

Unsupervised Classification (Clustering)

Classification

- Supervised
- **Unsupervised (clustering)**
 - K-Means
 - ISODATA

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
 - ...

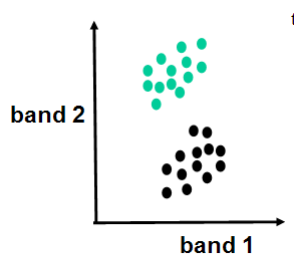
Unsupervised Classification (Clustering)

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 is the measure of similarity.
- Distance may be scaled in pixels, radiance, reflectance,
- Most effective if the clusters are disjoint.
- Requires the least amount of prior information to operate.

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



Two "patterns" in a two-dimensional measurement space are illustrated in the figure. 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.

- The relative distances may change when data are calibrated (digital counts ==> radiance) or atmospherically corrected or rescaled in ways that treat different spectral bands differently..
- If two features have two different units, they must be scaled to provide comparable variance. Otherwise the "distance" will be biased toward the feature with the smallest absolute range.

Minimum distance classification

Single prototypes: Each class (pattern) is represented by a single prototype vector, \mathbf{z} .

- Each class is modeled as a circular (or spherical) distribution in measurement space.
- A circular pattern suggests that there is no correlation among the measurements (x_1, x_2, \dots) , and that the variation within the class is the same for all measurements.
- Assume that there are m classes and that these classes are represented by the prototype vectors, $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \dots, \mathbf{z}_m$.
- A discriminant function for the i^{th} class may be derived directly from the expression for the **Euclidean distance**, $D_i(\mathbf{x})$, of a measurement vector, \mathbf{x} , from the prototype vector, \mathbf{z}_i :

$$D_i(\mathbf{x}) = |\mathbf{x} - \mathbf{z}_i| = [(\mathbf{x} - \mathbf{z}_i)'(\mathbf{x} - \mathbf{z}_i)]^{1/2}$$

where: $(\mathbf{x} - \mathbf{z}_i)$ denotes a column vector

$(\mathbf{x} - \mathbf{z}_i)'$ denotes the transpose, a row vector

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

- The larger (less negative) $d_i(\mathbf{x})$, the closer the measurement vector \mathbf{x} lies relative to the prototype vector \mathbf{z}_i .
- The maximum value of $d_i(\mathbf{x})$ is zero and occurs when \mathbf{x} matches the prototype vector exactly.

Several computationally convenient modifications can be made without altering the effect of $d_i(\mathbf{x})$.

- 1) Using the Euclidean distance, $d_i(\mathbf{x}) = -[(\mathbf{x} - \mathbf{z}_i)'(\mathbf{x} - \mathbf{z}_i)]^{1/2}$ is not computationally efficient.

$$d_i(\mathbf{x}) = -D_i(\mathbf{x}) = -|\mathbf{x} - \mathbf{z}_i| = -[(\mathbf{x} - \mathbf{z}_i)'(\mathbf{x} - \mathbf{z}_i)]^{1/2}$$

- 2) Use the square of the distance, thus eliminating the need for computing the square root.

- The magnitude of the discriminant function changes, but the value of the discriminant function is equal for any two points that are equidistant from the prototype.

$$d_i(\mathbf{x}) = -D_i(\mathbf{x}) = -|\mathbf{x} - \mathbf{z}_i| = -[(\mathbf{x} - \mathbf{z}_i)'(\mathbf{x} - \mathbf{z}_i)]$$

- 3) The computation may be further simplified by expanding the vector product and recognizing that not all the terms are class dependent:

$$d_i(\mathbf{x}) = -(\mathbf{x} - \mathbf{z}_i)'(\mathbf{x} - \mathbf{z}_i) = -[\mathbf{x}'\mathbf{x} - 2\mathbf{x}'\mathbf{z}_i + \mathbf{z}_i'\mathbf{z}_i]$$

- The term, $\mathbf{x}'\mathbf{x}$, is independent of class. This term does not alter the assignment of a measurement vector to a particular class.
- The term, $\mathbf{z}_i'\mathbf{z}_i$, is dependent on the class but not on the measurement vector, i.e., it is constant for a particular class and need only be computed once for each class
- The crucial term is the product, $\mathbf{x}'\mathbf{z}_i$

- 4) Another option is to simplify the distance computation itself. For example, instead of computing the Euclidean distance one might compute the crude, but simple "taxicab" distance,

$$D_i(\mathbf{x}) = \sum_{k=1}^{N_b} |x_k - z_{ki}|$$

where $N_b = \#$ of variables (spectral bands)

The taxicab distance is computed as a simple sum, but overestimates diagonal distances.

