototype selection:
- let the user select prototypes, i.e., one "example" pixel per class. (Reduces the utility of a clustering procedure.)
- devise an unbiased procedure for selecting prototypes (random selection, selection at vertices of an arbitrary grid, ….)
- Use the user-selected prototype or unbiased selection procedure as the starting point of an optimization procedure.

We will consider techniques which begin with an arbitrary starting point in measurement space and explore the data distribution to locate optimal cluster prototypes.

---

**A simple cluster seeking algorithm**

1. Select a threshold, T
   T is a representative distance in measurement space. The choice of T in this algorithm is entirely arbitrary; it is also the only input required of the user.

2. Select a pixel with measurement vector, x.
   The selection scheme is arbitrary. Pixels could be selected at random, however, for this example assume that the first pixel is selected from the upper left corner of the image, and that subsequent pixels are selected in order from left to right and from top to bottom of the image.

3. Let the first pixel be taken as the first cluster center, z₁.
A simple cluster seeking algorithm

4. Select the next pixel from the image.

5. Compute the distance functions, \( D_i(x) \).
   Compute the distance function for each of the classes established at this point, i.e.,
   compute \( D_i(x) \), for \( i=1,N \)
   where \( N = \) the number of classes. (\( N=1 \) initially.)

6. Compare the \( D_i(x) \) with \( T \).
   a) if \( D_i(x) < T \), then \( x \in \omega_i \).
   b) if \( D_i(x) > T \), for all \( i \), then let \( x \) become a new prototype vector: Assign \( x \rightarrow z_{N+1} \). (Do not compute \( D_N \) for pixels already assigned to an existing class.

8. Return to step #4 until all pixels are assigned to a class

9. After all pixels have been assigned to a cluster center, recompute the \( D_i(x) \) and reassign pixels according to the minimum \( D_i(x) \)

This simple clustering algorithm is extremely sensitive to the value of the threshold, \( T \), and the order in which pixels are selected.

- for a given \( T \), two different selection patterns can yield very different results.
- for the same selection pattern, a different value of \( T \) will lead to a different results.

These flaws are typical of clustering algorithms. All are sensitive to the starting selection of cluster centers and to the particular specification of the clustering criterion. The better algorithms handle the problems cleverly and without the severe problems that would be apparent with the above algorithm.
A simple cluster seeking algorithm

$z_1$ is the initial prototype.
$c_1, c_2, c_3, c_4$ represent actual clusters
$T$ is the threshold distance for inclusion in a cluster
$\omega_1, \omega_2, \omega_3, \omega_4$ represent assigned clusters
Maximin distance algorithm

The *maximin* (maximum-minimum) algorithm represents an attempt to define a less arbitrary and more repeatable distribution of cluster groups.

1) begin by identifying cluster regions that are farthest apart
2) define an initial threshold distance based on the separation of these cluster centers, and
3) continue selecting cluster centers and readjusting T until all possible cluster centers are established.

Note that, with the maximin algorithm, **NO** information is required from the user.

Maximin distance algorithm

1. Select a pixel, $x$, from the image at random.
2. Let $x$ be the first cluster center, $x \rightarrow z_1$
3. Sort through the remaining pixels to find the pixel, $x$, which is farthest from $z_1$.
4. Let the most distant pixel be the second cluster center, $x \rightarrow z_2$.
5. Find the distance, $T = |z_2-z_1|$, between the two cluster centers.
   
   $T$ will be an initial scaling distance used to determine the existence of the next cluster center.
6. Compute $D_{\text{min}}(x_j) = \min[D_i(x_j)]$ for $i=1,2$, for all remaining pixels in the image, i.e., find the distance to the closest cluster center for every pixel in the image.
**Maximin distance algorithm**

7. Find $D_{\text{max}}(x_m) = \max[D_{\text{min}}(x_j)]$ for all $j$.

Sort through all the distances determined in step 6 and select the maximum distance (select the maximum of the minimum distances). This procedure will find the pixel that is farthest (in measurement space) from either of the two cluster centers.

