CLUSTERS

Each element in a set of data elements is assigned to a representative group (cluster) , such that

-elements in each group share a common feature

-elements do not share that feature with elements in a different group

Why cluster?

Generally a cluster will allow **characterization of a large portion of the data** by the cluster itself, or possibly by a representative data point

Clustering does not, in and of itself, have *predictive* value.

CLUSTERS

- Often we are given only the data, with neither rules nor labels, and must determine the natural group membership inherent in the data themselves, with no prior knowledge of an outcome or label.

- Typically the features are numerical coordinates in n-space

- Many other features could be used, and would obviously change the clustering
  - Animal, Vegetable, Mineral
  - Earth, Wind, and Fire
  - Color
  - Shape
  - Number of hooves and wings
  - Number of STRs in a locus
  - Disease types
  - Genes expressed

CLASSICAL CLUSTERING

Clustering is an art and a philosophy. What makes things similar? For example, how would you form clusters of these elements?

- lemon
- strawberry
- lime
- frog
- cardinal
- canary

Choosing the features

By color

- Green
  - Frog
  - Lime
- Yellow
  - Lemon
  - Canary
- Red
  - Cardinal
  - Strawberry

By kingdom

- Plant
  - Strawberry
  - Lemon
  - Lime
- Animal
  - Frog
  - Cardinal
  - Canary

Choosing k, the number of clusters

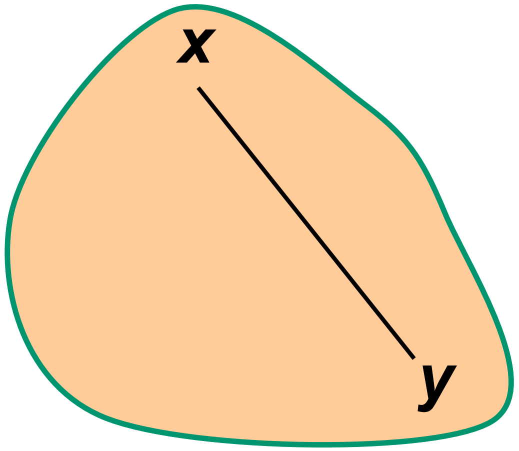
- A cluster for every data point is too many
- One cluster for all data points is too few
- What is the right number of clusters?
  - Minimization of a cost function *vs* elucidation of attributes

Numerical Data
Important Concept: ***k*** *vis à vis* ***m***

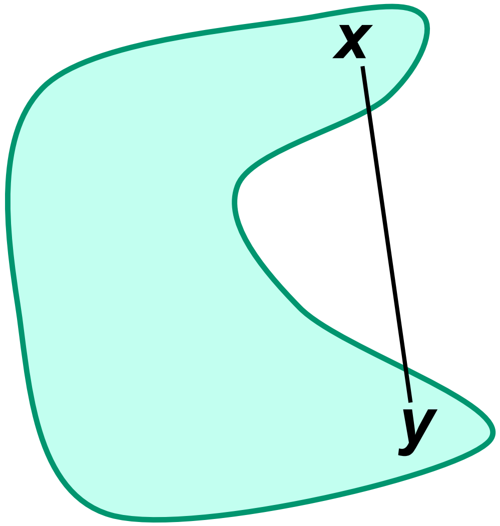
- Clusters may occur in hyperspace
  - You must be aware of the dimension (m) of the space in which you are working
- The n data points will have 1<k<n clusters
- The examples that follow illustrate k clusters in 2-dimensional space, but the algorithms all hold up for m-dimensional space

Concept: Convex Sets

Convex



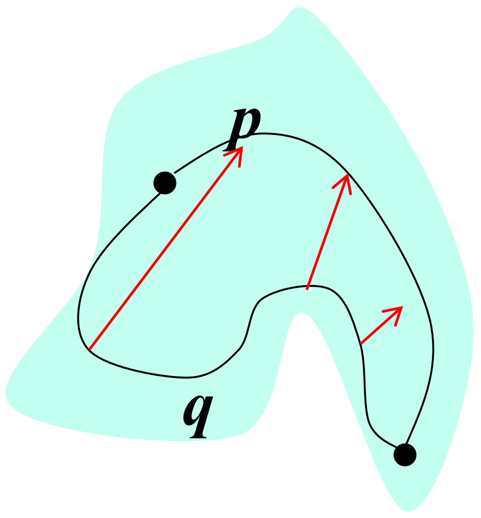
Non-convex



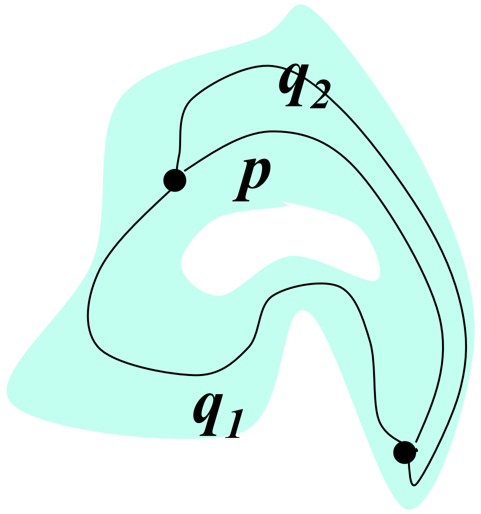
Convex: Every point on a line connecting any two points of the set must lie within the set.

Concept: Simply Connected Sets

If every path ***q*** within the set, sharing the same start and endpoints with another path ***p***, can be continuously deformed into ***p***, the set is simply connected, otherwise it is non-simply connected, or multiply connected



Simply Connected



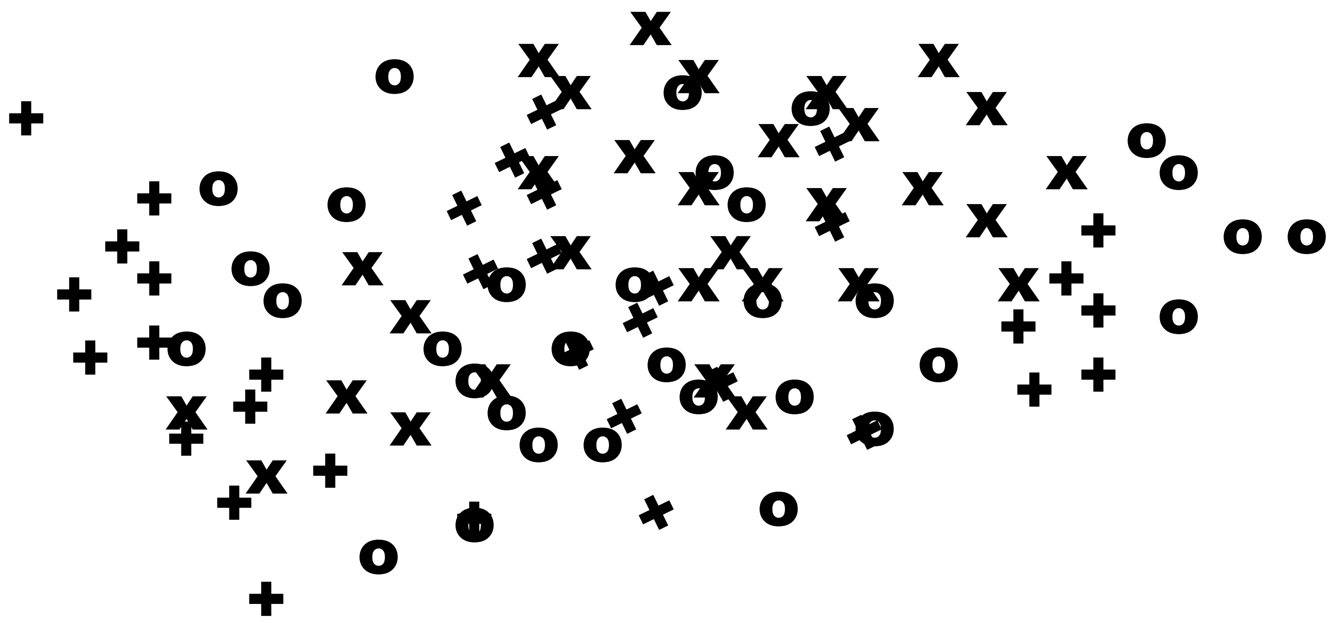
Multiply Connected

q_2 can be deformed into p, but q_1 cannot

Strategies for Clustering of Numerical Data

- Relation to a reference in each cluster
  - k-Means
- Data density and adjacency and sparseness
  - Neighbor-joining/Spectral
- Probability of existence
  - Model-based
- Topology-learning
  - Kohonen (self-organizing) maps

Sometimes there simply are no clustering schemes that will solve the problem



Are there actually clusters here ? Perhaps the clustering reflects some attribute other than distance-based separability?

Data Issues

- Types of Data
- Metrics
  - Similarities
  - Dissimilarities
- Scaling
- Handling Outliers
- Correlations

Caveat: Interval-scaled Data

- Interval Scaled
  - Distance between two points in one interval same as in another
  - Energy to heat 1 gm water from -40° to -38° same as heating 1 gram water from 1,000,000° to 1,000,002°

The usefulness of the difference depends on the context

From Finding Data in Groups-An Introduction to Cluster Analysis. Kaufman and Rousseeuw, 1989

Caveat:
Always look at the distribution!

- Binary
  - Symmetric: approximately equal choices- male/female, married/single
  - Asymetric: has 2 heads, has one head--info is only in the rarity (*cf* information theory) - default in the problem setup should be the common occurrence

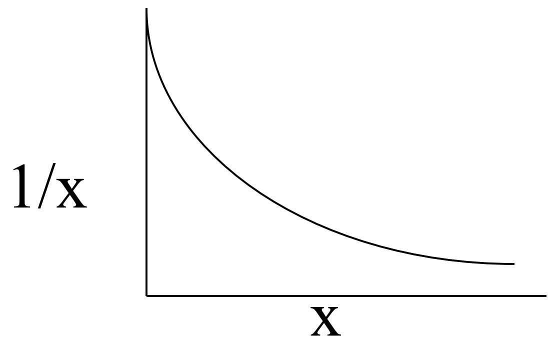
Different Dimensions

- Units change ⇒ patterns change

  eg changing cm to yards changes the scale
- Alternative: standardize the data- make it dimensionless
  - Also has a downside

Caveat:
Really, do always look at the distribution!

- Be particularly careful of ratios; they are exponentially distributed



  - Can use interval scaling
  - Can take log
  - Can do ordinal scaling (rank)

Similarities and Dissimilarities

- Distances are natural dissimilarities and are metrics
- Similarity would be "how much do they look alike?" - high score means very similar
  - Always nonnegative number
  - A correlation coefficient is a natural example (except for the negative part)
  - A common similarity is a 'reverse distance' measure , the Gaussian $e^{(-\alpha\|x-y\|)}$
  - Not always a metric

Definition: Metric

ρ is a function on a set **S**

$\rho(\mathbf{S}\times\mathbf{S})\rightarrow\Re$ such that:

- ρ is either 0 or a positive real number
- $\rho(a,a)=0$
- $\rho(a,b)= \rho(b,a)$
- $\rho(a,c) \leq \rho(a,b) + \rho(b,c)$

  where $a,b,c\in\mathbf{S}$

Then ρ is a metric

dist is an example of a metric

Metrics

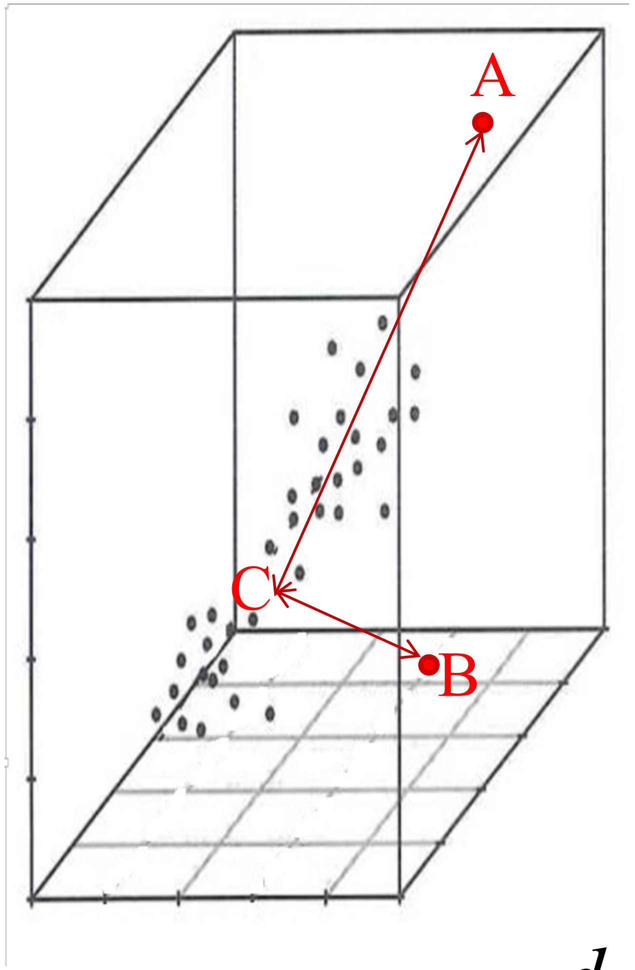
- Minkowski generalized metric

$$\left(\left|x_{i,1}-x_{j,1}\right|^{q}+\left|x_{i,2}-x_{j,2}\right|^{q}+\ldots\ldots\ldots+\left|x_{i,n}-x_{j,n}\right|^{q}\right)^{\frac{1}{q}}$$

  - Euclidian (q=2) or ‘usual’ : Pythagorean theorem
  - Manhattan (q=1) or taxicab: sum of sides
- Pipewrench : maximum

The Euclidian or 'usual metric' is the Minkowski with exponent 2 (Pythagorean)

Mahalanobis Distance



The axes are scaled differently. Therefore, the significance of comparing d_{AX} to d_{BX} is lost