Summary of Distance measures:

- 1) Euclidean distance: $d_i(\mathbf{x}) = -D_i(\mathbf{x}) = -[(\mathbf{x} - \mathbf{z}_i)'(\mathbf{x} - \mathbf{z}_i)]^{1/2}$
- 2) Square of the Euclidean distance: $d_i(\mathbf{x}) = -D_i^2(\mathbf{x}) = -[(\mathbf{x} - \mathbf{z}_i)'(\mathbf{x} - \mathbf{z}_i)]$
- 3) Square of Euclidean distance after eliminating components that are independent of class:
 $d_i(\mathbf{x}) = -[\mathbf{x}'\mathbf{z}_i + \frac{1}{2}(\mathbf{z}_i'\mathbf{z}_i)]$
- 4) taxicab distance: $d_i(\mathbf{x}) = -D_i(\mathbf{x}) = -\sum_{k=1}^{N_b} |x_k - z_{ki}|$

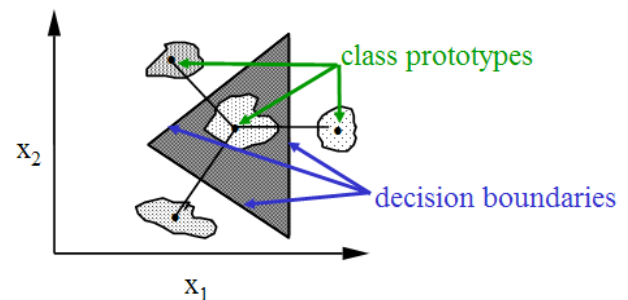
Clustering: single prototype decision boundary

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.

- 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.

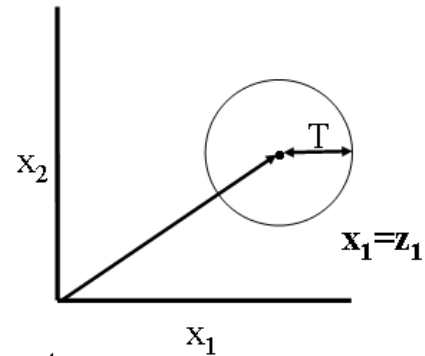
A simple cluster seeking algorithm

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, \mathbf{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, \mathbf{z}_1 .

4. Select the next pixel from the image.

5. Compute the distance functions, $D_i(\mathbf{x})$.

Compute the distance function for each of the classes established at this point, i.e.,

compute $D_i(\mathbf{x})$, for $i=1, N$

where N = the number of classes. ($N=1$ initially.)

6. Compare the $D_i(\mathbf{x})$ with T .

- if $D_i(\mathbf{x}) < T$, then $\mathbf{x} \in \omega_i$.
- if $D_i(\mathbf{x}) > T$, for all i , then let \mathbf{x} become a new prototype vector: Assign $\mathbf{x} \rightarrow \mathbf{z}_{N+1}$.
(Do not compute D_{N+1} for pixels already assigned to an existing class.)

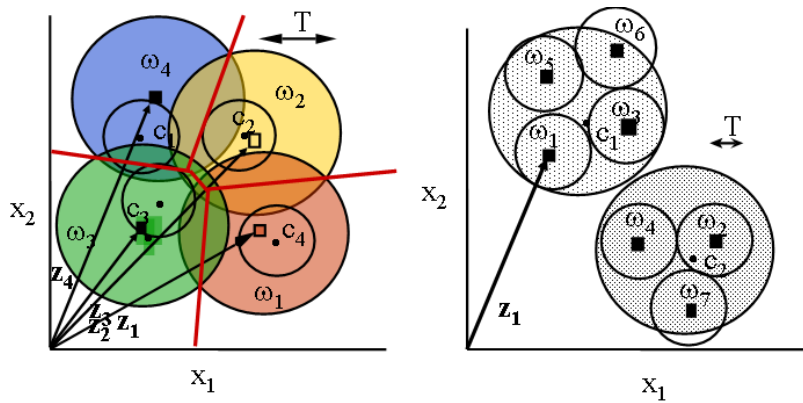
8. Return to step #4 until all pixels are assigned to a class

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

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.



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

Procedure:

1. Select a pixel, \mathbf{x} , from the image at random.
2. Let \mathbf{x} be the first cluster center, $\mathbf{x} \Rightarrow \mathbf{z}_1$
3. Sort through the remaining pixels to find the pixel, \mathbf{x} , which is farthest from \mathbf{z}_1 .
4. Let the most distant pixel be the second cluster center, $\mathbf{x} \Rightarrow \mathbf{z}_2$.
5. Find the distance, $T = |\mathbf{z}_2 - \mathbf{z}_1|$, between the two cluster centers.

T will be an initial scaling distance used to determine the existence of the next cluster center.

