Unsupervised Classification

- No training data are needed
- Only has to specify the number of classes

Clustering

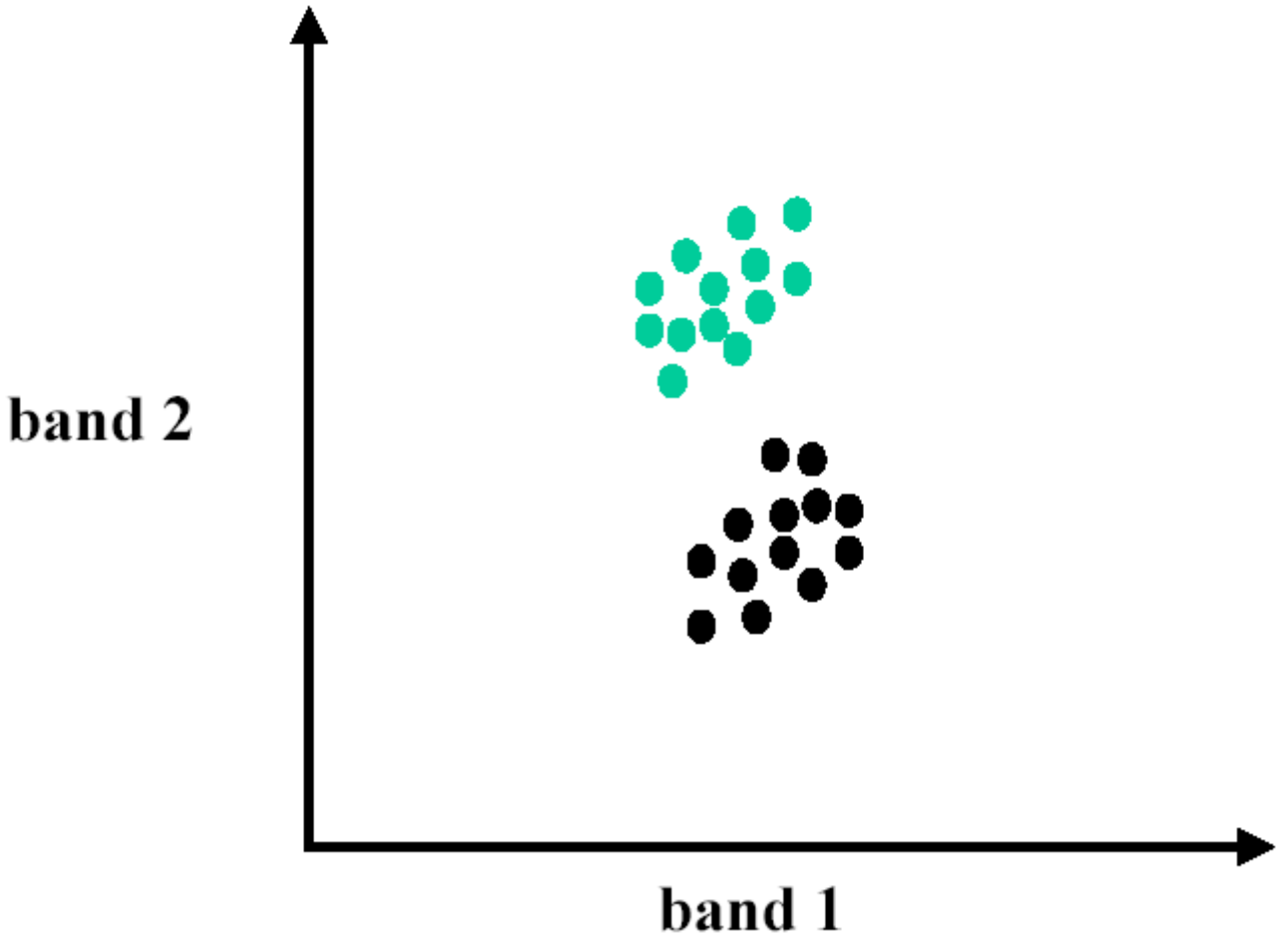
Pattern classification by distance functions.

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.

  Note: The "distance" will change when data are calibrated (digital counts ==> radiance) or atmospherically corrected or rescaled (PCA).
- Most effective if the clusters are disjoint.
- Requires the least amount of prior information to operate.

Clustering



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

Min. 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 space.
- A circular pattern suggests that there is no correlation among the measurements (x1, x2, . . .), and that the variation within the class is the same for all measurements.

Min. 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, $\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \ldots, \mathbf{z}_m$.
- A discriminant function for the i-th class may be derived directly from the expression for the ***Euclidean distance, $D_i(x)$***, of a measurement vector, **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

Min. Distance Classification

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

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

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

Min. Distance Classification Single Prototype

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

1. 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.

Min. Distance Classification Single Prototype

2. 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 and 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, $\mathbf{x}'\mathbf{z}_i$

Min. Distance Classification
Single Prototype

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

Another option is to simplify the distance computation itself. For example, replace the Euclidean distance with the crude, but simple "taxicab" distance:

$$D_1(x) = \sum_{k=1}^{N_b} |x_k - z_{ki}|$$

N_b: # of variables (spectral bands)

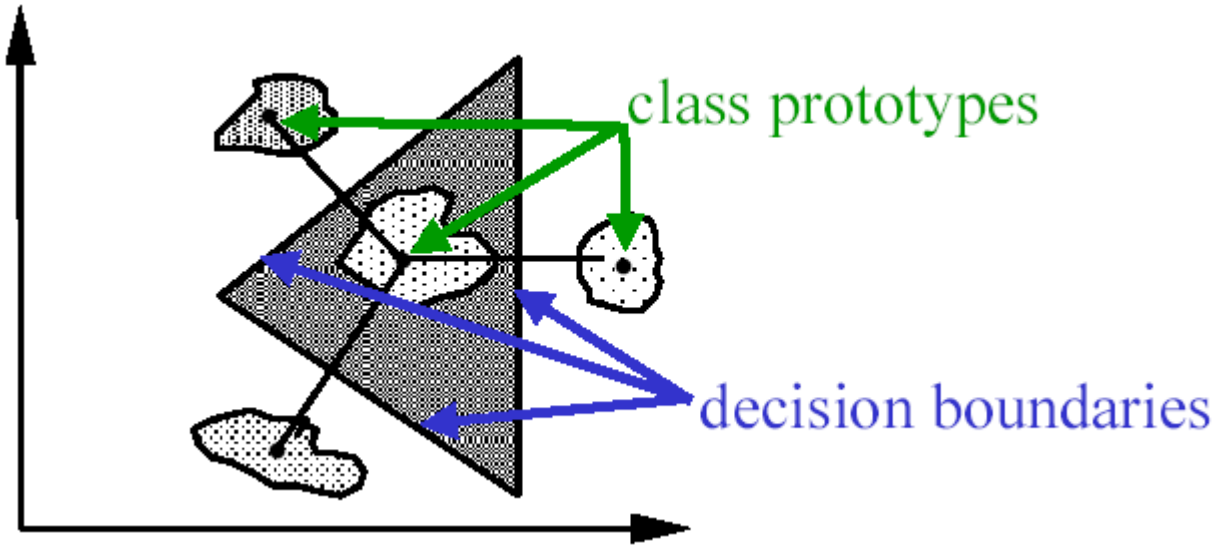
The taxicab distance is computed as a simple sum, but overestimates diagonal distances.

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.



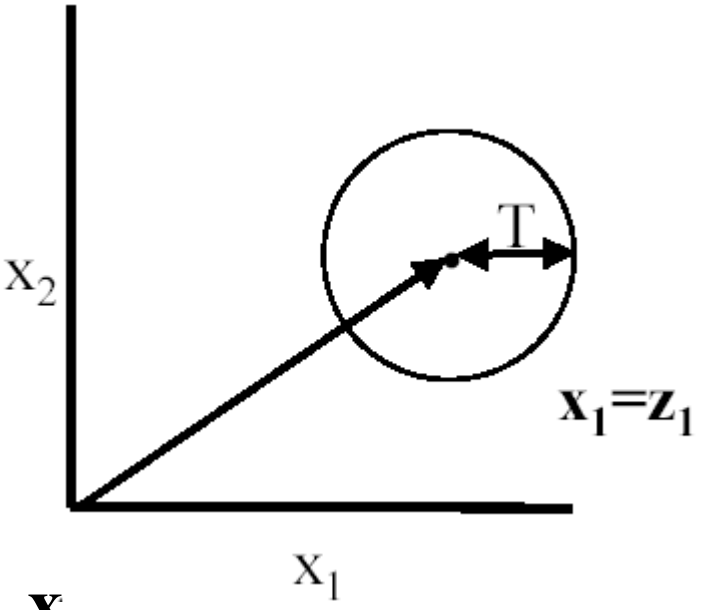
Seeking Clusters

- The results of clustering will depend strongly on the choice of the prototype.
- Requiring that the user select the prototypes for each class can reduce the utility of a clustering procedure.
- Alternatively
  - let the user-selected prototypes serve as a first approximation
  - automate the prototype definition

A simple cluster seeking procedure

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, z_1.