The Mahalanobis distance takes the covariance matrix into account to ‘standardize’ the scales (expressed in std. deviations)

$$d_M = \sqrt{(\mathbf{x}-\boldsymbol{\mu})^T \,\mathbf{Cov}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$

where $\mathbf{x}$ is the vector of the data point and $\boldsymbol{\mu}$ is the vector of the mean point

Distance Between Sets

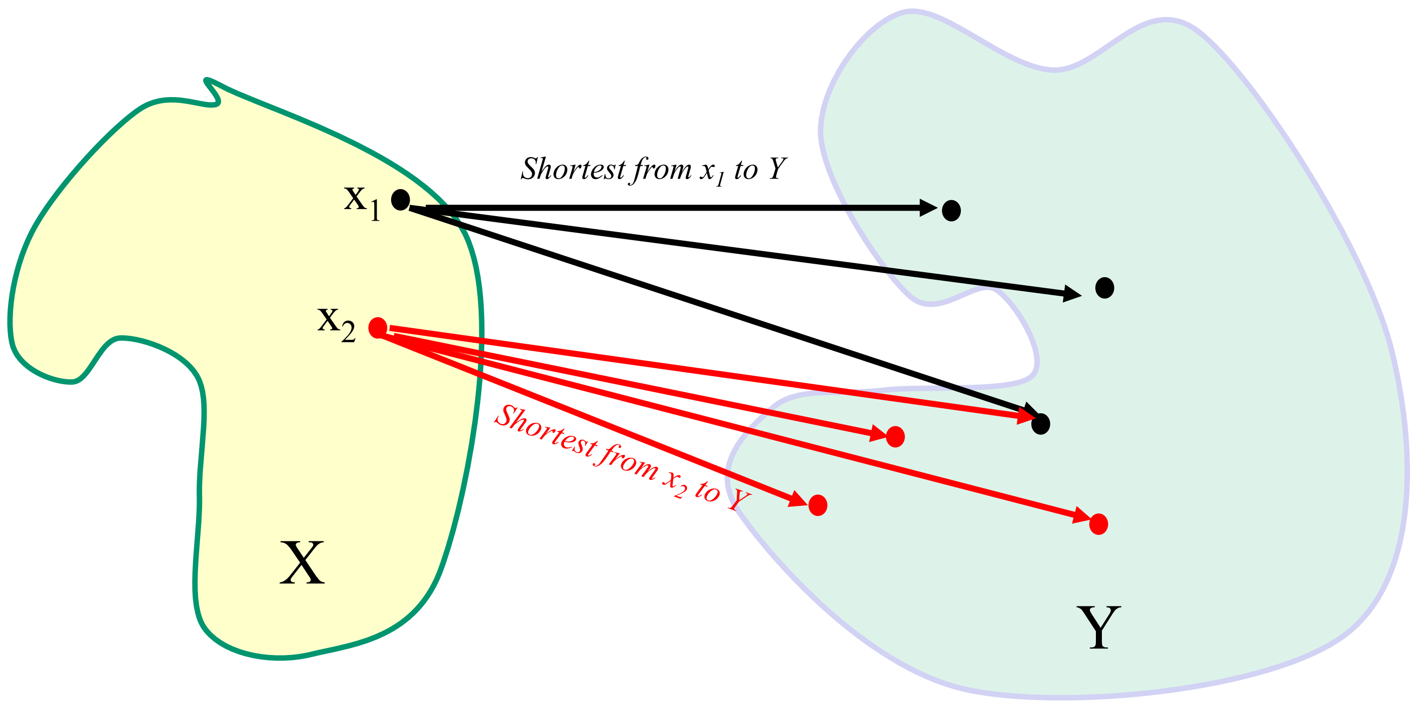
- Hausdorff distance

$$\rho(X,Y) = \max_{x\in X}\left[\min_{y\in Y}\{\rho(x,y)\}\right]$$

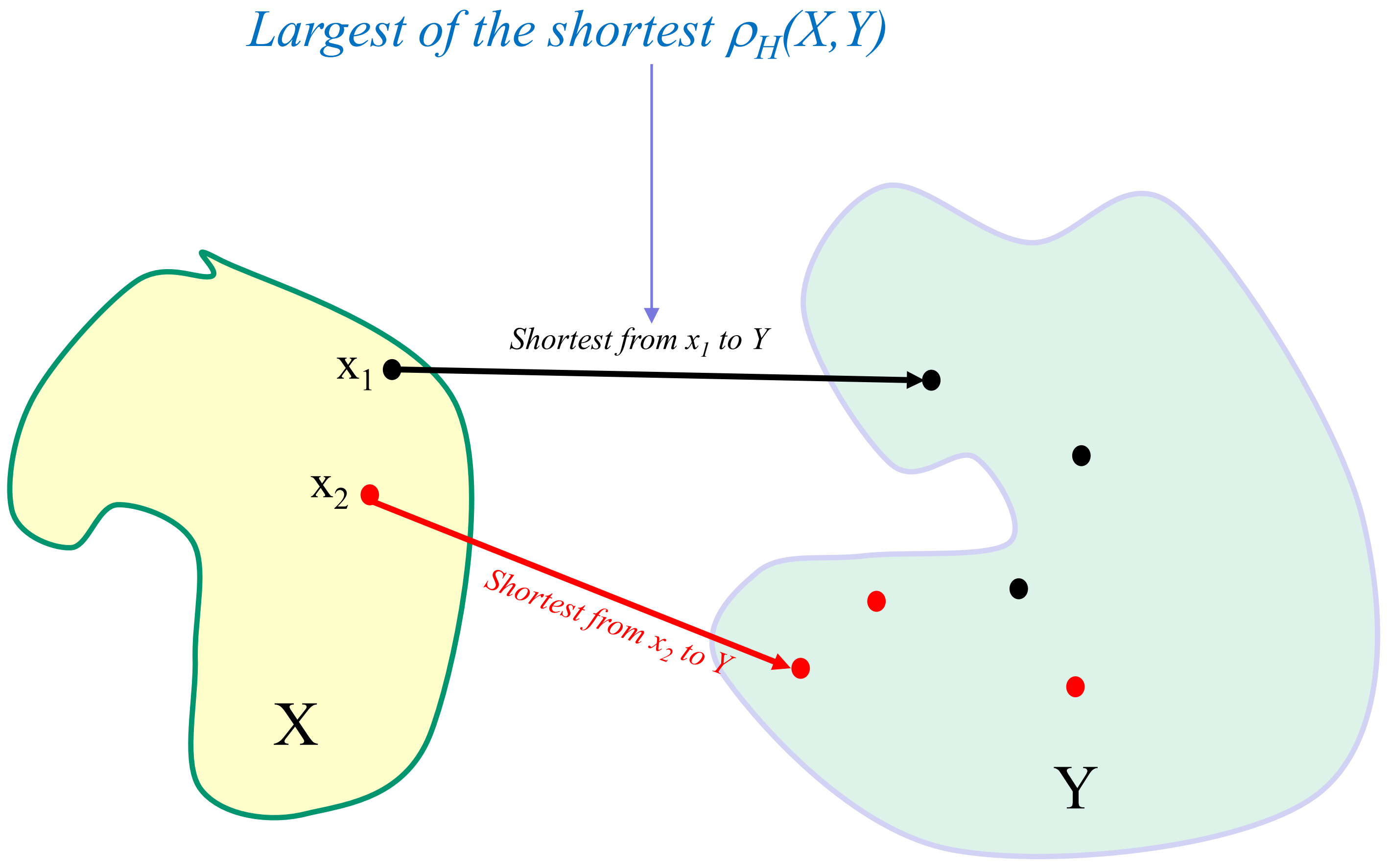
Find the largest of the distances of every point in set X to the nearest point y in Y.

The Hausdorff distance is not a true metric. Need to take max[$\rho_H(X,Y)$, $\rho_H(Y,X)$]

Hausdorff Distance



Hausdorff Distance


Largest of the shortest $\rho_H(X,Y)$
Shortest from x_1 *to* Y
x_1
x_2
Shortest from x_2 *to* Y
X
Y

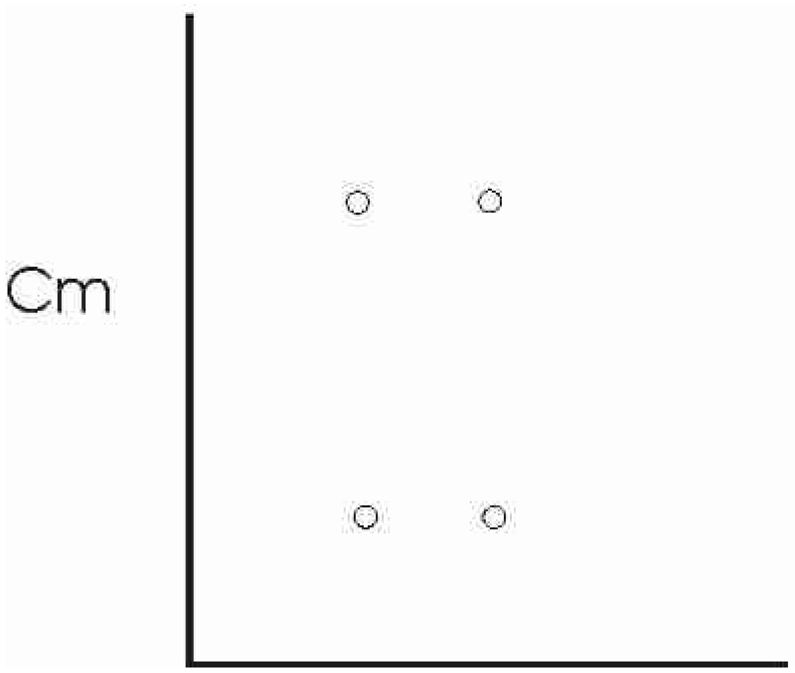
Outliers Variance *vis à vis* Mean Absolute Deviation

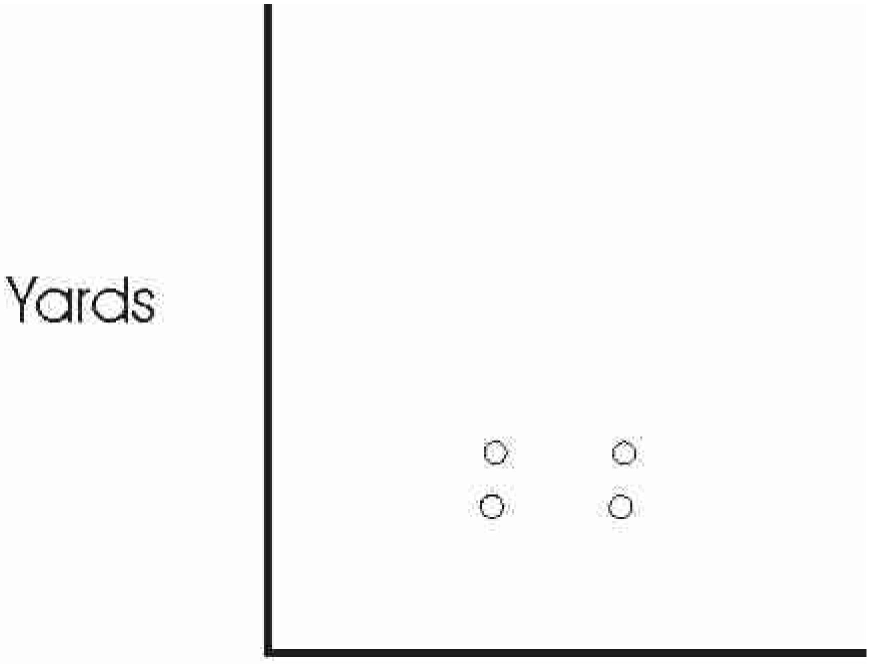
$$\mathrm{var} = \sqrt{\frac{1}{n-1}\left((w-\mu)^2 + (x-\mu)^2 + (y-\mu)^2 + ..\right)}$$

$$MAD = \sqrt{\frac{1}{n}\left(|\,w-\mu\,| + |\,x-\mu\,| + |\,y-\mu\,| + ..\right)}$$

Much less sensitive to outliers

Effect of Scale Units in Clustering





From <u>Finding Data in Groups-An Introduction to Cluster Analysis</u>. Kaufman and Rousseeuw, 1989

Eliminate Scale Effects By Use of Variance

- Define z-score

$$z_{i,j} = \frac{(x_{i,j} - \mu)^2}{s_{i,j}}$$

Confounder: Correlated Data

- When one data point ‘follows’ another under an action (is ***correlated***), there is redundancy in the data. The second data point offers no new information.