8. If $D_{\text{max}} > T/2$, then let $x_m \Rightarrow z_{n+1}$, otherwise,
   if $D_{\text{max}} < T/2$, then terminate the procedure.

   In words, if the maximum distance is greater than half the distance between the two closest cluster centers, then $x_m$ becomes a new cluster center, otherwise terminate the procedure.

9. Increment the number of cluster centers by 1: $N_c = N_c + 1$.

10. Reset the scaling distance: $T = \sum_{i=1}^{n} \sum_{j=1}^{n} |z_i - z_j| / \sum_{k=1}^{n} k(k+1)/2$


---

**Maximin distance algorithm: examples**

- Cluster centers are labeled according to the order of selection:
- Clearly separable, circular clusters of about the same size can often be located by the maximin algorithm, but
- More realistic distributions may be improperly partitioned.
**Clustering: Computing efficiency**

- Because of the large volume of data and the need to repeat complex computations many times, it is crucial to streamline computations as much as possible. Clever, efficient programming is essential.

- The maximin algorithm, for example, is computationally demanding.
  - Each time a new cluster center is selected, the feature space distance must be computed for every point from every cluster center.
  - On the $i^{th}$ pass through an $m \times n$ pixel image there will be $i+1$ clusters requiring $m \times n \times i$ distance computations.

**Clustering: K-Means**

**K-means Algorithm - adaptive cluster centers**

- In the previous clustering examples, once a point has been selected as a clustering center, it remains a clustering center, even if it is a relatively poor representative of its cluster.

- The K-means algorithm allows the cluster centers to shift in order to optimize a performance index.

- Many variations of the K-means algorithm have been developed, but the steps of a basic procedure will be shown here.
**Clustering: K-Means**

1. **Choose K initial cluster centers, z₁, z₂, z₃, ..., zₖ.**
   - Requires
     - the number of desired clusters and
     - an initial list of clusters centers. These are typically selected by the algorithm (often by an unspecified method).
   - If selected by the routine cluster centers may be chosen in a variety of ways:
     a) random or structured selection from the image data,
     b) applying some a priori information (e.g., training data),
     c) selecting points based on preliminary calculations (minimum/maximum gray values, variance in each band, localized data density, etc.), or
     d) applying a theoretical principal independent of actual data.

2. **Distribute the samples among the K means**
   Samples are assigned to the class represented by the nearest cluster center, i.e.:
   \[
   x \in S_i(n) \quad \text{if} \quad |x - z_i(n)| \leq |x - z_j(n)|
   \]
   for all \( j = 1, 2, 3, \ldots, k \); where \( i \neq j \)

   \( S_i(n) \) is the set of samples whose cluster center is \( z_i(n) \), where \( n \) indicates that this is the \( n^{th} \) iteration of this procedure.

3. **Compute new cluster centers for each set \( S_i(n) \)**
   Find a new value for each \( z_i \). The new cluster center, \( z_i(n+1) \) will be the mean of all the points in \( S_i(n) \) such that:
   \[
   z_i(n+1) = \frac{\sum_{x \in S_i(n)} x}{|S_i(n)|}
   \]
Clustering: K-Means

4. Compare \( z_i(n) \) and \( z_i(n+1) \) for all \( i \)

Compute the distance between each pair of points for the consecutive iterations.

Then:

a) If there is no substantial change, terminate the procedure. Some possible criteria for termination
   i) if \(| z_i(n+1) - z_i(n) | < T \) for all \( i \)
   ii) if \( \sum_{j=1}^{k} | z_i(n+1) - z_i(n) | < T \) for all \( i \)

b) otherwise return to step 2. for the next iteration.
**Clustering: K-Means**

class 1 mean: (4.70, 4.11)
class 2 mean: (8.97, 2.30)
class 3 mean: (11.40, 5.91)

cluster 1 prototype: (3.6, 3.1) \(\Rightarrow\) (3.7, 3.2)
cluster 2 prototype: (9.6, 2.7) \(\Rightarrow\) (9.3, 2.6)
cluster 3 prototype: (6.2, 5.7) \(\Rightarrow\) (6.9, 5.8)
cluster 4 prototype: (11.5, 6.1) \(\Rightarrow\) (11.6, 6.0)

**Enter**

- **K** = number of cluster centers desired.
- \(\varepsilon_t\) = Enter a change threshold (0-100%) which is used to end the iterative process "when the number of pixels in each class changes by less than the threshold".
- **maxiter** = maximum # iteration allowed and optionally
- **maxstddev** = standard deviation to use around the class mean
- **maxdist** maximum allowable distance error (in DN).