A simple cluster seeking procedure

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

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

6. Compare the $D_i(x)$ with T.

a) if Di(x) < T, then x ∈ ωi.

b) if Di(x) > T, for all i, then let x become a new prototype vector: Assign **x --> z_{N+1}**. (Do not compute DN+1 for pixels already assigned to an existing class.

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

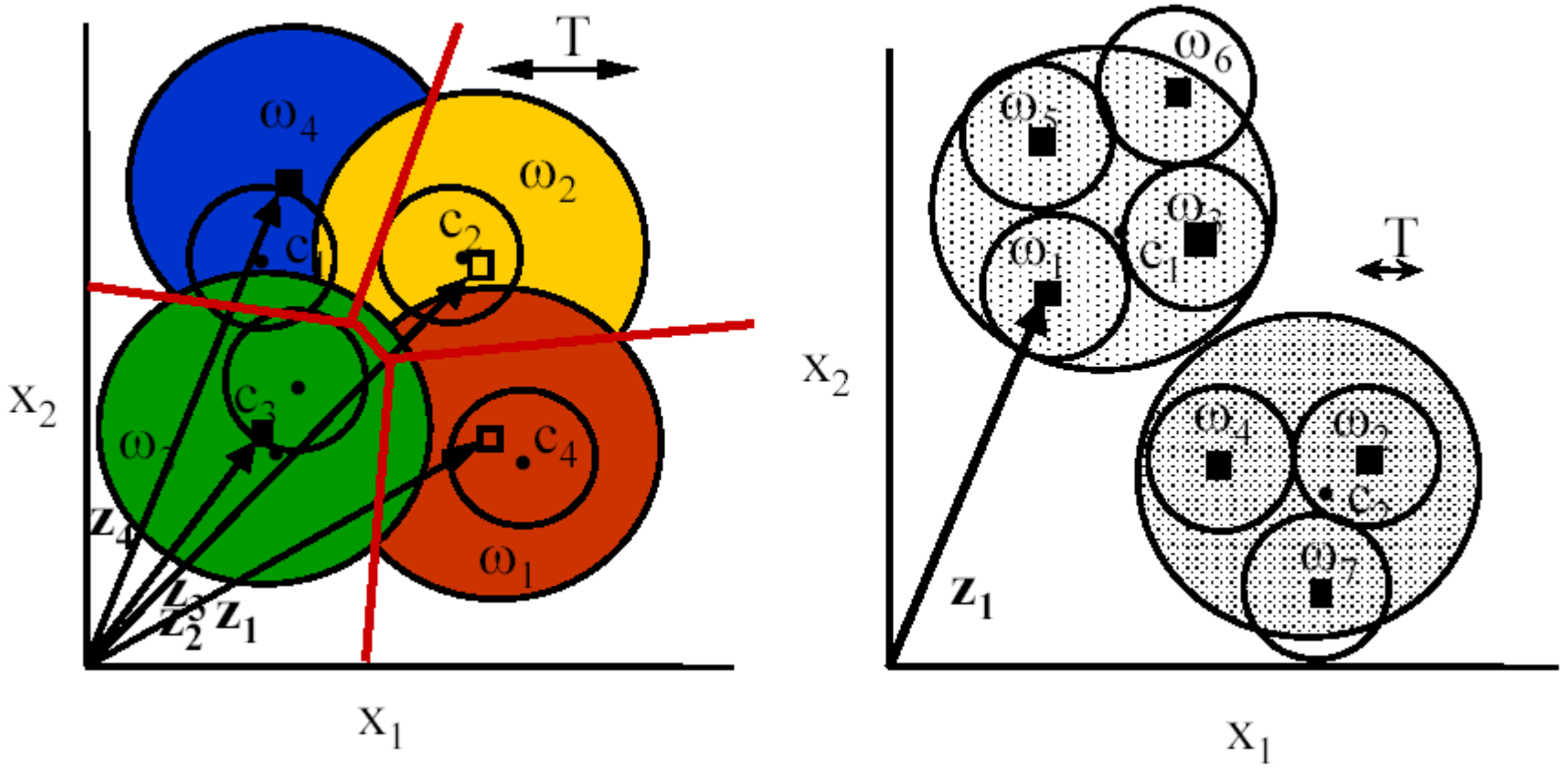
8. After all pixels have been assigned to a cluster center, recompute the Di(x) and reassign pixels according to the minimum Di(x)

A simple cluster seeking procedure

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

ω_1, ω_2, ω_3, ω_4 represent assigned clusters

Maxmin 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 the cluster regions which 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.

Maxmin Distance Procedure

1. Select a pixel, **x**, from the image at random.
2. Let **x** be the first cluster center, **x** ==> $\mathbf{z}_1$
3. Sort through the remaining pixels to find the pixel, **x**, which is farthest from $\mathbf{z}_1$.
4. Let the most distant pixel be the second cluster center, **x** ==> $\mathbf{z}_2$
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.

Maxmin Distance Procedure

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 $\mathbf{x}_m \Longrightarrow \mathbf{z}_{n+1}$, otherwise, if $D_{max} < T/2$, exit.

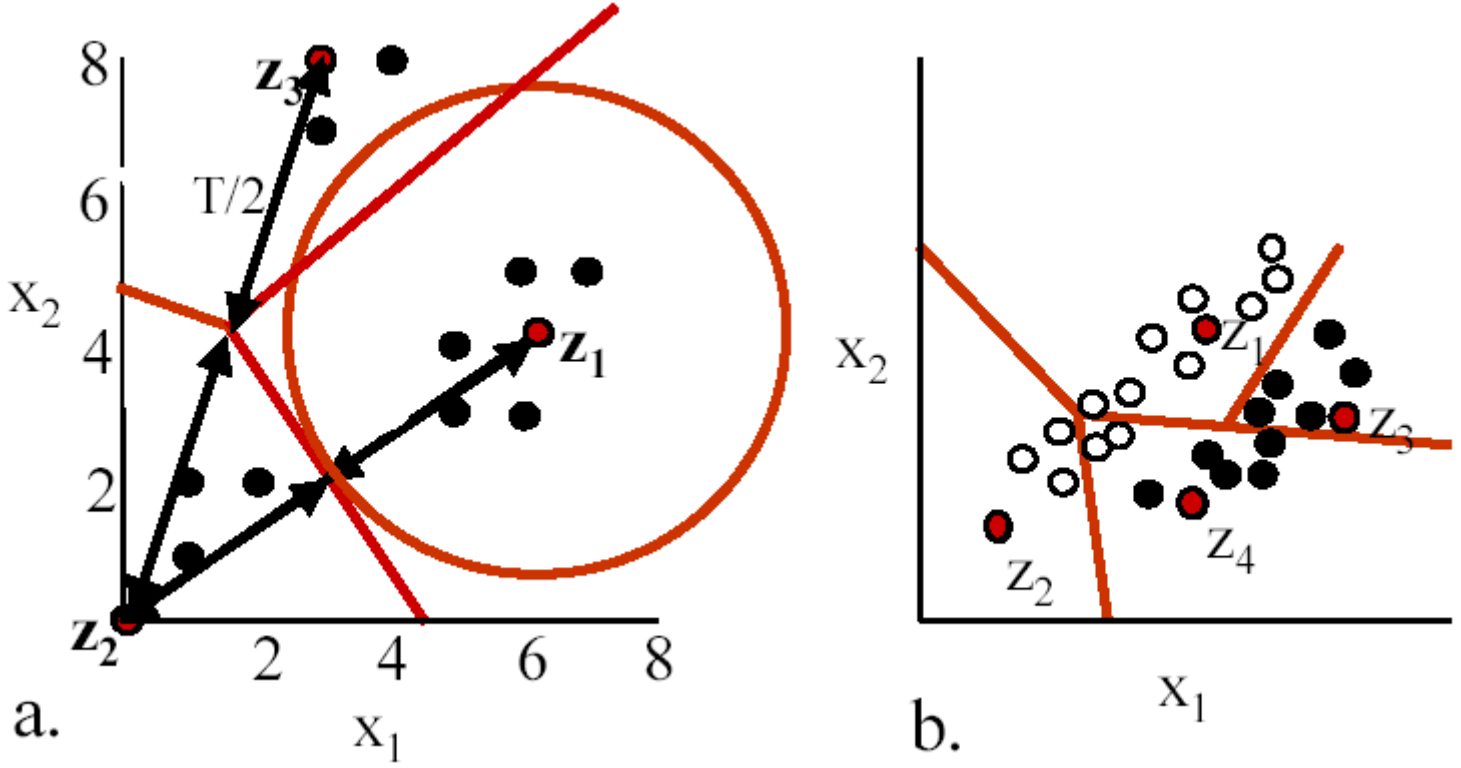
   In words, if the maximum distance is greater than half the distance between the two closest cluster centers, then $\mathbf{x}_m$ becomes a new cluster center, otherwise terminate the procedure.