6. Compute $D_{\min}(\mathbf{x}_j) = \min[D_i(\mathbf{x}_j)]$ 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.
7. Find $D_{\max}(\mathbf{x}_m) = \max[D_{\min}(\mathbf{x}_j)]$ 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 $D_{\max} > T/2$, then let $\mathbf{x}_m \Rightarrow z_{n+1}$, otherwise,
if $D_{\max} < T/2$, then terminate the procedure.

In words, if the maximum distance is greater than half the distance between the two closest cluster centers, then \mathbf{x}_m becomes a new cluster center, otherwise terminate the procedure.

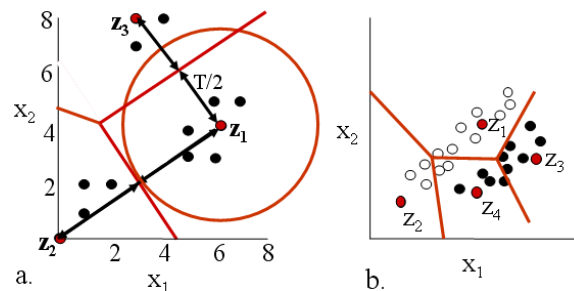
9. Increment the number of cluster centers by 1: $N_c = N_c + 1$.

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

11. Return to step 6.

Example:

- 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.
- 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 $m*n*i$ distance computations.



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.

Procedure:

1. Choose K initial cluster centers, $z_1, z_2, z_3, \dots, z_k$.

- Requires a list of desired clusters. 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 should be 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 $j = 1, 2, 3, \dots, k$; 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:

$$z_i(n+1) = \sum_{x \in S_i(n)} x$$

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

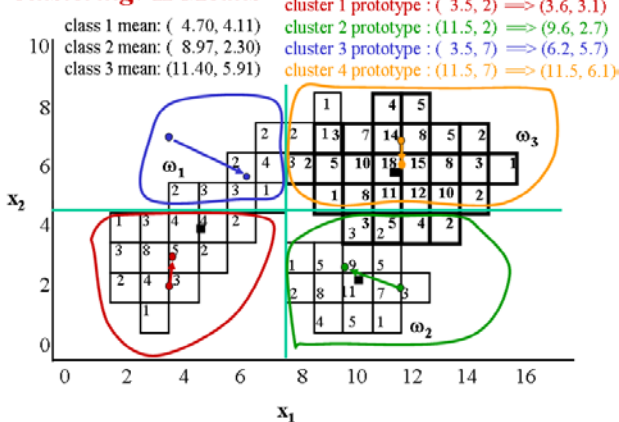
Compute the distance between each pair of points for the consecutive iterations. Then:

- If there is no substantial change, terminate the procedure. Some possible criteria for termination are:

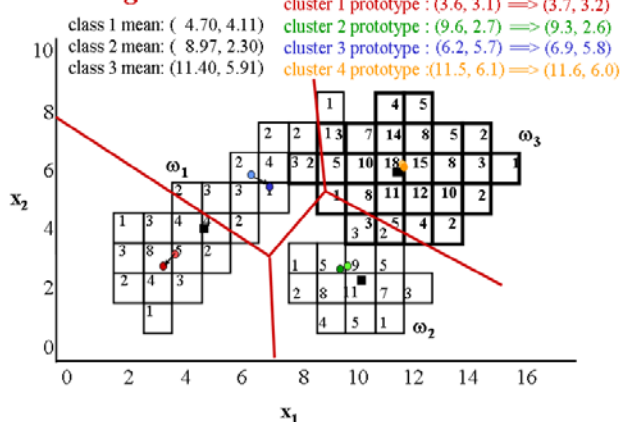
- if $|z_i(n+1) - z_i(n)| < T$ for all i
- if $\sum_{j=1}^k |z_i(n+1) - z_i(n)| < T$ for all i

- otherwise return to step 2. for the next iteration.

Clustering: K-Means



Clustering: K-Means



Enter

K = number of cluster centers desired.

ϵ_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

maxstdev = standard deviation to use around the class mean

and/or

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.

K-Means Parameters

Number of Classes

5

Change Threshold % (0-100)

5.00

Maximum Iterations

1

Maximum Stdev From Mean

Maximum Distance Error

Output Result to

☒ File
 ☐ Memory

Enter Output Filename

Choose

OK

Queue

Cancel

Help

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.

1. Select an initial set of clustering centers, z_1, z_2, \dots, 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 pr 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 = number of cluster centers desired.

The actual number of clusters returned at the end of clustering will be between $K/2$ and $2K$. (*ENVI: specify a minimum and maximum.*)

N_{\min} = minimum number of samples allowed for a viable cluster.

σ_{\lim} = maximum allowable size of a single cluster.

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.

N_{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.

ϵ = stopping criterion

Stop if the total change in position of the cluster centers falls below this limit.