The classification uses the smaller of the two optional values to determine which pixels will be classified. If neither parameter is entered, then all pixels will be classified.
Clustering: ISODATA

ISODATA: Iterative Self-Organizing Data Analysis Technique A

The ISODATA algorithm is essentially a refinement of the K-Means algorithm. The specific refinements are:

1) Clusters that have too few members are discarded.
2) Clusters that have too many members are split into two new cluster groups.
3) Clusters that are too large (too disperse) are split into two new cluster groups.
4) If two cluster centers are too close together they are merged.
Clustering: ISODATA

1. Select an initial set of clustering centers, $z_1, z_2, \ldots, z_{N_c}$
   - All algorithms require the user to specify the number of initial cluster centers.
   - Some algorithms allow the user to specify the specific locations of the cluster centers or to choose the process by which the cluster centers are selected; *(ENVI: no)*
   - Most simply assign the cluster centers (often with an undocumented procedure).
     *(ENVI: assigns prototypes "evenly distributed in the data space")*
   - Clustering algorithms tend to be sensitive to the number and location of the initial cluster centers. Selecting a "good" set of seed values, can not only speed up the clustering process (faster convergence) but can also lead to "better" results.

2. Specify process parameters
   - $K = \text{number of cluster centers desired.}$
     The actual number of clusters returned at the end of clustering will be within a range bounding the number specified by the user.
     *(ENVI: specify a minimum and maximum.)*
   - $N_{\text{min}} = \text{minimum number of samples allowed for a viable cluster.}$
   - $\sigma_{\text{lim}} = \text{maximum allowable size of a single cluster.}$
   - $D_{\text{lump}} = \text{lumping parameter -- a distance in gray values.}$
     If two cluster centers are closer than the lumping parameter distance, the clusters will be grouped into a single cluster.
   - $N_{\text{lump}} = \text{maximum number of pairs of cluster centers which can be lumped during one iteration.}$
   - $N_{\text{iter}} = \text{number of iterations allowed.}$
     If the procedure is going to work given the initial parameters, it should converge fairly quickly, i.e., in a few iterations. If the routine does not converge "quickly" it could go on indefinitely.
   - $\varepsilon = \text{stopping criterion}$
     Stop if the total change in position of the cluster centers falls below this limit.
**Clustering: ISODATA**

3. Distribute samples among the cluster centers

\[ x \in \omega_j \text{ iff } |x - z_j| \leq |x - z_i| \]

where: \( S_j \) = the set of samples assigned to cluster center \( z_j \)

\( i, j = 1, 2, 3, \ldots, N_c; \quad i \neq j \)

\( N_c = \) number of cluster centers

4. Discard sample subsets with fewer than \( N_{\text{min}} \) members

If \( S_j \) contains \( N_j \) members and \( N_j < N_{\text{min}} \), discard \( z_j \) and reduce \( N_c \) by 1. The members of the discarded class may be dealt with in a number of ways, peculiar to the specific algorithm. They may be:

a. incorporated into other clusters (ENVI)

b. ignored for this iteration

c. assigned to an unclassified group and ignored for the rest of the procedure.