- For ideal clustering, we must consider ***data reduction*** when there is correlation of the variables- PCA,ICA

CLUSTERING STRATEGIES

- Bottom up –Hierarchical, Agglomerative

- Top Down- Divisive

Another Viewpoint on Strategy

Data density

vis à vis

Data connectedness

Agglomerative Clustering

- Start with every point, and gather ‘like’ points together step-wise

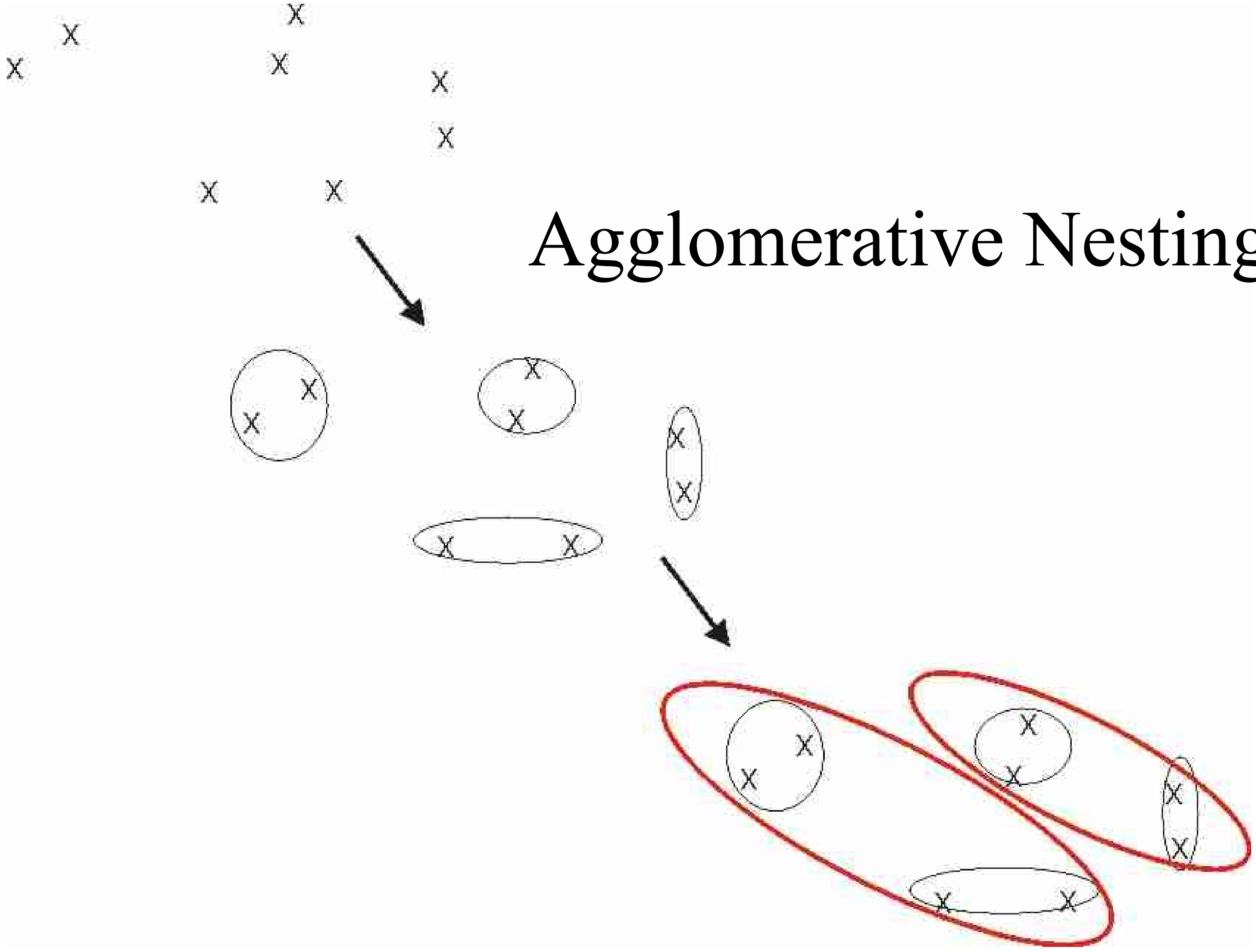
Hierarchical Methods

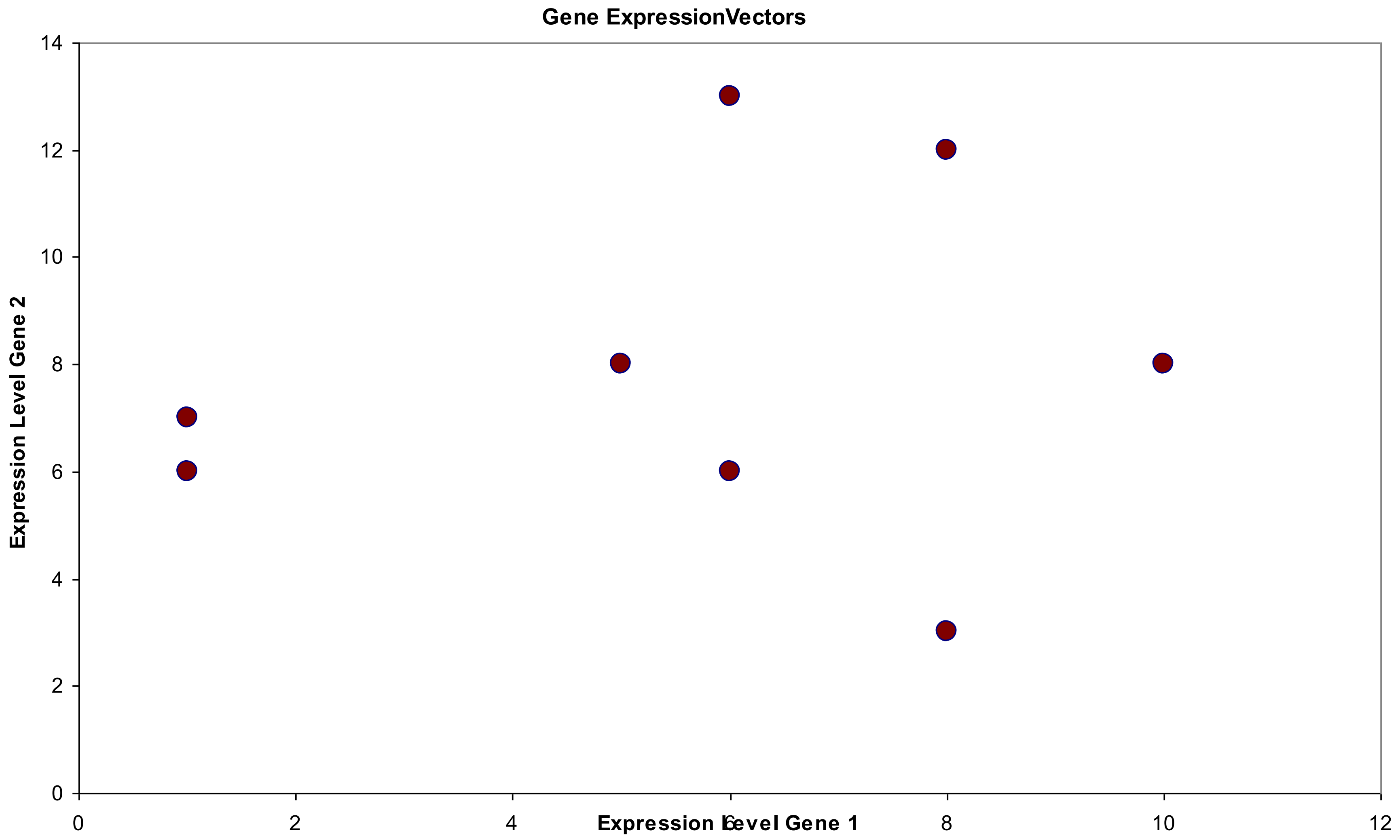
<u>Example</u>

Every point is a cluster

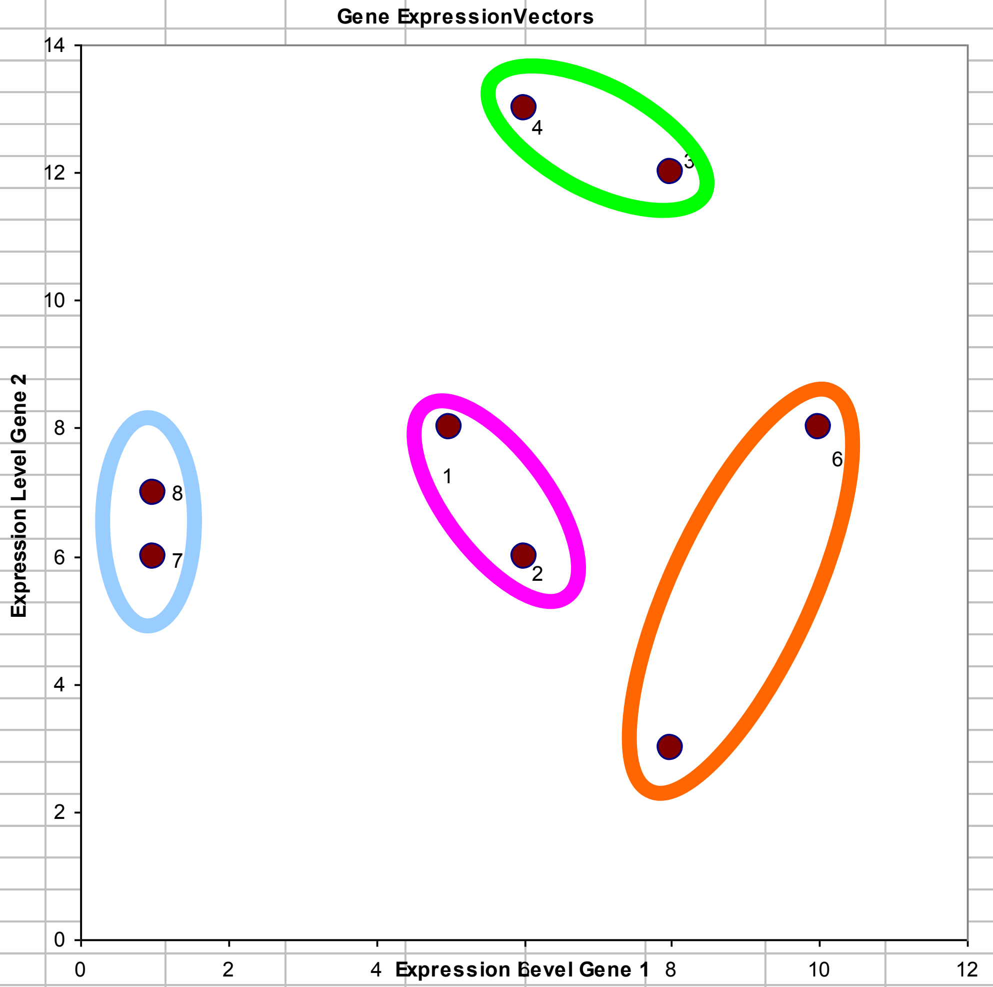
- Pick two closest points and make a new cluster (Nearest Neighbor concept)
- Repeat over the remaining n-1clusters
- Recurse

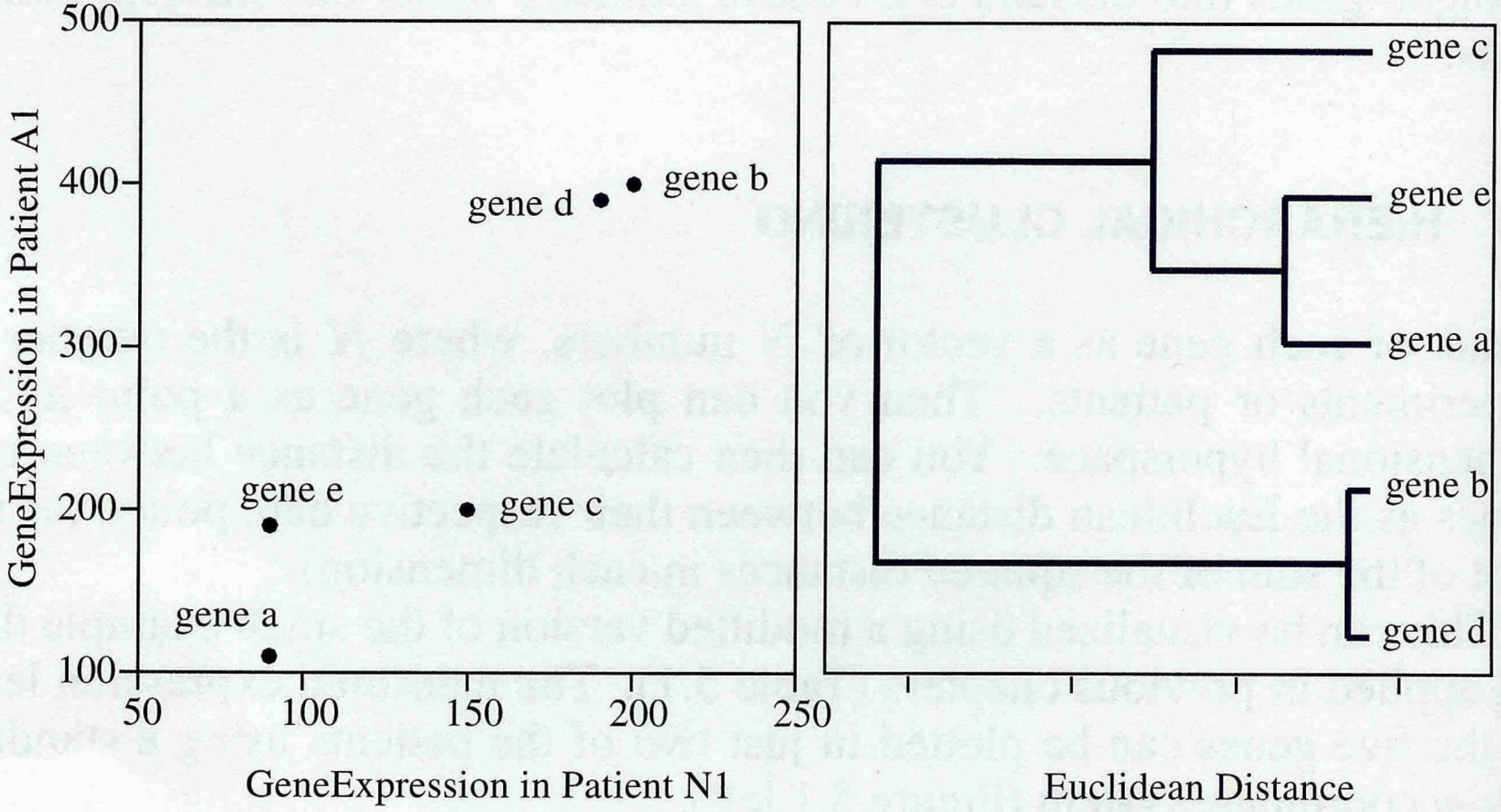
Agglomerative Nesting




Gene ExpressionVectors
Expression Level Gene 2
Expression Level Gene 1
0
2
4
6
8
10
12
14

	1	2	3	4	5	6	7	8
1	0.000							
2	2.236	0.000						
3	5.000	6.325	0.000					
4	5.099	7.000	2.236	0.000				
5	5.831	3.606	9.000	10.198	0.000			
6	5.000	4.472	4.472	6.403	5.385	0.000		
7	4.472	5.000	9.220	8.602	7.616	9.220	0.000	
8	4.123	5.099	8.602	7.810	8.062	9.055	1.000	0.000





From Analysis of DNA Microarray Data, Knudsen,2002

Divisive Clustering

- Start with *all* data points, then partition them according to some rule.