3. Distribute samples among the cluster centers

$$\mathbf{x} \in \omega_j \text{ iff } |\mathbf{x} - \mathbf{z}_j| \leq |\mathbf{x} - \mathbf{z}_i|$$

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

$$i, j = 1, 2, 3, \dots, N_c; \quad i \neq j$$

N_c = number of cluster centers

4. *Discard sample subsets with fewer than N_{min} members*

If S_j contains N_j members and $N_j < N_{min}$, 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:

- incorporated into other clusters
- ignored for this iteration
- assigned to an unclassified group and ignored for the rest of the procedure.

5. *Compute new cluster centers for each set S_j*

$$\mathbf{z}_j = \frac{1}{N_j} \sum_{\mathbf{x} \in S_j} \mathbf{x}, \quad j = 1, 2, 3, \dots,$$

6. *Compute the cluster size parameter, D_j*

$$D_j = \frac{1}{N_j} \sum_{\mathbf{x} \in S_j} |\mathbf{x} - \mathbf{z}_j|, \quad j = 1, 2, 3, \dots, N_c$$

7. *Compute the overall size, D_s*

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

$$D_s = \frac{1}{N} \sum_{j=1}^{N_c} N_j D_j \quad \text{where } N \text{ is the total number of pixels: } N = \sum_{j=1}^{N_c} N_j$$

8. *Branch Point*

- on the last iteration, set $D_{lump} = 0$ and go to step 12,
- if $N_c < K/2$ (or K_{min}) go to step 9,
- on even iterations or if $N_c > 2K$ (or K_{max}), go to step 12,
- otherwise go to step 9.

9. *Find the standard deviation vector, σ_i , for each class.*

$$\sigma_{ij} = \frac{1}{N_j} \left[\sum_{\mathbf{x} \in S_j} (x_i - z_{ij})^2 \right]^{1/2} \quad \text{or} \quad \sigma_j = \frac{1}{N_j} \left[\sum_{\mathbf{x} \in S_j} (\mathbf{x} - \mathbf{z}_j)^2 \right]^{1/2}$$

σ_{ij} represents the standard deviation of all the samples in S_j along the principal coordinate axes, x_i , in feature space.

10. *Find the maximum component of each σ_i*

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

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

11. Test for splitting

IF $\sigma_{j,\max} > \sigma_{\lim}$
(if the variance for any cluster exceeds the limiting value)
AND EITHER $D_j > D_s$ and $n_j > 2(K+1)$ (or $n_j \geq K_{\max}$)
(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 $\mathbf{z}_{N_c} = (z_i, z_2, \dots, z_j + ks_{j,\max}, \dots, z_n)$
 $\mathbf{z}_{N_c} = (z_i, z_2, \dots, z_j - ks_{j,\max}, \dots, z_n)$
 $N_c = N_c + 1$
If splitting occurs then go to step 2; otherwise continue.

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

$$D(i,j) = | \mathbf{z}_i - \mathbf{z}_j |, \quad i = 2, \dots, N_c$$

$$j = 1, 2, \dots, i-1$$

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

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

14. Lump clusters

$D_k(i,j)$ is the distance between cluster centers at \mathbf{z}_i and \mathbf{z}_j . If neither \mathbf{z}_i nor \mathbf{z}_j has been used for lumping during this iteration, then merge the two clusters:

$$\mathbf{z}_{\text{lump}} = \frac{1}{N_i + N_j} [\mathbf{z}_i N_i - \mathbf{z}_j N_j]$$

delete \mathbf{z}_{i1} and \mathbf{z}_{j1} and decrement N_c : $N_c = N_c - 1$

Note: Only pairwise lumping is allowed. If any \mathbf{z}_i appears in more than one pair, only the first pair will be merged

15. Continue or terminate

IF N_{iter} iterations have already been performed.
OR IF on the k th iteration: $N_c(k) = N_c(k-1)$
and $| \mathbf{z}_i(k) - \mathbf{z}_i(k-1) | < \epsilon$
THEN TERMINATE
ELSE CONTINUE

Specify process parameters

Number of Clusters

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.

Stopping Criteria

N_{iter} = number of iterations allowed.

If the change in the prototype vectors is very small (system is converging, errors are small), the computations should stop.

Distribution Characteristics

N_{min} = minimum number of samples allowed for a viable cluster.

σ_{lim} = maximum allowable size of a single cluster.

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.

Criteria for allowing unclassified pixels

Maximum Stdev from mean

Maximum Distance Error

ISODATA Parameters

Number of Classes: Min 5 Max 10

Maximum Iterations 1

Change Threshold % (0-100) 5.00

Minimum # Pixel in Class 0

Maximum Class Stdev 1.000

Minimum Class Distance 5.000

Maximum # Merge Pairs 2

Maximum Stdev From Mean

Maximum Distance Error

Output Result to ☒ File ☐ Memory

Enter Output Filename Choose

OK Queue Cancel Help