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

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

11. Return to step 6.

Maxmin Distance Example



Cluster centers are labeled according to the order of selection:

a) clearly separable, circular clusters of about the same size can often be located by the maximin algorithm, but
b) more realistic distributions may be improperly partitioned.

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 x n pixel image there will be i+1 clusters requiring m*n*i distance computations.

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.

K-Means

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

- Requires a list of desired clusters. These may be supplied by the user or selected by the algorithm.
- 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.

K-Means

2. Distribute the samples among the K means

Samples should be assigned to the class represented by the nearest cluster center, i.e.:

$$x \in S_i(n) \text{ if } |x - z_i(n)| \leq |x - z_j(n)|$$

for all j = 1, 2, 3, . . . , k; where i ≠ j

$S_i(n)$ is the set of samples whose cluster center is $\mathbf{z}_i(n)$, 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 $\mathbf{z}_i$. The new cluster center, $\mathbf{z}_i(n+1)$ will be the mean of all the points in $S_i(n)$ such that:

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

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. If there is no substantial change, terminate the procedure, otherwise return to step 2. for the next iteration. Some possible criteria for termination are:

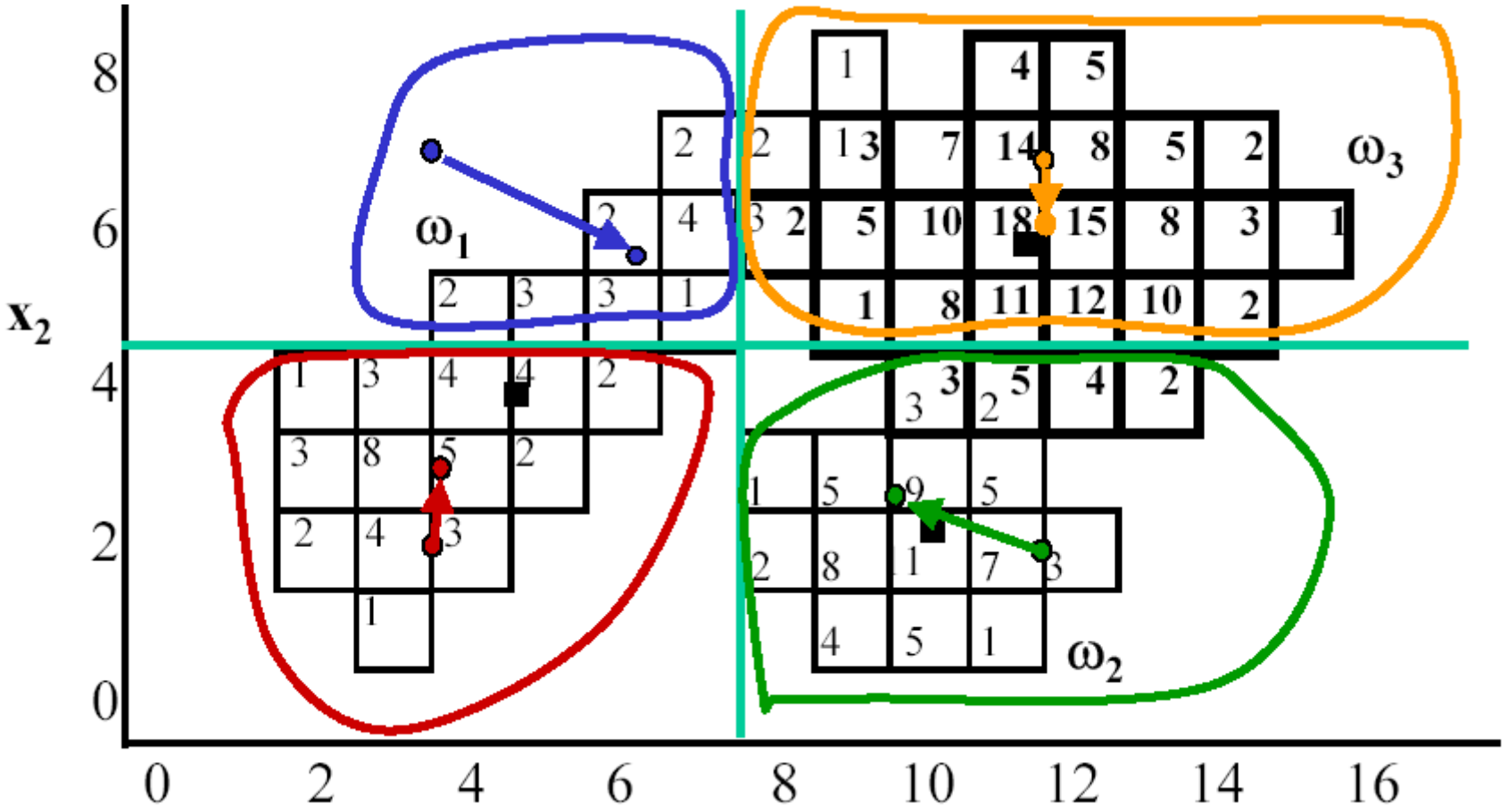
a) if $|\mathbf{z}_i(n+1) - \mathbf{z}_i(n)| < \mathbf{T}$ for all i

b) for all i, if $\sum_{j=1}^{k} |\boldsymbol{z}_i(n+1) - \boldsymbol{z}_j(n)| < T$