5. Compute new cluster centers for each set \( S_i \)

\[ z_j = \frac{1}{N_j} \sum_{x \in S_j} x, \quad j = 1, 2, 3, \ldots, N_c \]

6. Compute the cluster size parameter, \( D_j \)

\[ D_j = \frac{1}{N_j} \sum_{x \in S_j} |x - z_j|, \quad j = 1, 2, 3, \ldots, N_c \]

7. Compute the overall size, \( D_s \)

\( D_s \) is weighted average of distances of samples from their respective cluster centers:

\[ D_s = \frac{1}{N} \sum_{j=1}^{N_c} N_j D_j \]

where \( N \) is the total number of pixels:

\[ N = \sum_{j=1}^{N_c} N_j \]
a) on the last iteration, set \( D_{\text{lump}} = 0 \) and go to step 12,
b) if \( N_c < K/2 \) (or \( K_{\text{min}} \)), go to step 9,
c) on even iterations or if \( N_c > 2K \) (or \( K_{\text{max}} \)), go to step 12,
d) otherwise go to step 9.

10. Find the maximum component of each \( \sigma_j \)

\[ \sigma_{j,\text{max}} = \max [\sigma_{ij}] \]

This will be the primary measure of the size of the cluster.

11. Test for splitting

If \( \sigma_{\text{max}} > \sigma_{\text{lim}} \)

AND EITHER \( D_j > D_s \) and \( n_j < 2(k+1) \) (or \( n_j < K_{\text{max}} \))

OR \( N_c < K/2 \) (if there are too few clusters)

THEN split the cluster, \( S_j \), into two new clusters.

Let \( z_1 = (z_{2}, z_{3}, \ldots, z_{j} + k_{\text{j, max}}, \ldots, z_{n}) \)
\[ z_{N_c+1} = (z_{2}, z_{3}, \ldots, z_{j} - k_{\text{j, max}}, \ldots, z_{n}) \]
\[ N_c = N_c + 1 \]

If splitting occurs then go to step 2; otherwise continue.

12. Compute the pairwise distances, \( D(i,j) \) between all cluster centers

\[ D(i,j) = |z_i - z_j|, \quad i = 2, \ldots, N_c \]
\[ j = 1, 2, \ldots, i-1 \]

13. Sort by length and find the \( N_{\text{lump}} \) smallest distances, such that \( D(i,j) < D_{\text{lump}} \)

Find the set of \( D(k(i,j)) < D_{\text{lump}} \),
where: \( k = 1, 2, 3, \ldots, N_{\text{lump}} \)
and \( D(i,j) < D_{\text{lump}} \)

14. Lump clusters

\( D_k(i,j) \) is the distance between cluster centers at \( z_i \) and \( z_j \). If neither \( z_i \) nor \( z_j \) has been used for lumping during this iteration, then merge the two clusters:

\[ z_{\text{lump}} = \frac{1}{N_i + N_j} \left[ N_i z_i - N_j z_j \right] \]

delete \( z_i \) and \( z_j \) and decrement \( N_c \):
\[ N_c = N_c - 1 \]

Note: Only pairwise lumping is allowed. If any \( z_i \) appears in more than one pair, only the first pair will be merged.
**Clustering: ISODATA**

11. **Test for splitting**

IF \( \sigma_{j,\text{max}} > \sigma_{\text{lim}} \)

(if the std. dev. for any cluster exceeds the limiting value)

AND EITHER \( D_j > D_s \) and \( n_j < 2(K+1) \) (or \( n_j < K_{\text{max}} \))

(if the cluster is more disperse than the average and if splitting will not result in exceeding the upper limit of clusters.

OR \( N_c < K/2 \) (if there are too few clusters)

THEN split the cluster, \( S_j \), into two new clusters.

Let \( z_{N_c} = (z_1, z_2, \ldots, z_j + ks_{j,\text{max}}, \ldots, z_n) \)

\( z_{N_c} = (z_1, z_2, \ldots, z_j - ks_{j,\text{max}}, \ldots, z_n) \)

\( N_c = N_c + 1 \)

If splitting occurs then go to step 2; otherwise continue.