Ways to classify Divisive Clustering

- Hardness
  - Hard Clustering (Partitioning)
    - Each group has at least one element
    - Each element belongs to only one group
  - Soft Clustering (Fuzzy Analysis)
    - Each element has probabilistic membership in one or more clusters

---

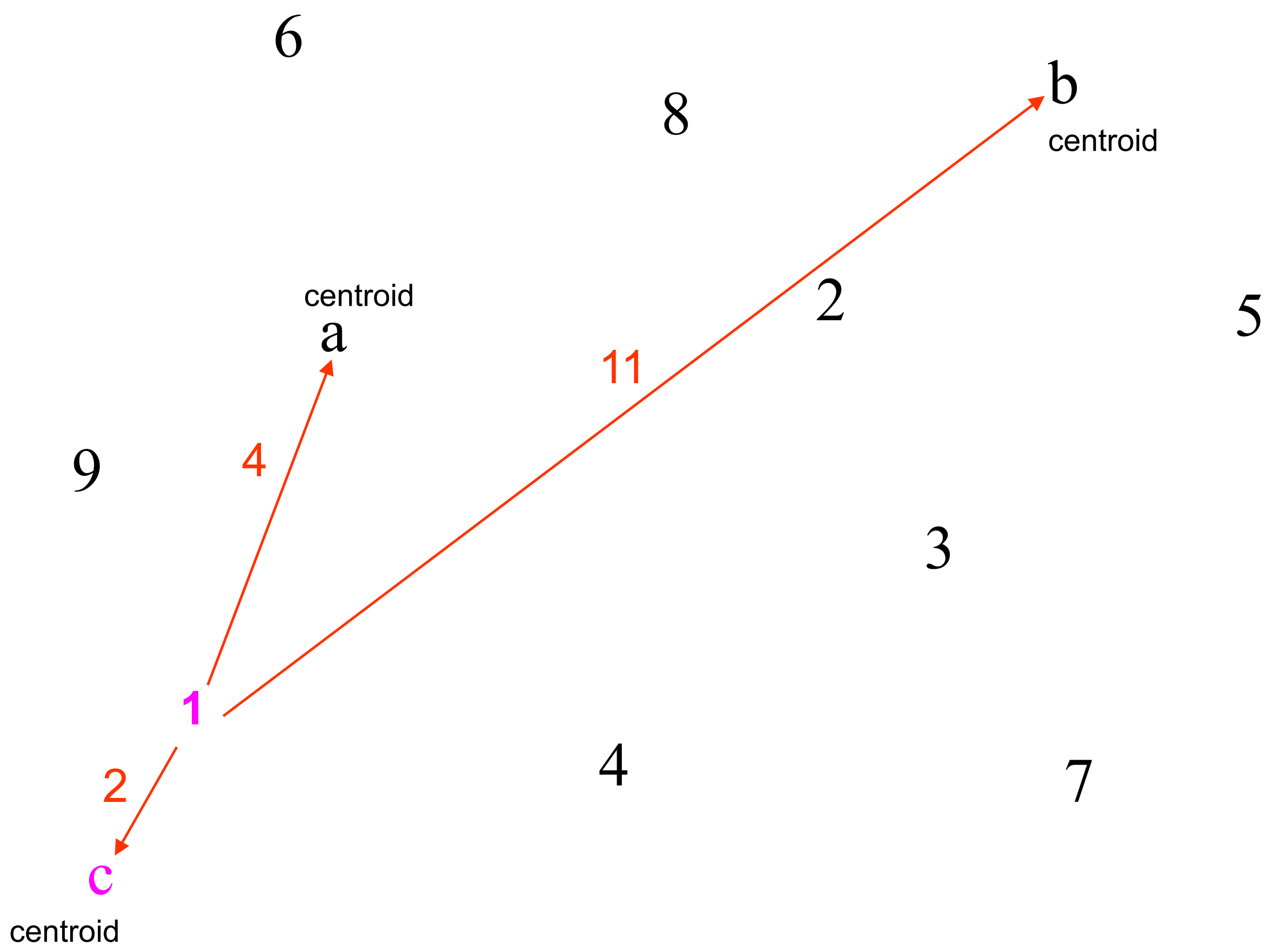
- Number of clusters
  - Fixed
  - Variable
- Representative elements
  - Basis of the algorithm
  - Not part of the algorithm

A Workhorse: The K-Means Algorithm

1. Define k seed centroids (set of coordinates, not a rep. element)
2. Compute f.

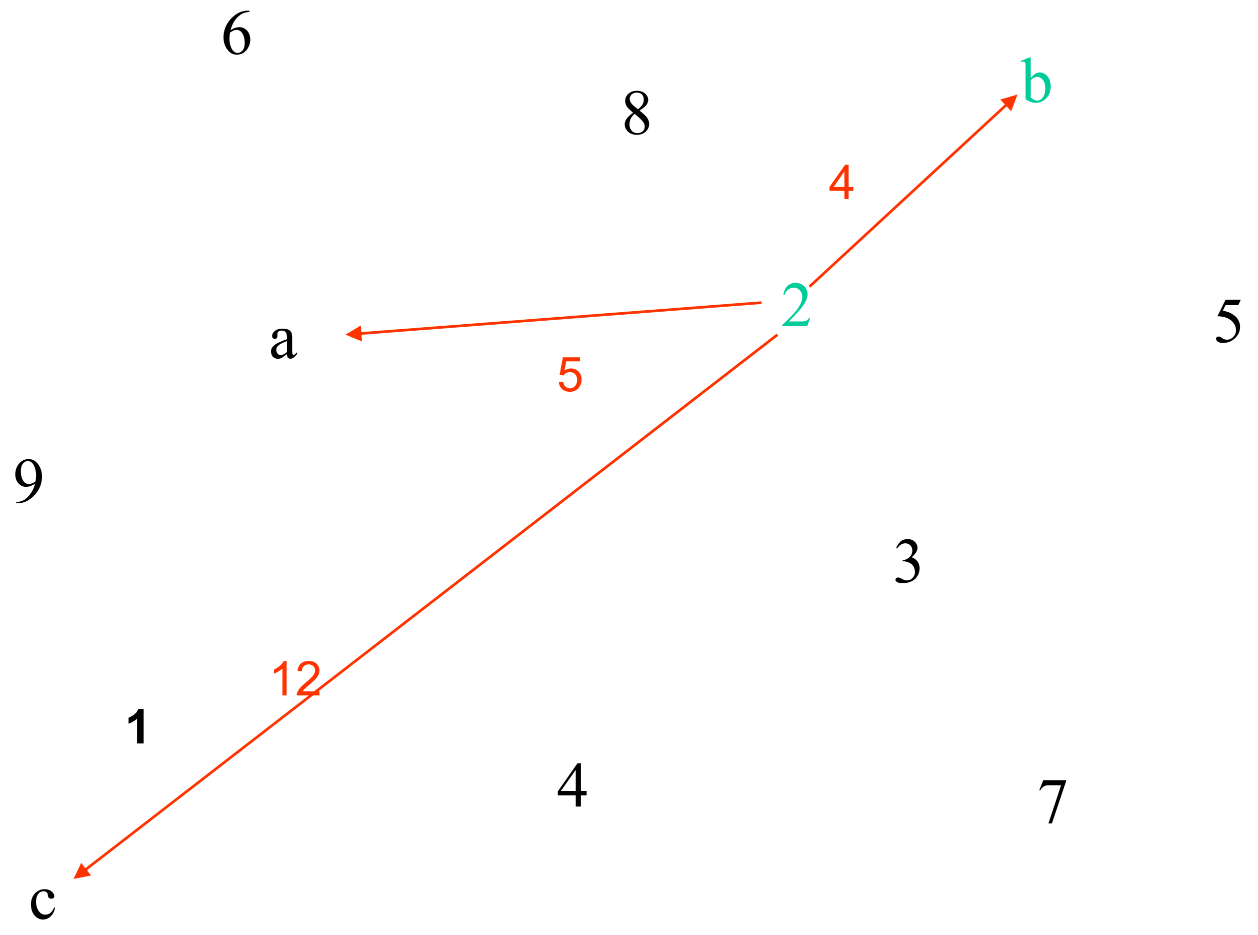
   Cost function f = sum of squares of distance from each element to its centroid)
3. Quit when no change
4. Recalculate centroid based on current elements in cluster
5. Go to 2

k-means



k-Means
Find Closest reference point

	Dist to a	Dist to b	Dist to c
Point 1	4	11	2
Point 2			


6
8
b
4
2
a
5
5
9
3
12
1
4
7
c

k-Means

	Dist to a	Dist to b	Dist to c
Point 1	4	11	2
Point 2	5	4	12

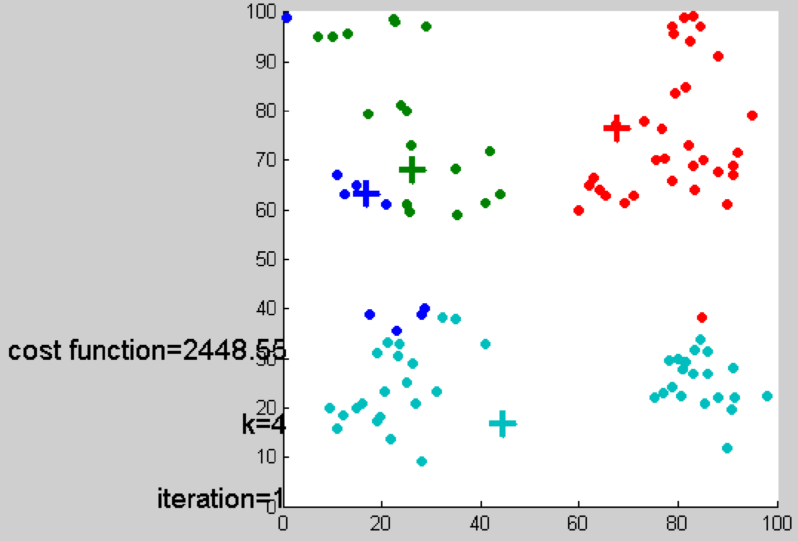
Summary of K-Means Algorithm

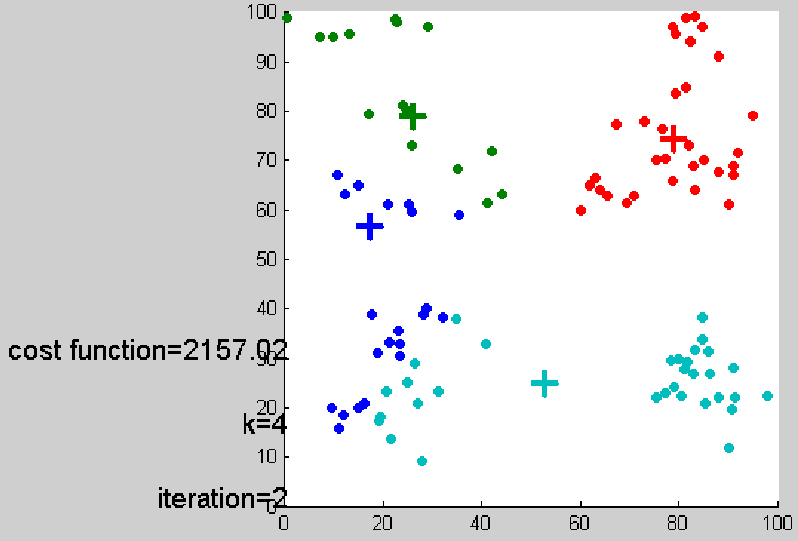
- Label each data point with the name of the species whose centroid the data point is closest to

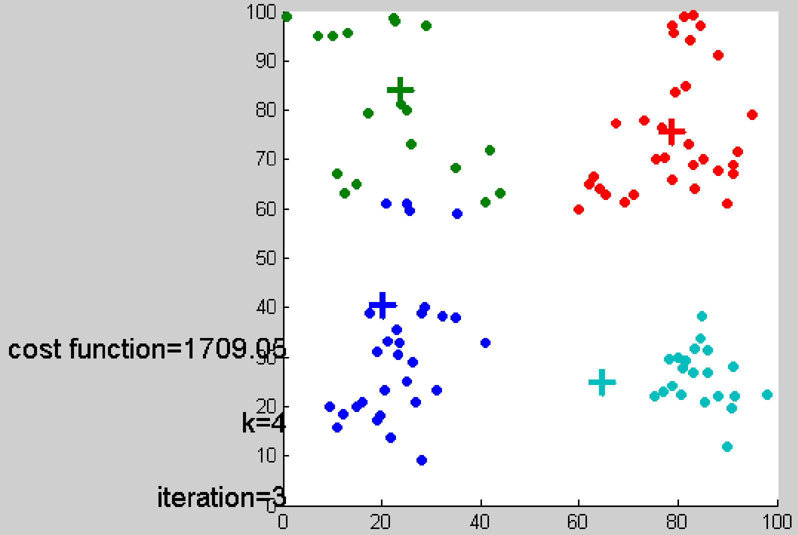
Summary of K-Means Algorithm

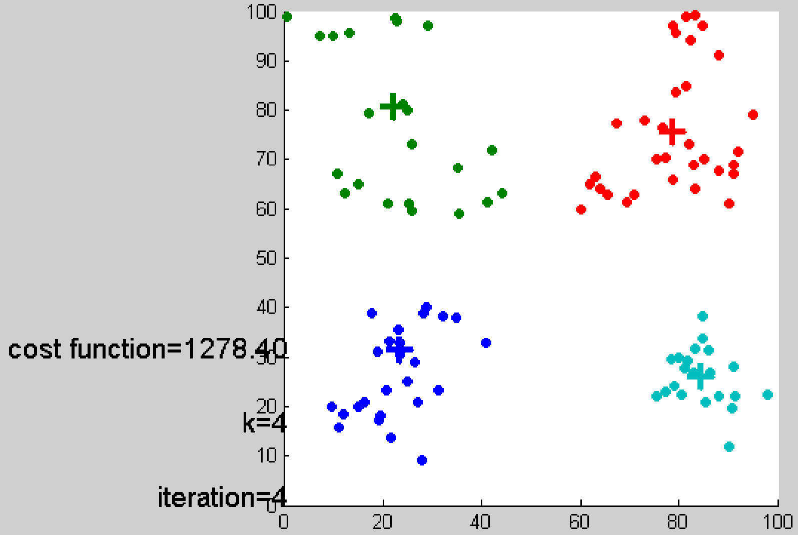
1. For each species, find the sum of squares difference between the member data points and their centroids.
2. Get a cost function by summing the sum of squares of each species
3. Minimize this number
4. This is a greedy algorithm-may have to repeat many times to escape a local minimum