K-Means

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

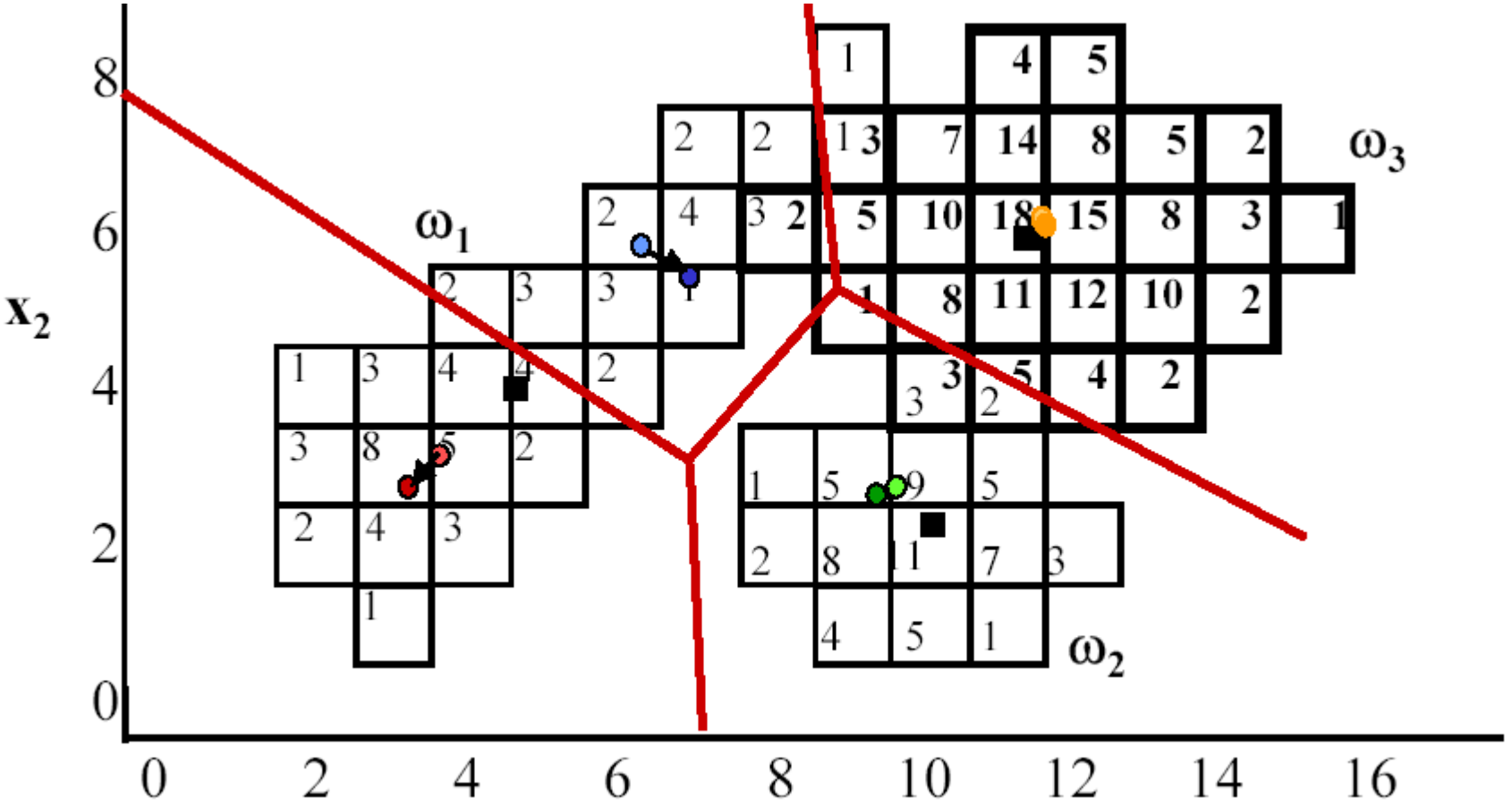
cluster 1 prototype : (3.5, 2) ==> (3.6, 3.1)
cluster 2 prototype : (11.5, 2)==> (9.6, 2.7)
cluster 3 prototype : (3.5, 7) ==> (6.2, 5.7)
cluster 4 prototype : (11.5, 7)==> (11.5, 6.1)



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) => (3.7, 3.2)
cluster 2 prototype : (6.2, 5.7) => (9.3, 2.6)
cluster 3 prototype : (9.6, 2.7) => (6.9, 5.8)
cluster 4 prototype : (11.5, 6.1) => (11.6, 6.0)



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.

ISODATA

1. Select an initial set of cluster centers, $z_1, z_2, z_3, \ldots, z_{nc}$.

- 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;
- Most simply assign the cluster centers (often with an undocumented procedure).
- 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.

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_{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.

ISODATA

3. Distribute samples among the cluster centers

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

S_j = the set of samples assigned to cluster center z_j

i,j = 1, 2, 3, ..., Nc; $i \neq j$

Nc = number of cluster centers

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

If Sj contains N_j members and $N_j < N_{min}$, discard $\mathbf{z}_j$ 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

b. ignored for this iteration

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

ISODATA

5. Compute new cluster centers for each set S_i

$$z_j = \frac{1}{N_j} \sum_{x \in S_j} x \,, j = 1, 2, 3, \ldots$$

6. Compute the cluster size parameter, D_j

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

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.

ISODATA

8. Branch Point

a) on the last iteration, set $D_{lump} = 0$ and go to step 12,

b) if Nc < K/2 *(or Kmin)* go to step 9,

c) on even iterations or if Nc > 2K *(or Kmax)*, go to step 12,

d) otherwise go to step 9.

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

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

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.

ISODATA

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* $n_j > K_{max}$*)*

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

OR $Nc < K/2$ (if there are too few clusters)

THEN split the cluster, Sj, into two new clusters.

Let $\mathbf{z}_{Nc} = (z_i, z_2, \ldots, z_j + ks_{j,max}, \ldots, z_n)$

$\mathbf{z}_{Nc} = (z_i, z_2, \ldots, z_j - ks_{j,max}, \ldots, z_n)$

$Nc = Nc + 1$

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

ISODATA

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

$$D(i,j) = | \mathbf{z}_i - \mathbf{z}_j | \text{ , } i = 2, \ldots, Nc$$
$$j = 1, 2, \ldots, i\text{-}1$$

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

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

ISODATA

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:

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

delete $\mathbf{z}_{il}$ and $\mathbf{z}_{jl}$ and decrement Nc: Nc= Nc - 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.

ISODATA

15. Continue or terminate

IF N_{iter} iterations have already been performed.

OR IF on the kth iteration: $Nc(k) = Nc(k-1)$

and $| \mathbf{z}_i(k) - \mathbf{z}_i(k-1) | < \varepsilon$

THEN TERMINATE

ELSE CONTINUE

ISODATA

ISODATA Processing 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_{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.

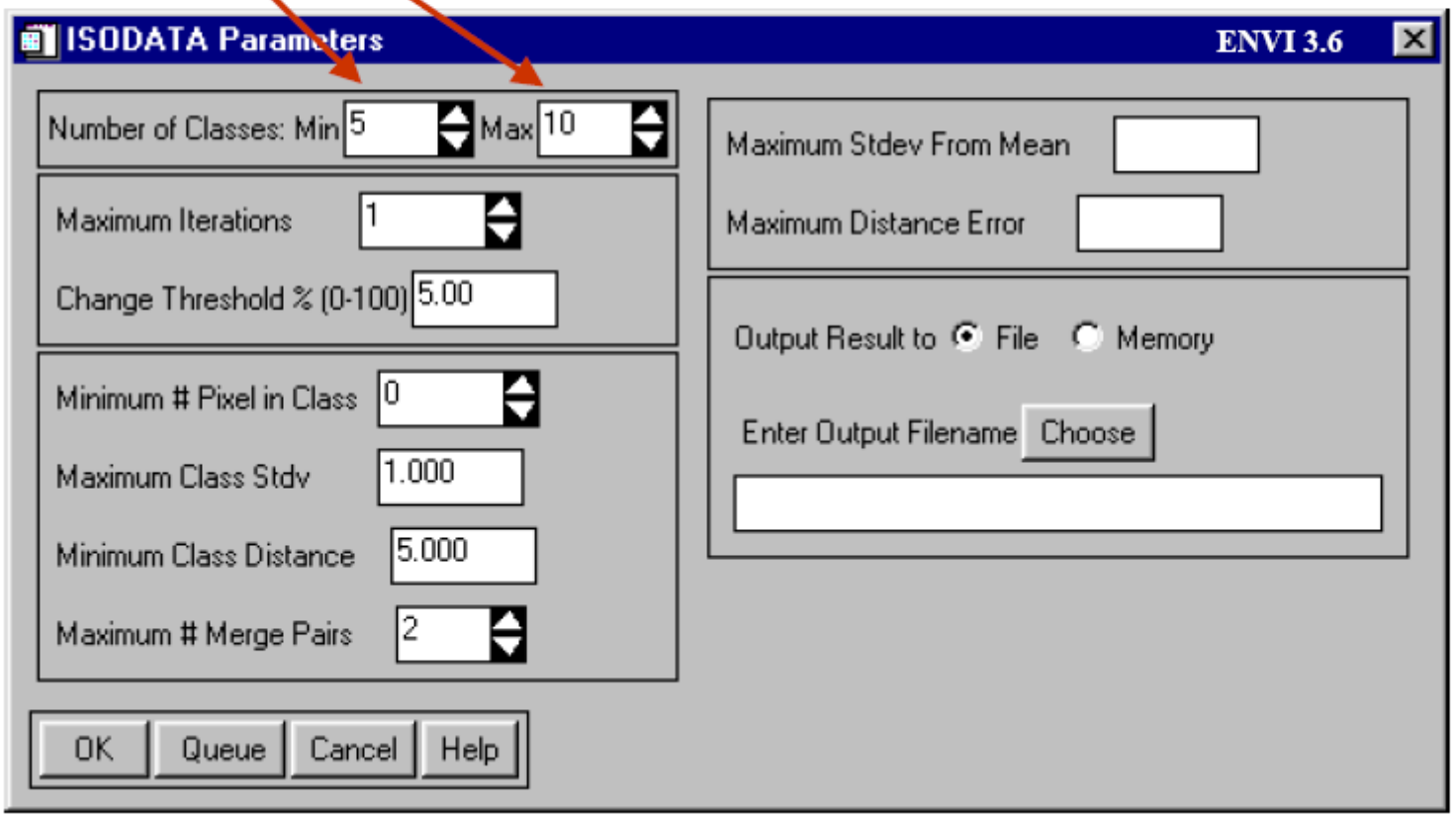
N_{lump} = maximum number of pairs of cluster centers which can be lumped during one iteration.