---

12. **Compute the pairwise distances, \( D(i,j) \) between all cluster centers**

\[
D(i,j) = |z_i - z_j|, \quad i = 2, \ldots, N_c
\]

\[
j = 1, 2, \ldots, i-1
\]

13. **Sort by length and find the \( N_{\text{lump}} \) smallest distances, such that \( D(i,j) < D_{\text{lump}} \)**

Find the set of \( D_k(i,j) < D_{\text{lump}} \), where: \( k = 1, 2, 3, \ldots, N_{\text{lump}} \)

and \( D_k(i,j) < D_{k+1}(i,j) \)
Clustering: ISODATA

14. Lump clusters

$D_k(i,j)$ is the distance between cluster centers at $z_i$ and $z_j$. If neither $z_i$ nor $z_j$ has been used for lumping during this iteration, then merge the two clusters:

$$z_{\text{lump}} = \frac{1}{N_i + N_j} \left( z_i N_i - z_j N_j \right)$$

delete $z_{il}$ and $z_{jl}$ and decrement $N_c$: $N_c = N_c - 1$

Note: Only pairwise lumping is allowed. If any $z_i$ appears in more than one pair, only the first pair will be merged.

15. Continue or terminate

IF $N_{\text{iter}}$ iterations have already been performed.

OR IF on the kth iteration: $N_c(k) = N_c(k-1)$ and $|z_i(k) - z_i(k-1)| < \varepsilon$

THEN TERMINATE

ELSE CONTINUE
**Clustering: ISODATA**

2. **Specify process parameters**

   K = number of cluster centers desired.
   The actual number of clusters returned at the end of clustering will be between K/2 and 2K.

   \[ N_{\text{min}} \] = minimum number of samples allowed for a viable cluster.
   \[ \sigma_{\text{lim}} \] = maximum allowable size of a single cluster.

   \[ D_{\text{lump}} \] = lumping parameter -- a distance in gray values.
   If two cluster centers are closer than the lumping parameter distance, the clusters will be grouped into a single cluster.

   \[ N_{\text{lump}} \] = maximum number of pairs of cluster centers which can be lumped during one iteration.

   \[ N_{\text{iter}} \] = number of iterations allowed.
   If the procedure is going to work given the initial parameters, it should converge fairly quickly, i.e., in a few iterations. If the routine does not converge "quickly" it could go on indefinitely.

---

**K = number of cluster centers desired.**

- In some implementations, the actual number of clusters returned at the end of clustering will be between K/2 and 2K.
- ENVI allows the user to specify the acceptable range explicitly.
\( \text{N}_{\text{iter}} = \text{number of iterations allowed.} \)

- Should be large enough to allow convergence if the parameters are appropriate for the data set.
- More difficult discriminations may take longer to converge.

If the change in the prototype vectors is smaller than this value, the computations will stop.

- If the number is too small, convergence will be very slow. If it is very small the convergence criteria may never be met.
- If the number is too large, convergence will be quick, but probably not very useful.
\[ N_{\text{min}} = \text{minimum number of samples allowed for a viable cluster.} \]
$$\sigma_{lim} = \text{maximum allowable size of a single cluster.}$$

$$D_j = \frac{1}{N_j} \sum_{x \in S_j} |x - z_j|, \quad j = 1, 2, 3, \ldots, N_c$$
**D_{lump}** = lumping parameter -- a distance in gray values. If two cluster centers are closer than the lumping parameter distance, the clusters will be grouped into a single cluster.

**N_{lump}** = maximum number of pairs of cluster centers which can be lumped during one iteration.

**Clumping pairs sorted by distance**

<table>
<thead>
<tr>
<th>Z_3</th>
<th>Z_4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z_6</td>
<td>Z_7</td>
</tr>
<tr>
<td>Z_4</td>
<td>Z_5</td>
</tr>
</tbody>
</table>

**Output Result to:** File, Memory

**Enter Output Filename:** Choose

**ISODATA Parameters**

- Number of Classes: Max
- Maximum Iteration: 10
- Change Threshold (%): 0.05
- Maximum Class StdDev: 1.000
- Minimum Class StdDev: 0
- Minimum Class Distance: 5.000
- Maximum Class Weight: 2

**ENVI 3.6**
Criteria for allowing unclassified pixels