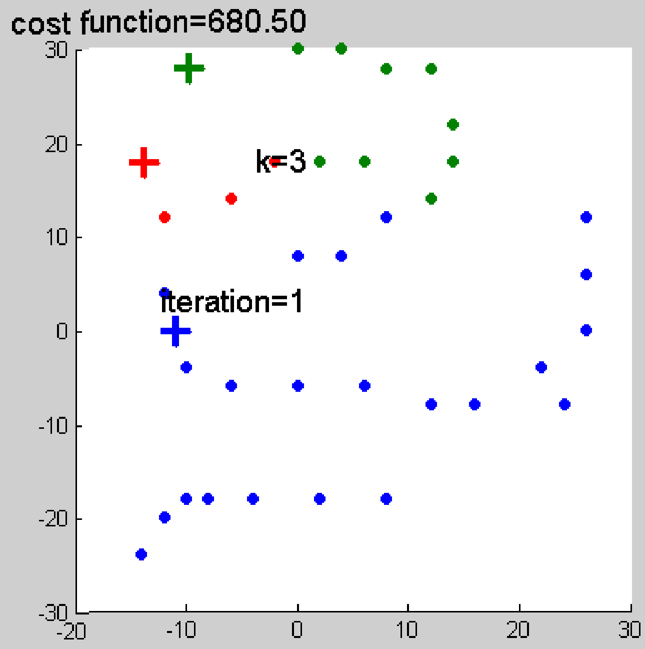
Successive iterations of the k-means algorithm. Each centroid is a large '+' . Starting locations of centroids were randomly selected. Each centroid 's color matches the corresponding element in its cluster

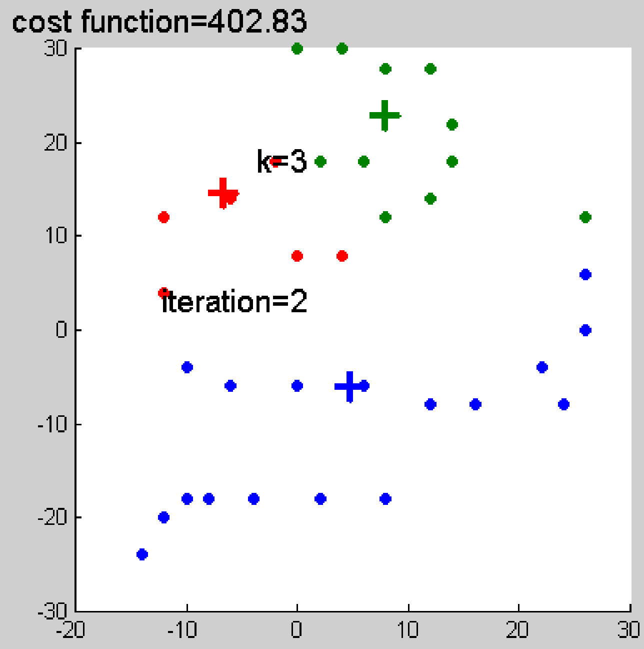




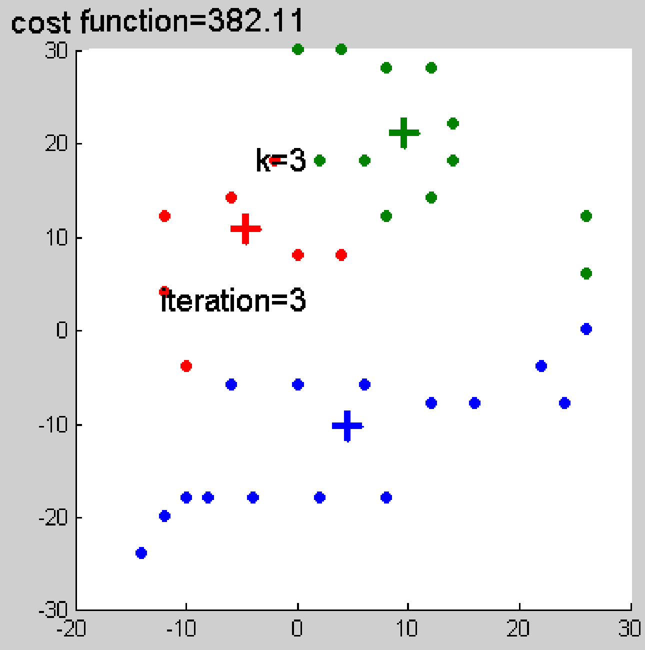


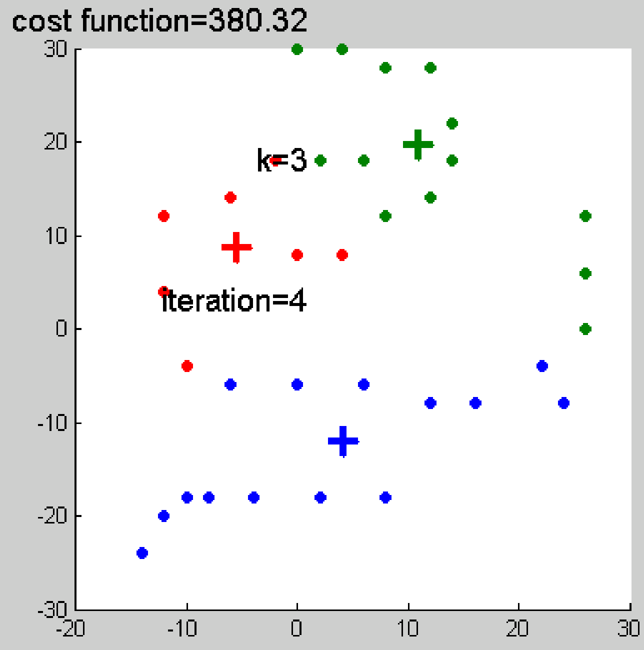


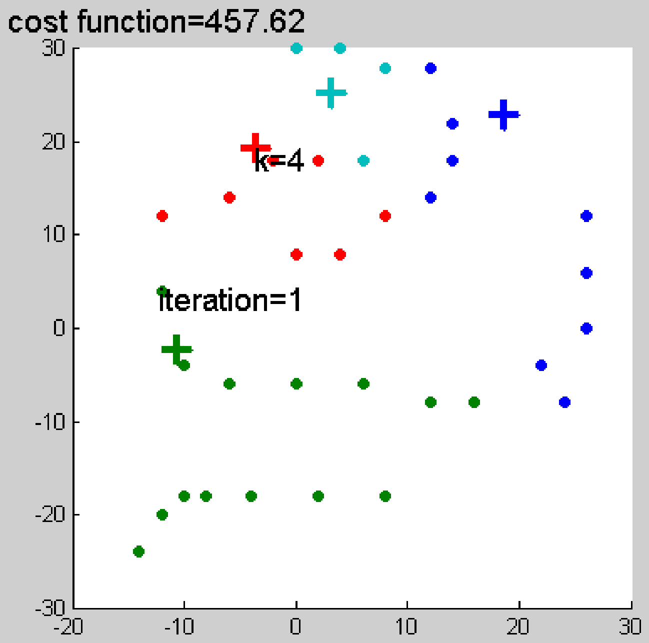




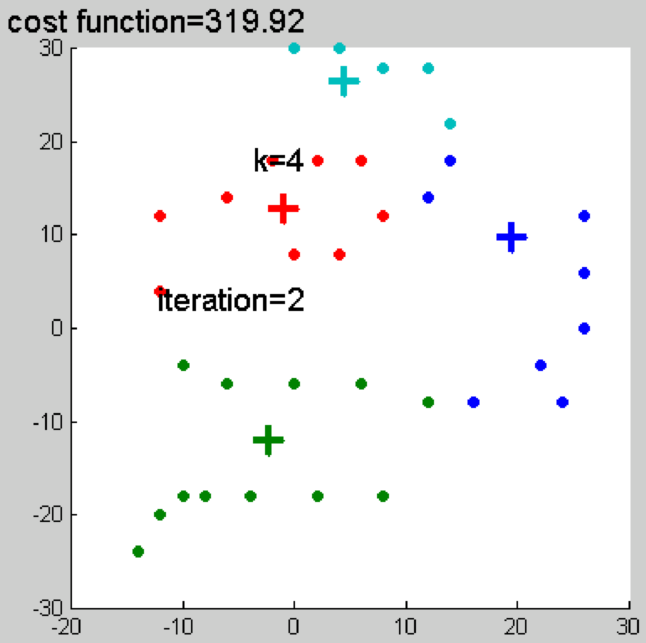
The k-means algorithm performs poorly on non-convex sets

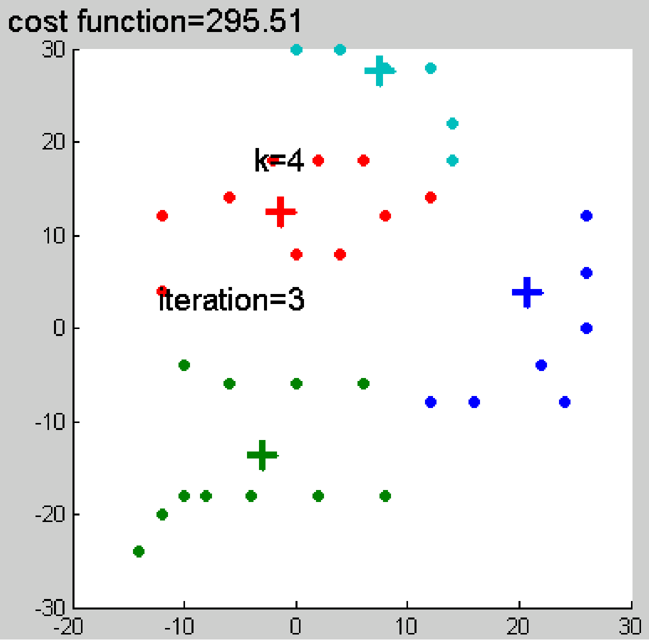




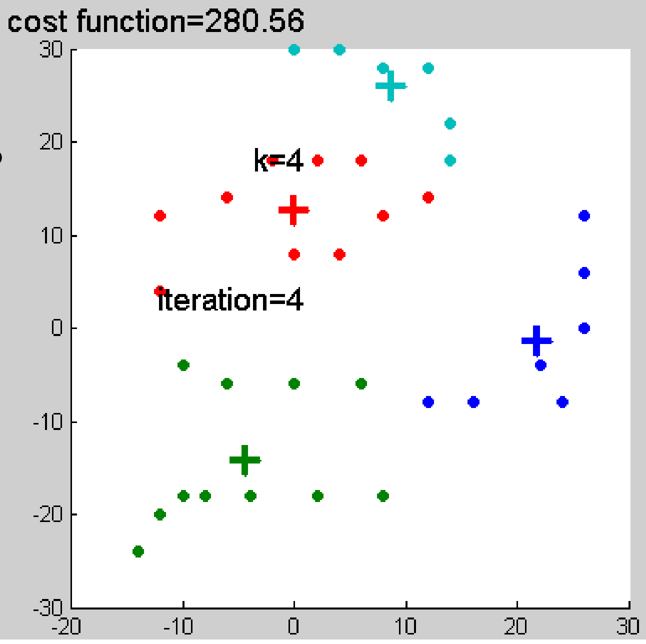


Increasing k perhaps looks better, but the intuitive notion is that there are only 3 clusters