N_{iter} = number of iterations allowed.

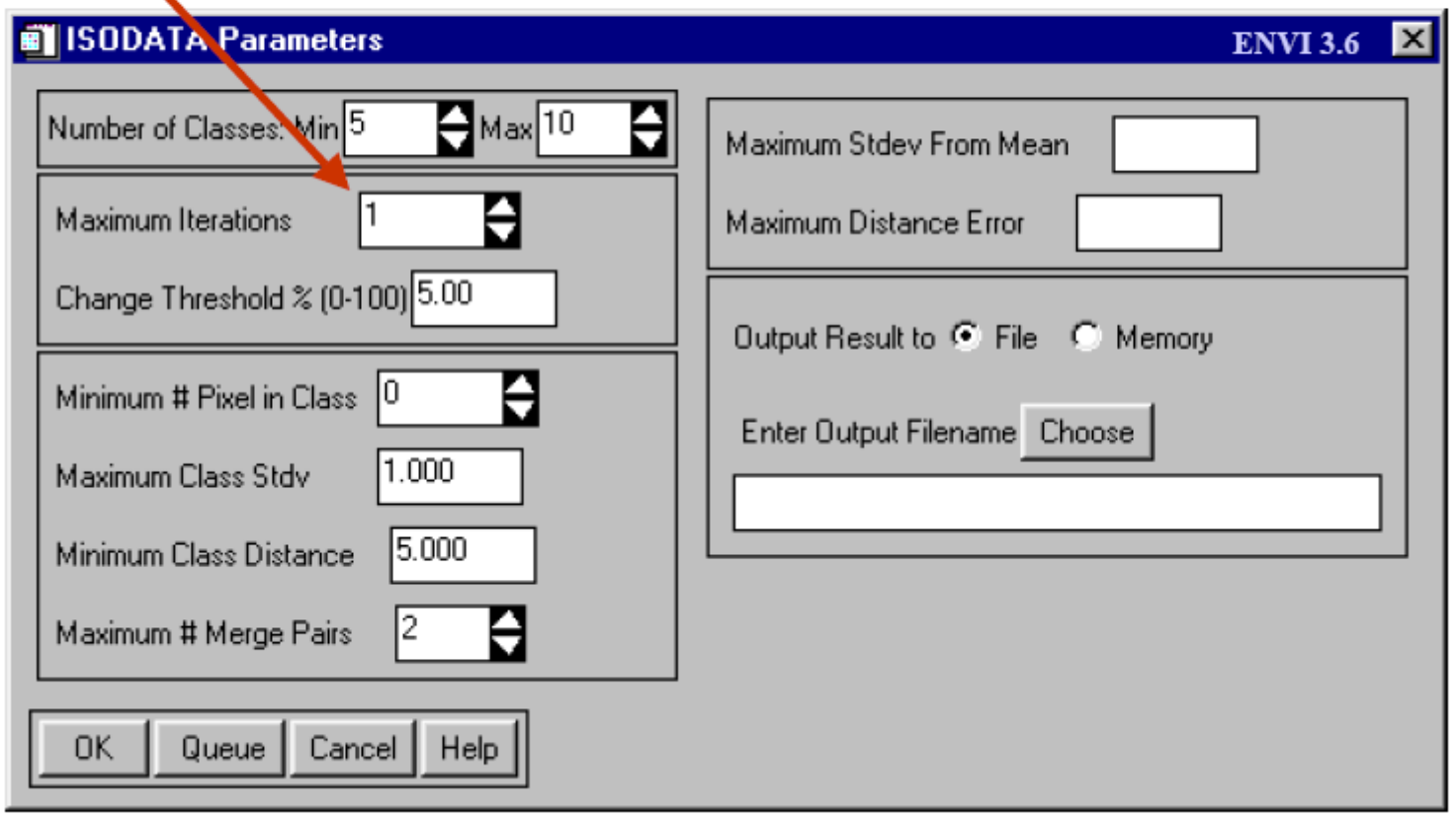
If the procedure is going to work given the initial parameters, it should converge fairly quickly, i.e., in a few iterations. If the routine does not converge "quickly" it could go on indefinitely.

K = number of cluster centers desired.

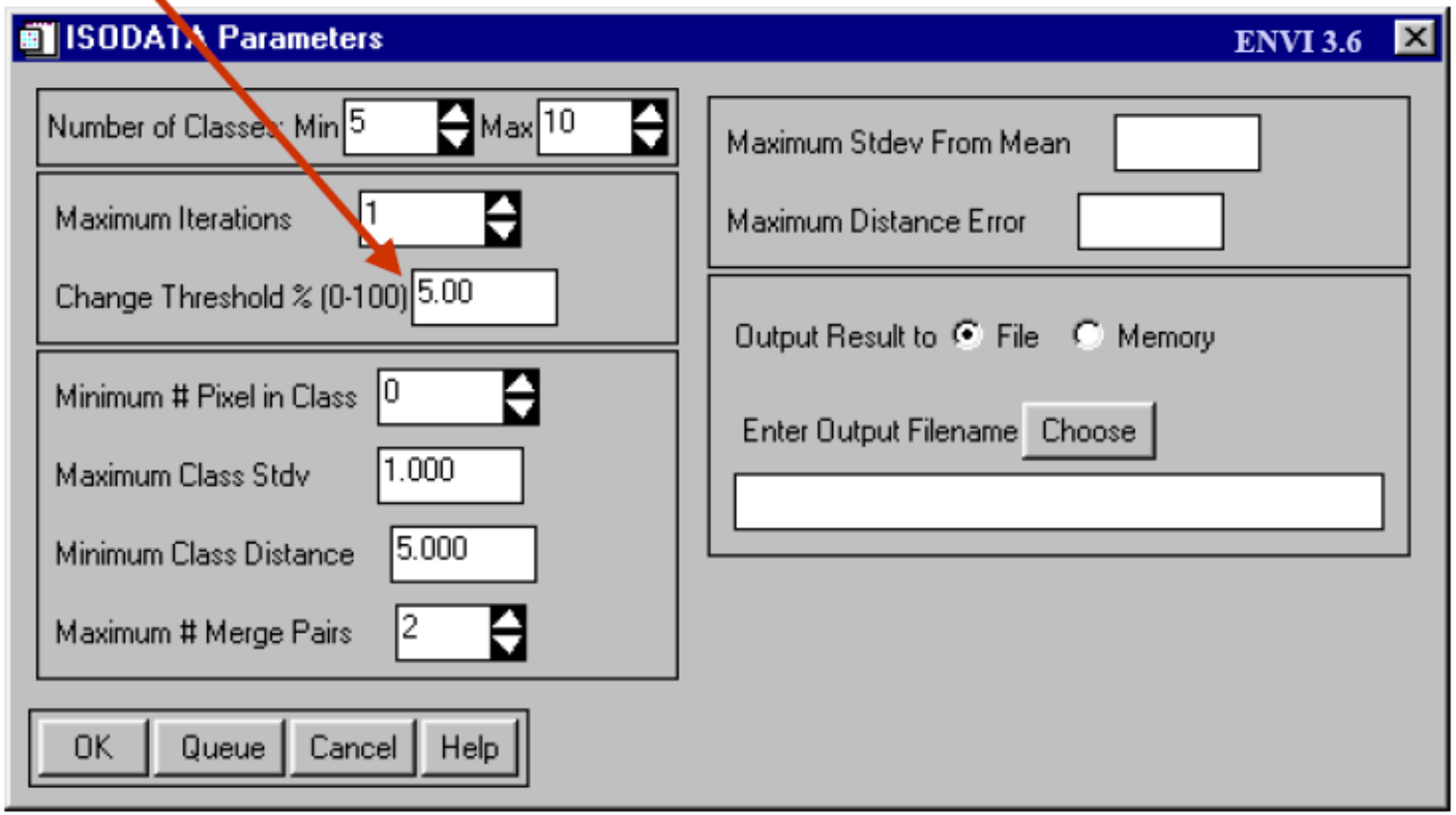
- In some implementations, the actual number of clusters returned at the end of clustering will be between K/2 and 2K.
- ENVI allows the user to specify the acceptable range explicitly.