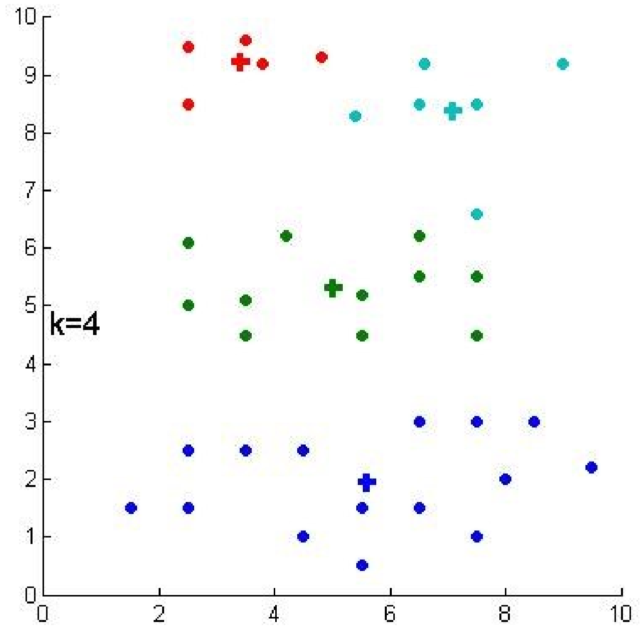


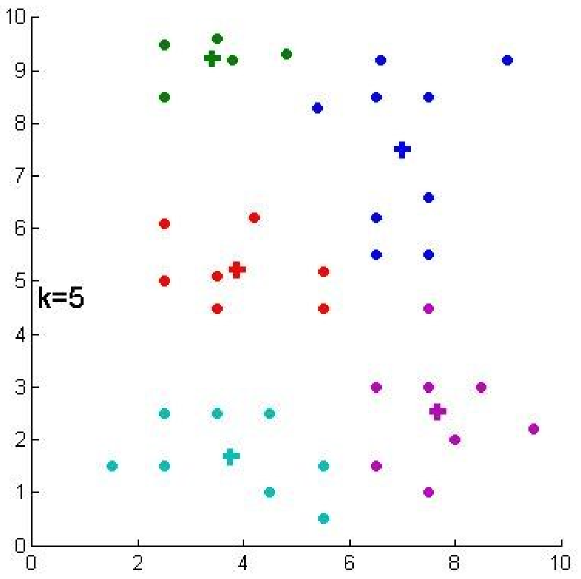
Remember, cost functions are not comparable across clusterings with differing k's, so this cost of 281 is better than the 380 in the previous (k=3) clustering, but it is not a legitimate comparison



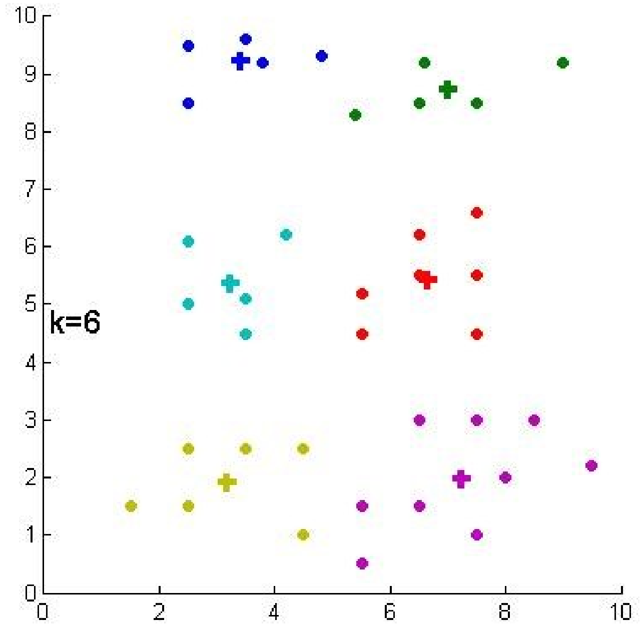
How well did we do?

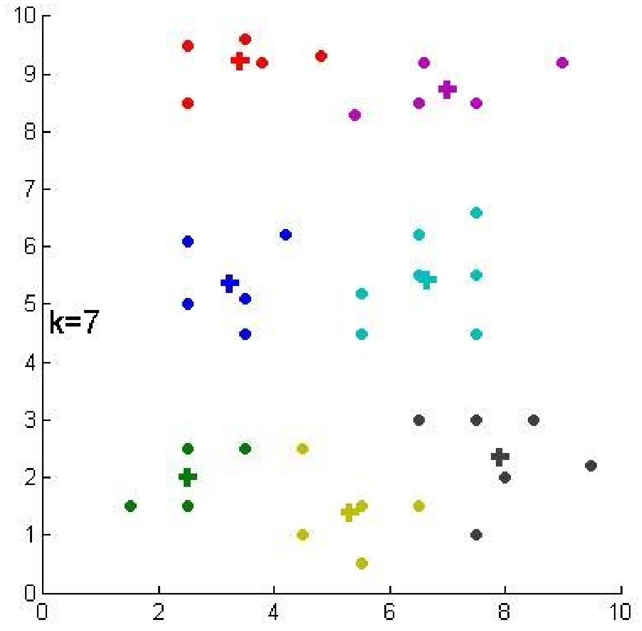
Measures of validity <u>must</u> take into account the number of clusters, k. It is easy to make the clustering look good with a very large k. The trick is finding the correct k, if indeed any specific k is correct.





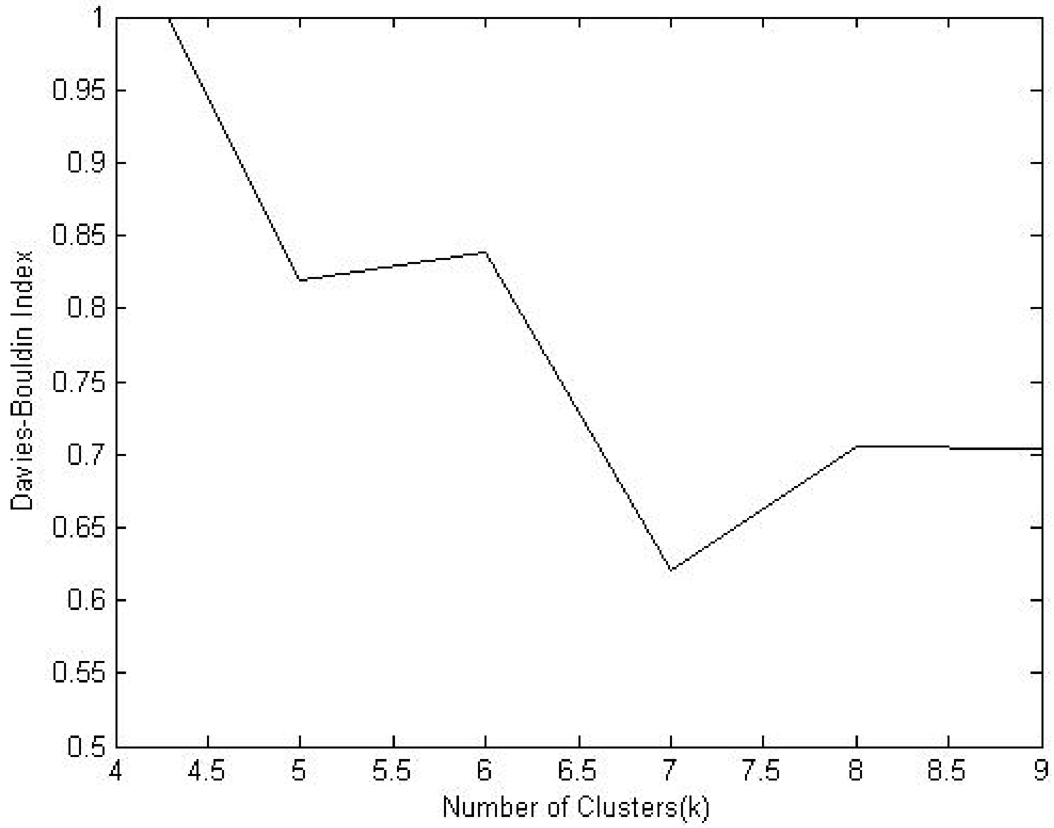
Same data- different k's





Cluster Validity

- The Davies-Bouldin Index accounts for the number of clusters



Here the Davies-Bouldin Indices for the clustering problem in the previous slide are plotted for k 4 to 9. The optimum (minimum index value) for this problem corresponds to k=7

Model-Based Clustering
Using EM to find the most probable cluster

The concept

- Create a model by predetermining
  - How many clusters
  - How the data might be distributed within a cluster (typically Gaussian or Poisson)
- Determine the most probable assignment of data elements to clusters

For instance…

Suppose we predetermined that there would be two clusters, A and B, and the distribution of data within the clusters would be Gaussian

For each data point, we need to know $\mu_1, \sigma^2{}_1, \mu_2, \sigma^2{}_2$, $p \in$ A, $p \in$ B

(although with just 2 groups we can figure out $p \in$ B)

Hidden variables

- We could guess at the means (make it easy and set $\sigma^2=1$ for both clusters), then see what the $p \in \mathrm{A}$, $p \in \mathrm{B}$ were.

- We could then maximize the likelihood of the data given the parameters

But this is old hat…

This is precisely the Gaussian mixture problem we encountered when introducing the concept of EM in this course.

We solved this (at least to a local minimum) by an iterative process of calculating the expectation of the data, given the parameters, and maximizing the likelihood of the parameters, given the data

To recapitulate…..

- Guess the parameter
- Expectation: Find the expected data, given the parameter
- Maximization: Find the most likely parameter, given the data, by finding the sample mean
- Repeat and converge to a solution for μ_{ML}

Kicking it up a notch..

Two new issues

1. More than 2 distributions (clusters)
   - Not really a problem; we wrote an algorithm for k Gaussian mixtures
2. More than 1 dimension
   - Consider each dimension a variable in a multivariate problem
     - If the variables (dimensions) are independent, not a problem; consider their product as a single variable (dimension)
     - If they are not independent, it gets messy

Consider the Gaussian pdf in one dimension

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2}\left(\frac{x-\overline{x}}{\sigma}\right)^2}$$

The term for variance in the exponent can be re-written (using the notation μ to represent the mean)

$$\left(\frac{x-\overline{x}}{\sigma}\right)^2 = (x-\mu)\left(\sigma^2\right)^{-1}(x-\mu)$$

Now consider the Gaussian pdf in many dimensions

Values for x and μ become vector-valued, and the term for variance becomes a matrix for co-variance

$$(\vec{x}-\vec{\mu}) \;\angle\; (\vec{x}-\vec{\mu})$$

And finally, the univariate normalizing constant

$$(2\pi)^{-\frac{1}{2}}\left(\sigma^2\right)^{-\frac{1}{2}}$$

must be changed to a more general constant that makes the volume under the surface of the multivariate density function unity for some *p*. The new constant is

$$(2\pi)^{-\frac{p}{2}}\,|\Sigma|^{-\frac{1}{2}}$$

where $|\Sigma|$ is the determinant of the covariance matrix

So here is the multivariate density that
we must use to maximize the data

$$f(\vec{x}) = (2\pi)^{-\frac{p}{2}} \left|\Sigma\right|^{-\frac{1}{2}} e^{(\vec{x}-\vec{\mu})^T \Sigma^{-1} (\vec{x}-\vec{\mu})}$$

We would, of course, need to generalize the Poisson, or any other cluster density for that matter, in a similar fashion

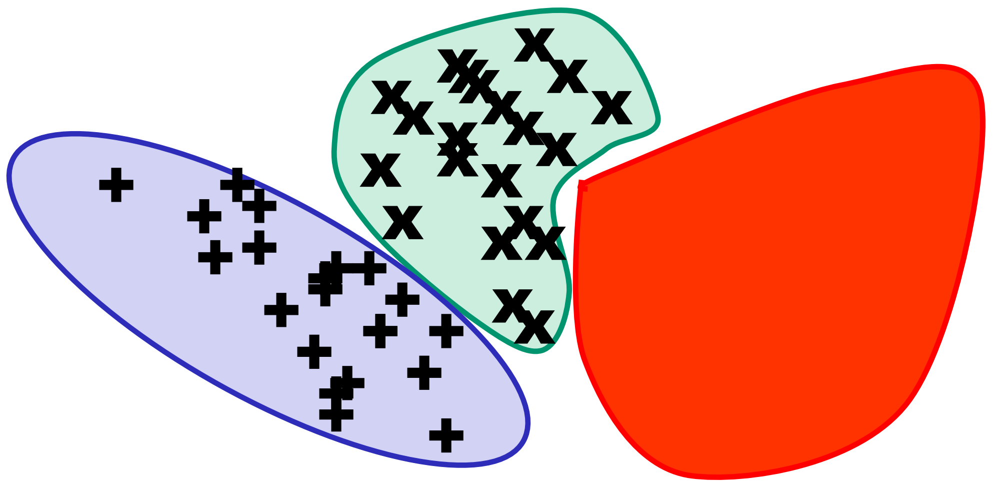
Disadvantages of a Model-Based Clustering

- Like all EM algorithms, local minima are possible
- Sensitive to starting positions of the centroids
- Very sensitive to k
- Clusters identified are all convex
- The algorithm in higher dimensions is difficult to implement

Advantage

- The clusters found need not be simply connected

Changing topology…

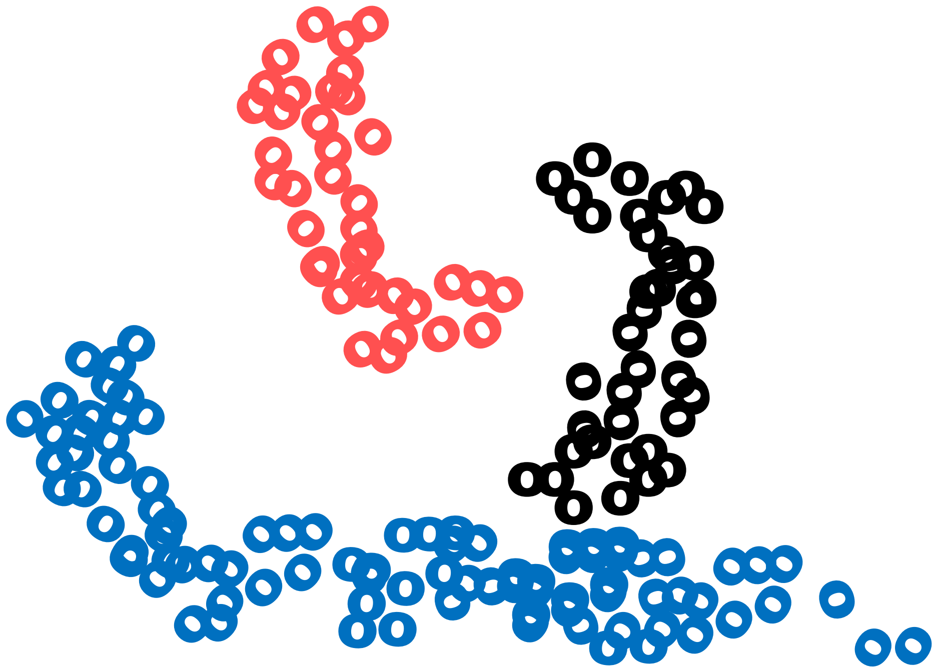


When data are brought closer than the minimum radial distance of all sets, then ***the sets can no longer be convex yet retain separability***. Here the purple and pink sets are still convex, but the green set cannot be convex and, at the same time, separate the data.

Still, while it is true that they are not all convex, each set is **simply connected** however. That fact admits other clustering schemes

Here there are 3 easily identifiable clusters, but because of non-convexity, they do not lend themselves to centroid –referenced or model-referenced strategies, particularly as the clusters draw close together.

Other schemes that are motivated by the topology suggested by the data, rather than pure distance, are required. An idea is to base schemes on nearest-neighbor relationships or topology-learning because the representative sets are simply connected.



Unknown k: Learning (unsupervised) algorithms

Self Organizing Map

(Kohonen Map)

Kohonen Map

- Present a set of data points to a set of learning vectors (neurons) arrayed in a specific geometry
  - The dimensionality of the learning vector corresponds to the number of features
- Each neuron will learn the natural clustering based on
  - Distance from data point
  - Number of iterations (becomes less reactive in time)

Special ANN

Competitive dynamics

- Self excitation
- Neighbor suppression

Self-Organizing Map (Kohonen Map)

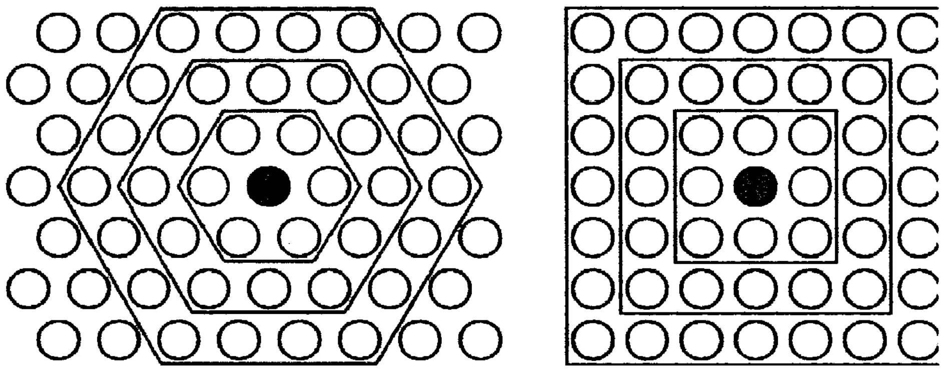
- Select a geometry of learning neurons onto which is superimposed the data elements
- Train each neuron to assimilate features of the nearest data element
- Train adjacent neurons to adapt those features partially

Algorithm

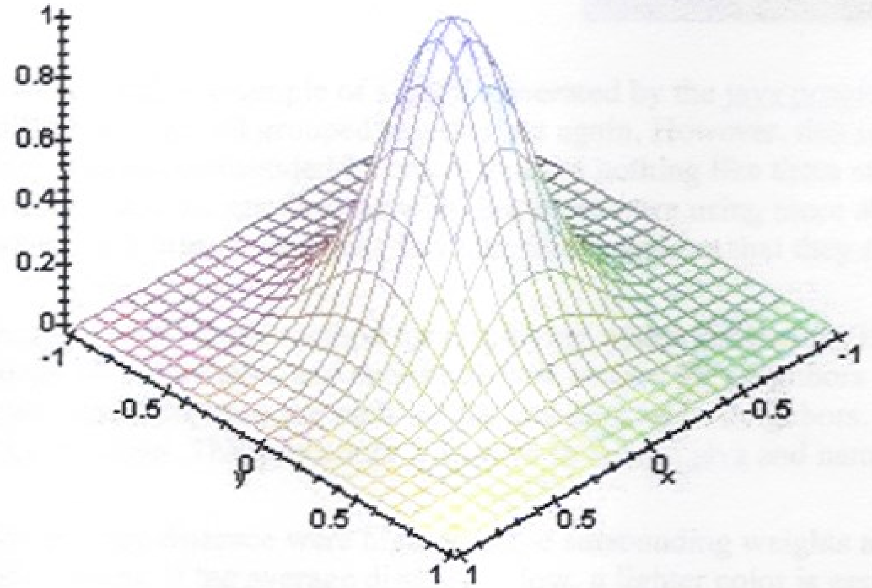
- Present a sample
- Find the weight vector ‘nearest’ to the sample
- Modify* the weight vector to be ‘like’ the sample
- Modify** neighbors likewise
- Present another sample and repeat

*Depends on time elapsed (iteration)

**Depends on distance from the index weight vector

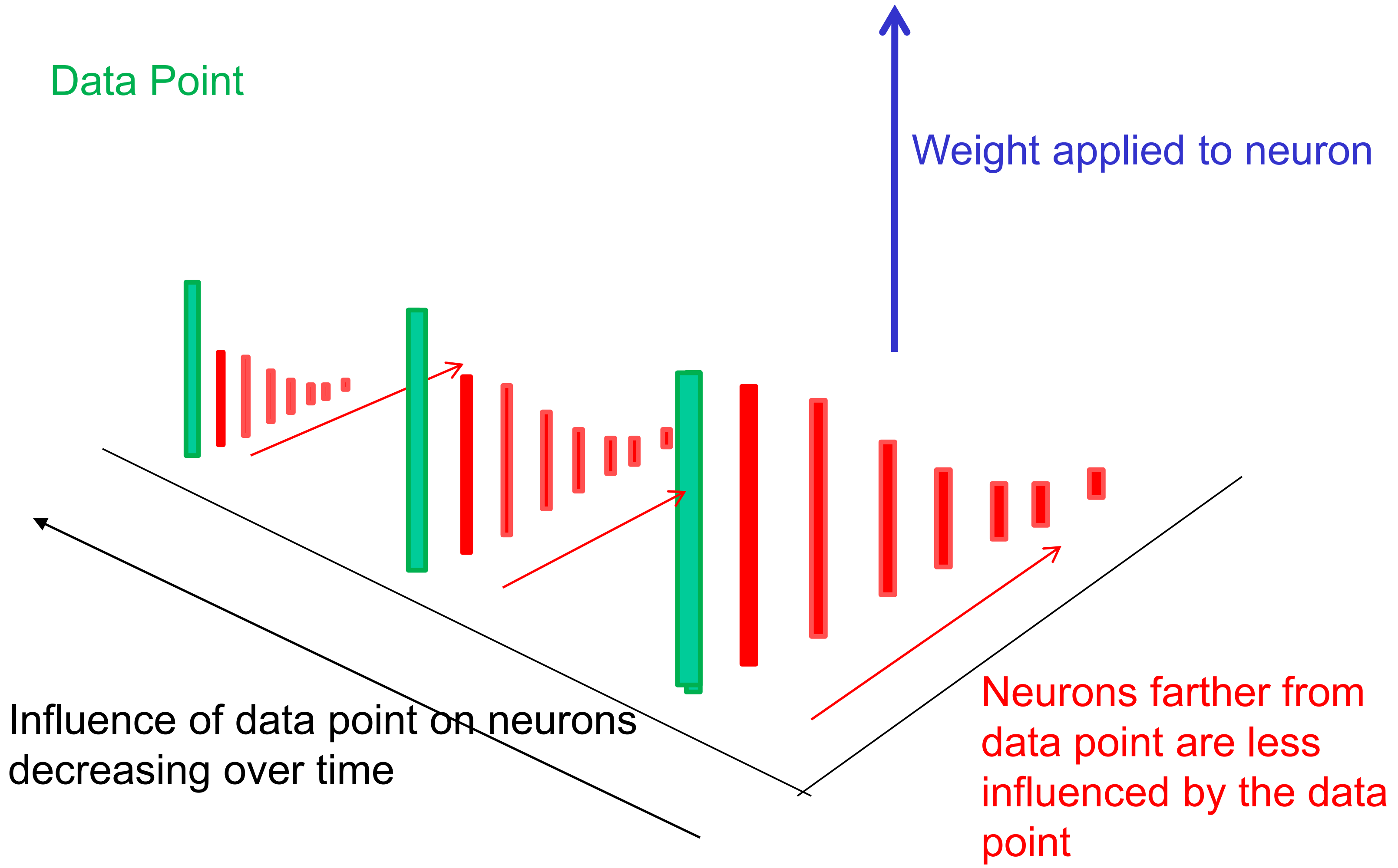


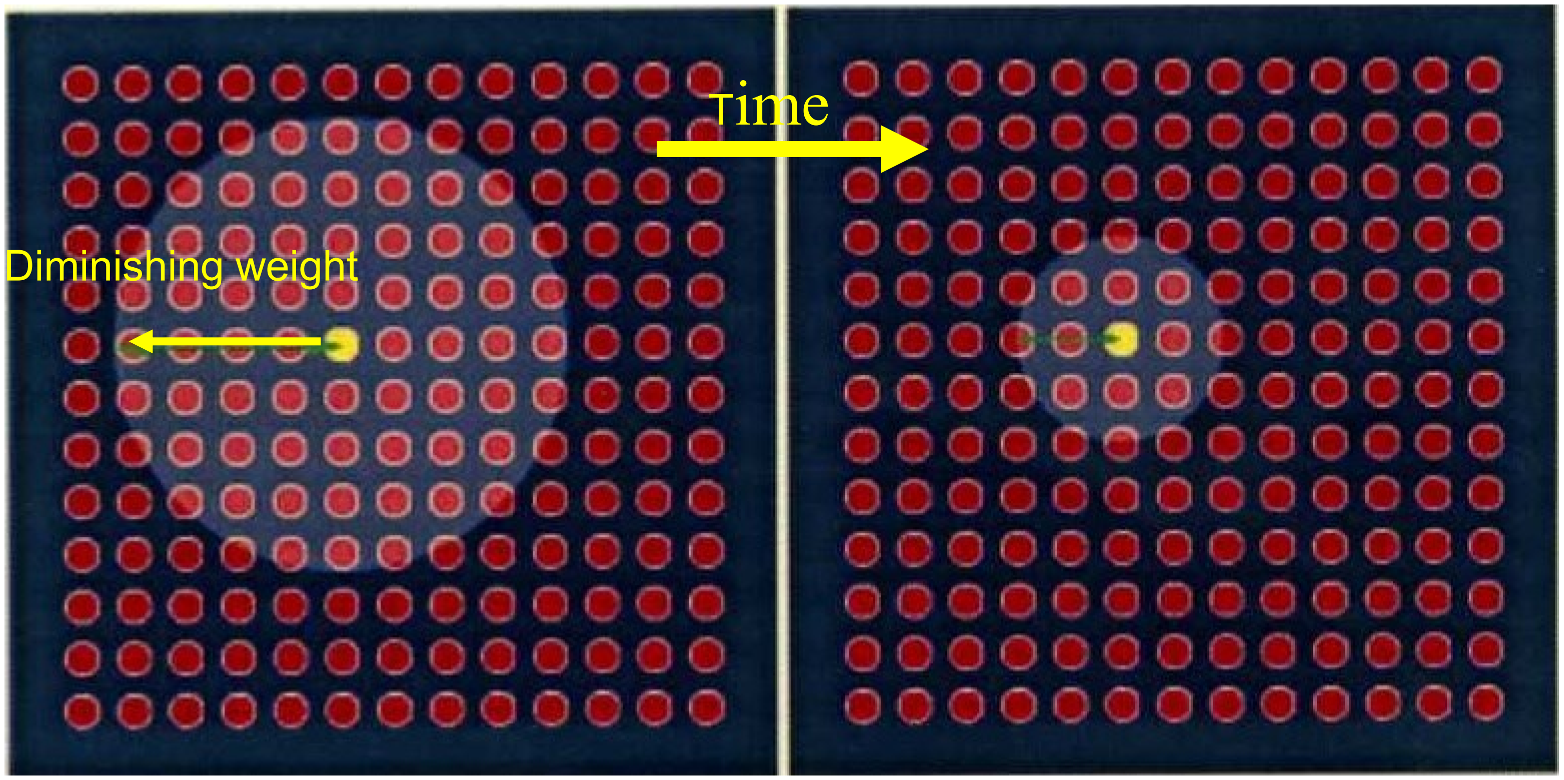
Determination of Neighbors - 2 schemes



$$f := (x, y) \rightarrow \mathrm{e}^{\left(-6.666666667\sqrt{x^2+y^2}^{\,2}\right)}$$