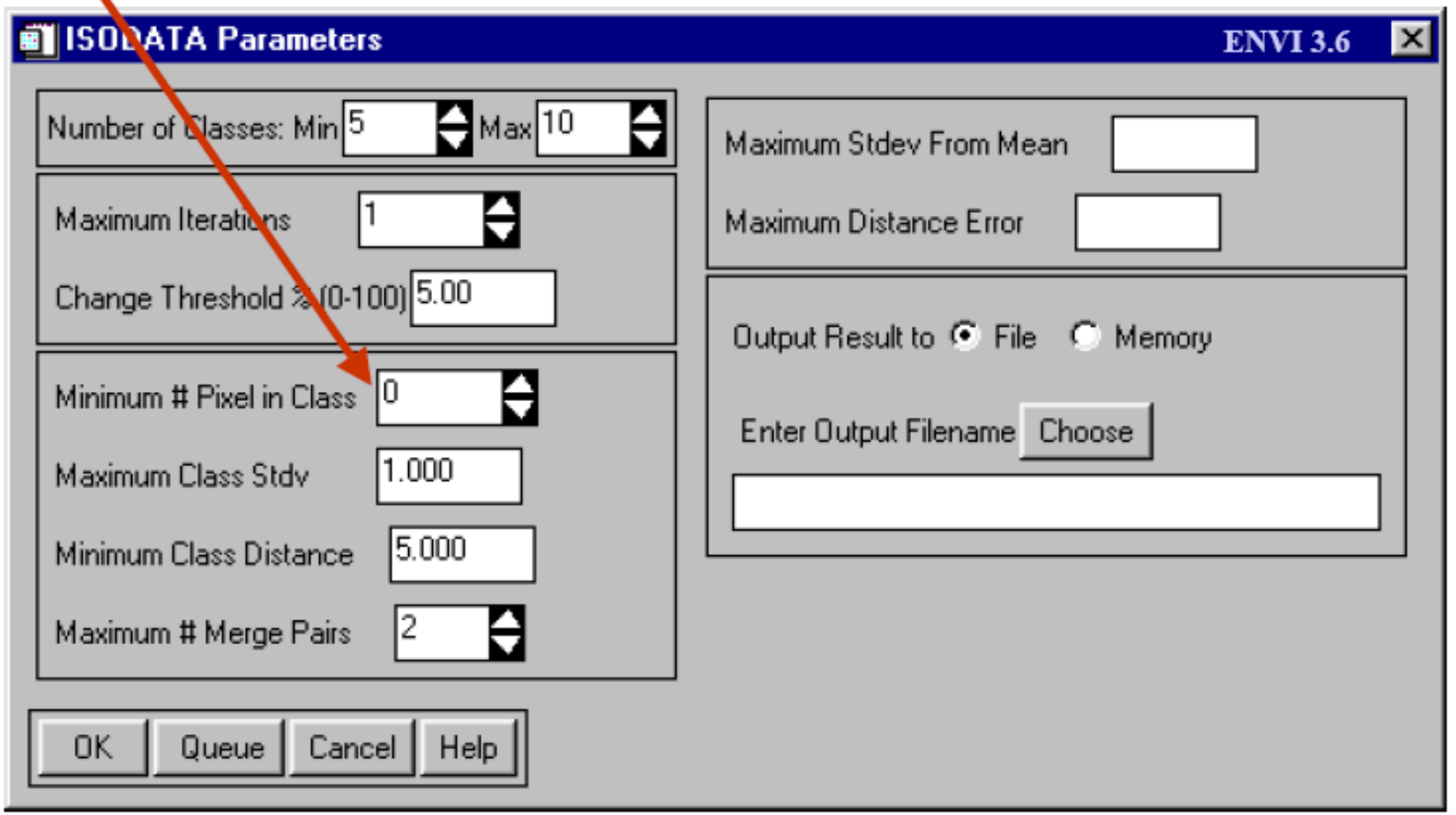
N_{iter} = number of iterations allowed.



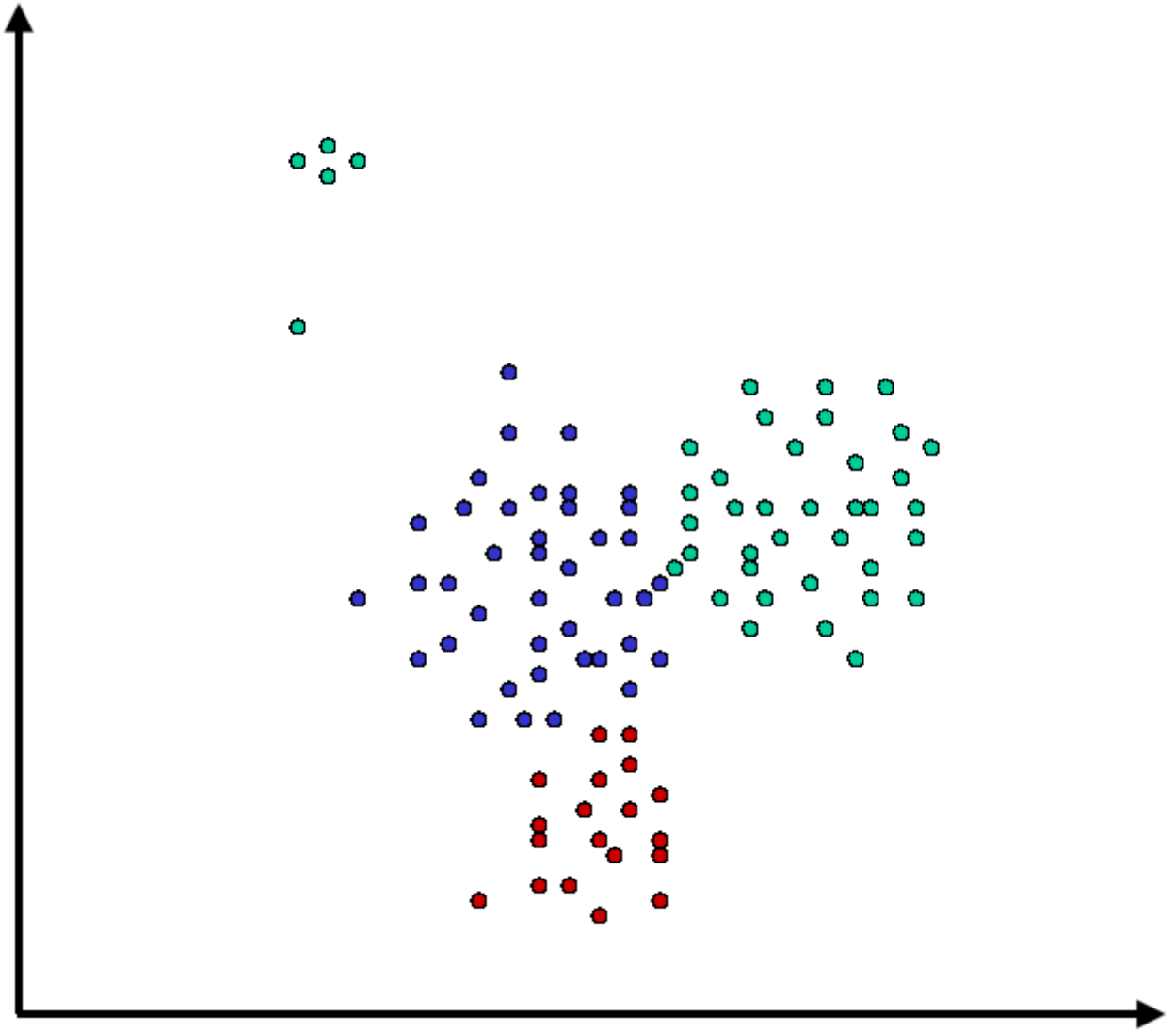
If the change in the prototype vectors is very small (system is converging, errors are small), the computations should stop.



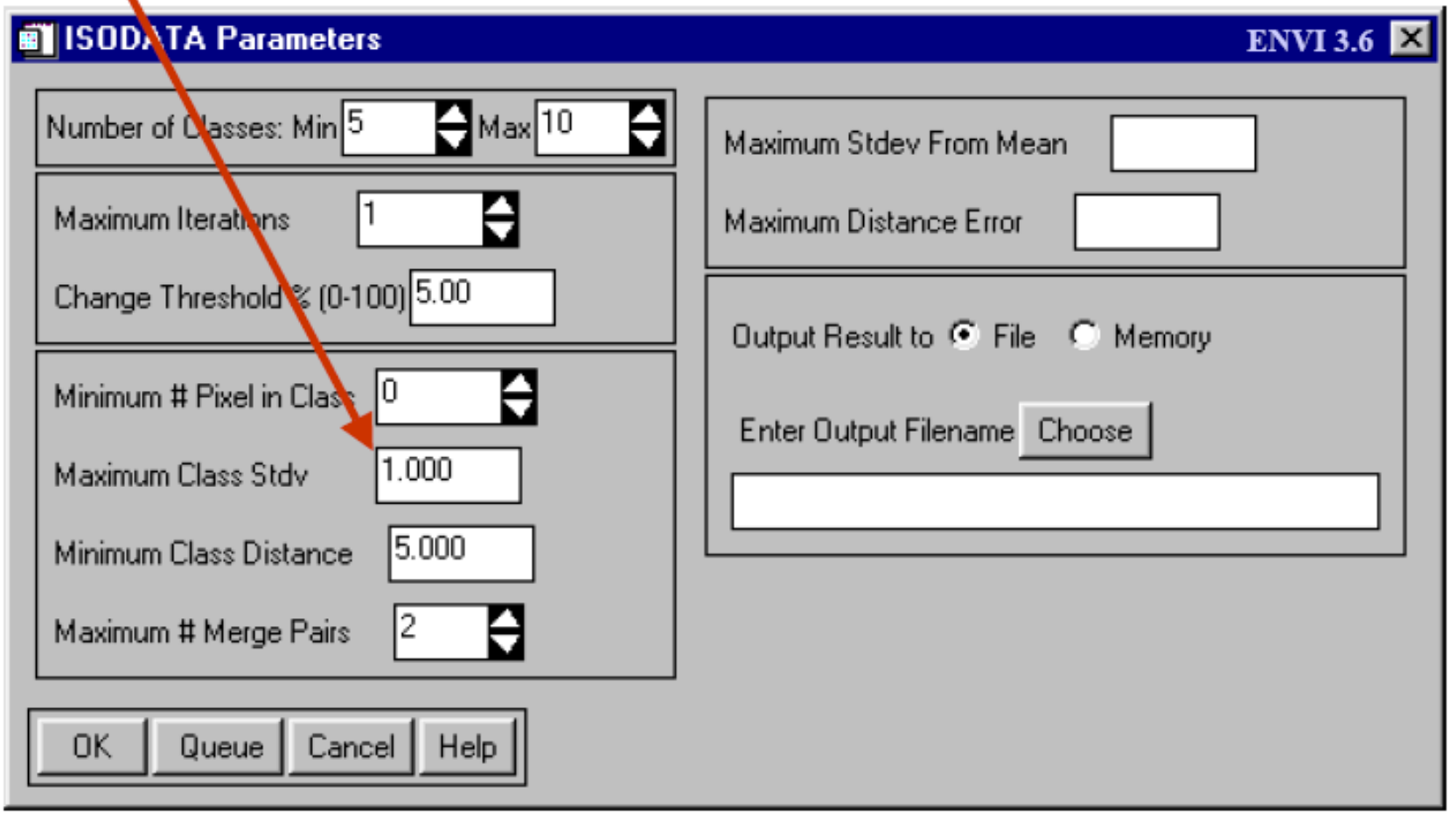
N_{min} = minimum number of samples allowed for a viable cluster.



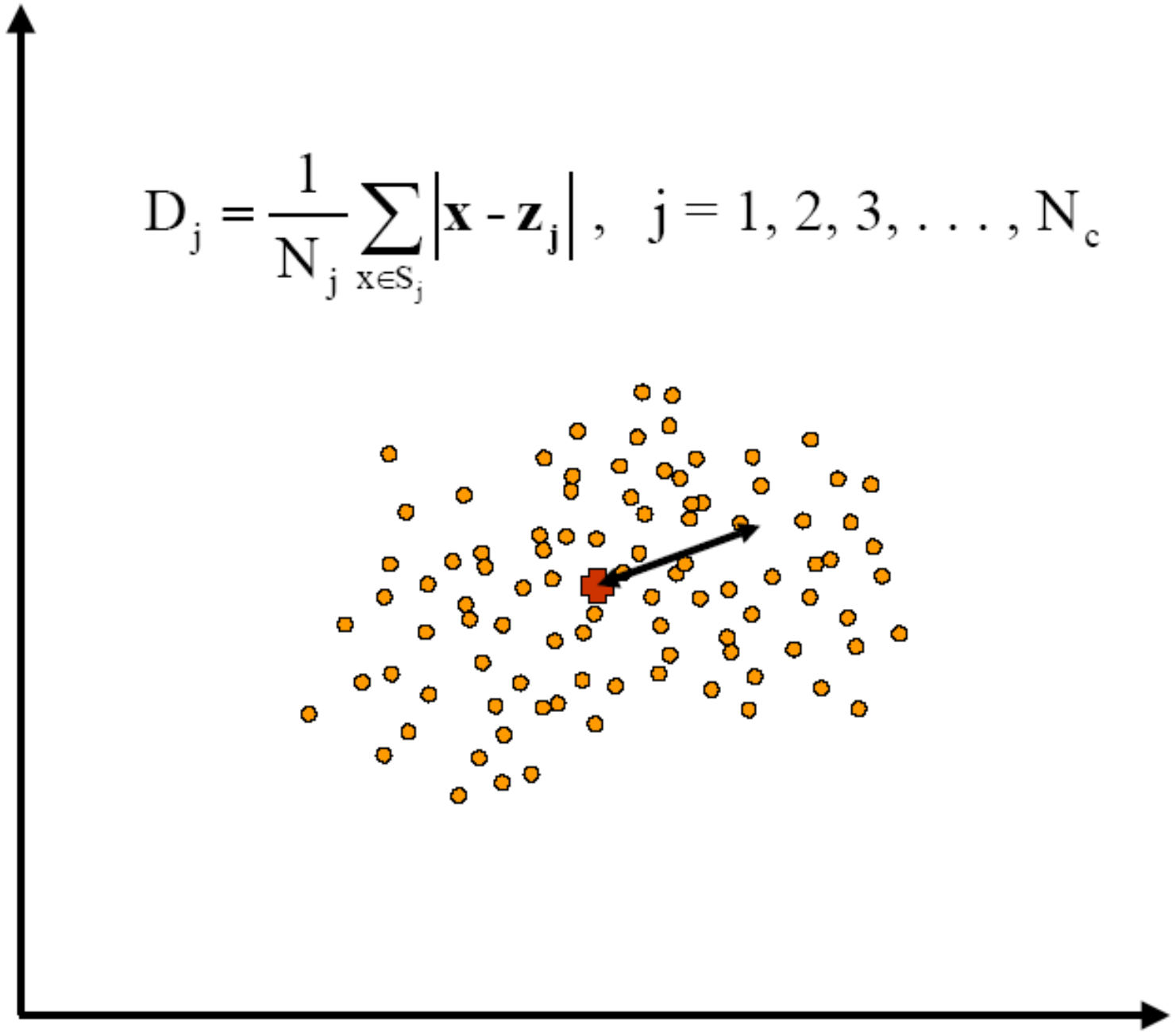
Unsupervised Classification



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



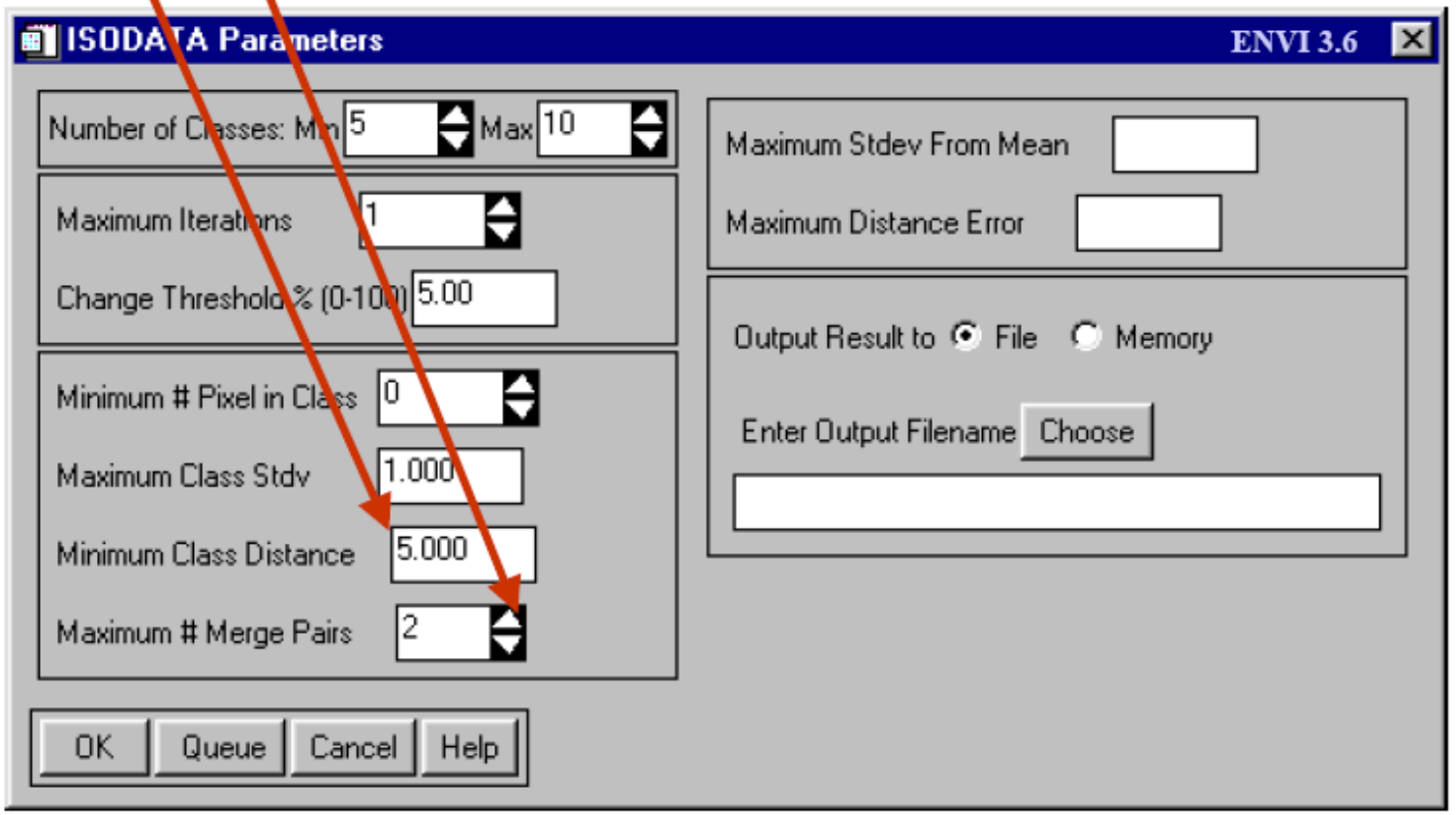
$$D_j = \frac{1}{N_j} \sum_{x \in S_j} \left| \mathbf{x} - \mathbf{z}_j \right|, \quad j = 1, 2, 3, \ldots, N_c$$



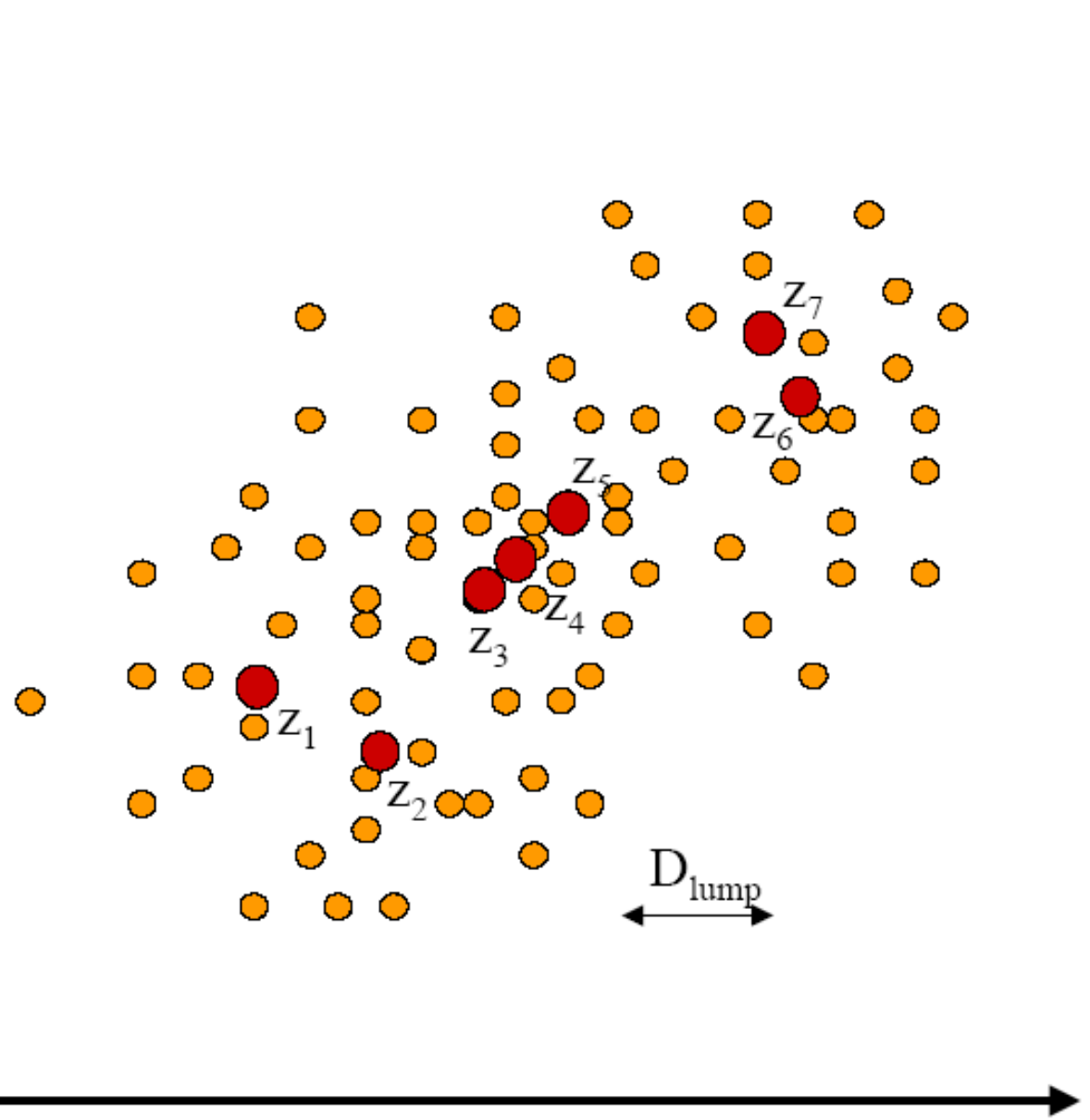
D_{lump} = lumping parameter -- a distance in gray values.

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	
z_3	z_4
z_6	z_7
z_4	z_5



Criteria for allowing unclassified pixels