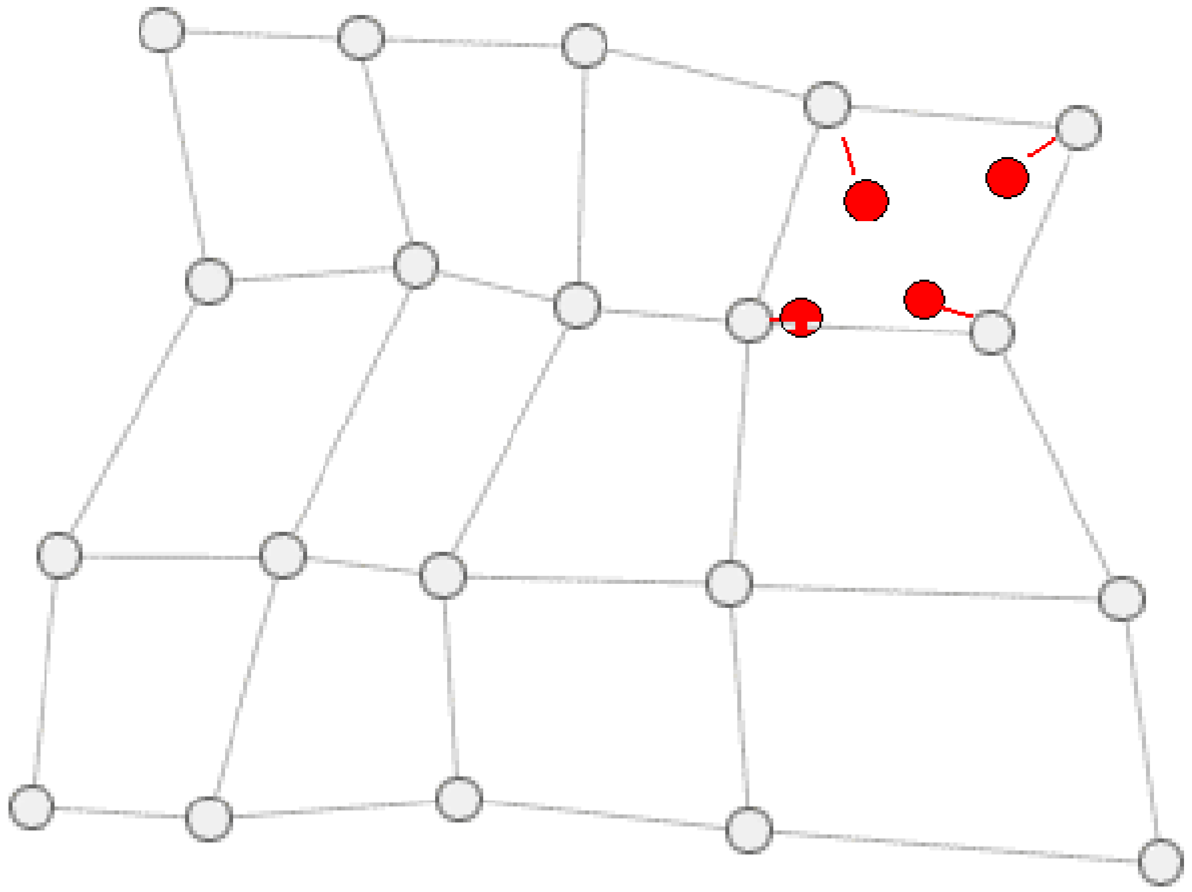
Determination of Neighbors – another scheme


Data Point
Weight applied to neuron
Influence of data point on neurons decreasing over time
Neurons farther from data point are less influenced by the data point



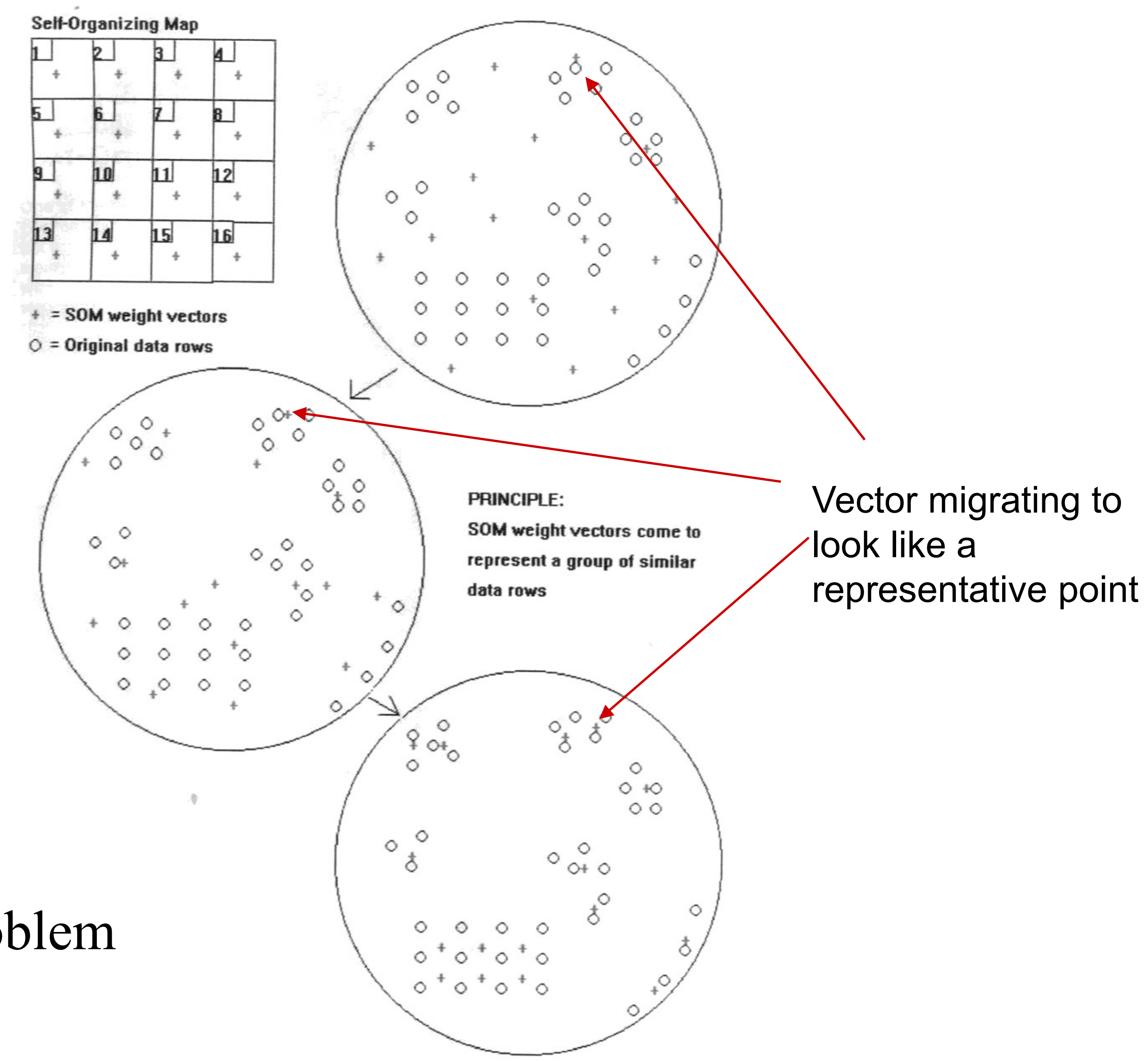
Neighborhood influence diminishes over distance

Neighborhood radius diminishes over time

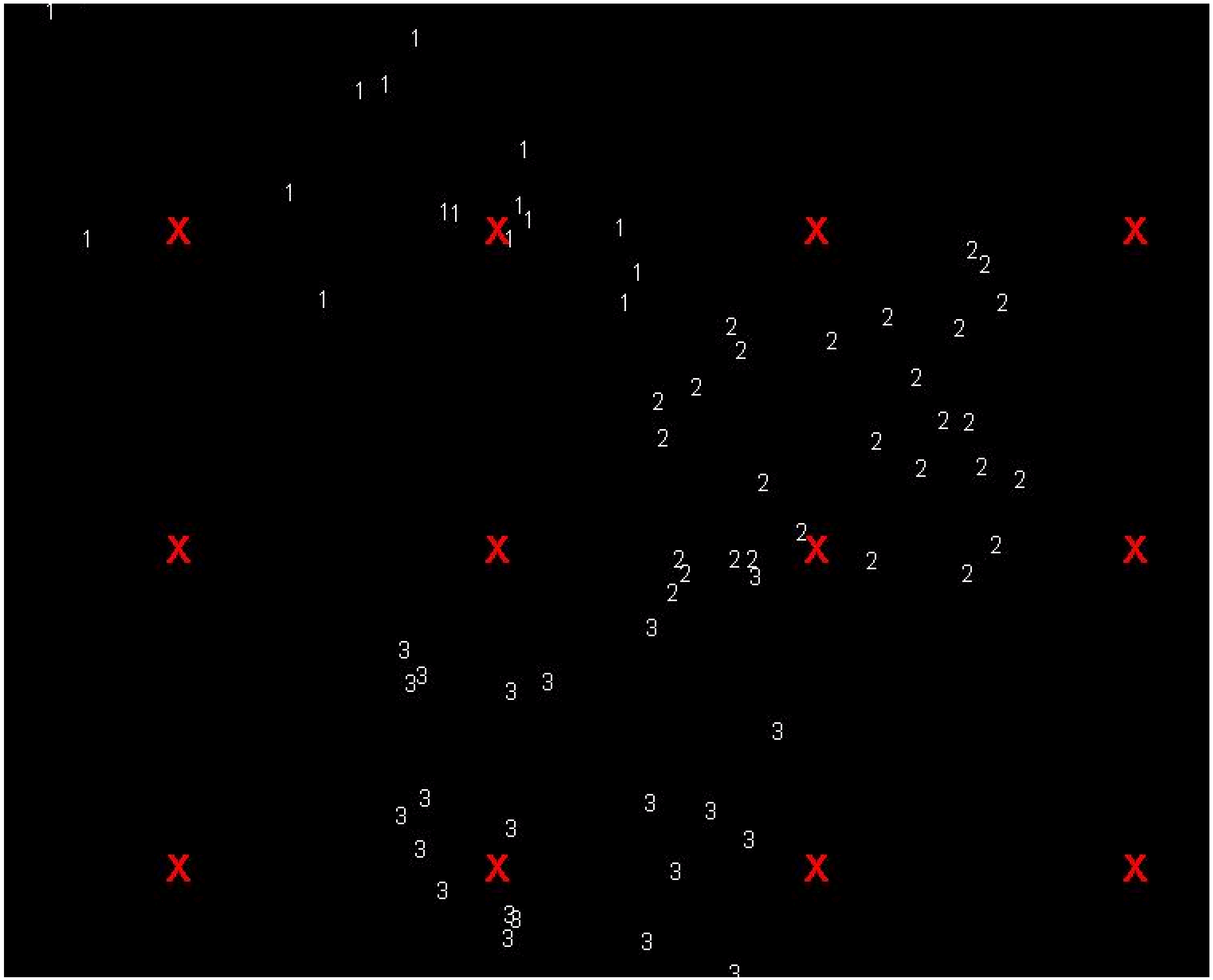


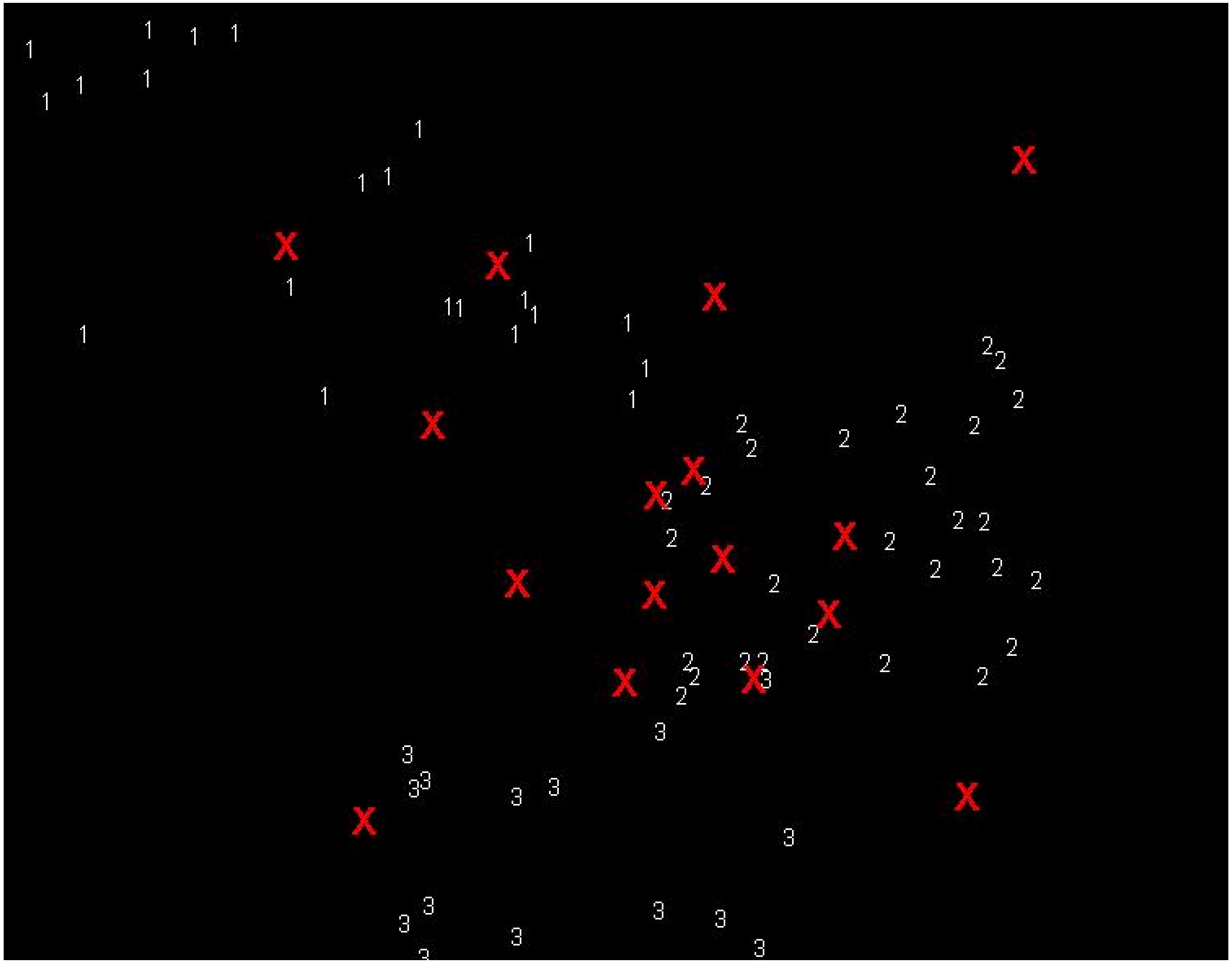
LEARNING

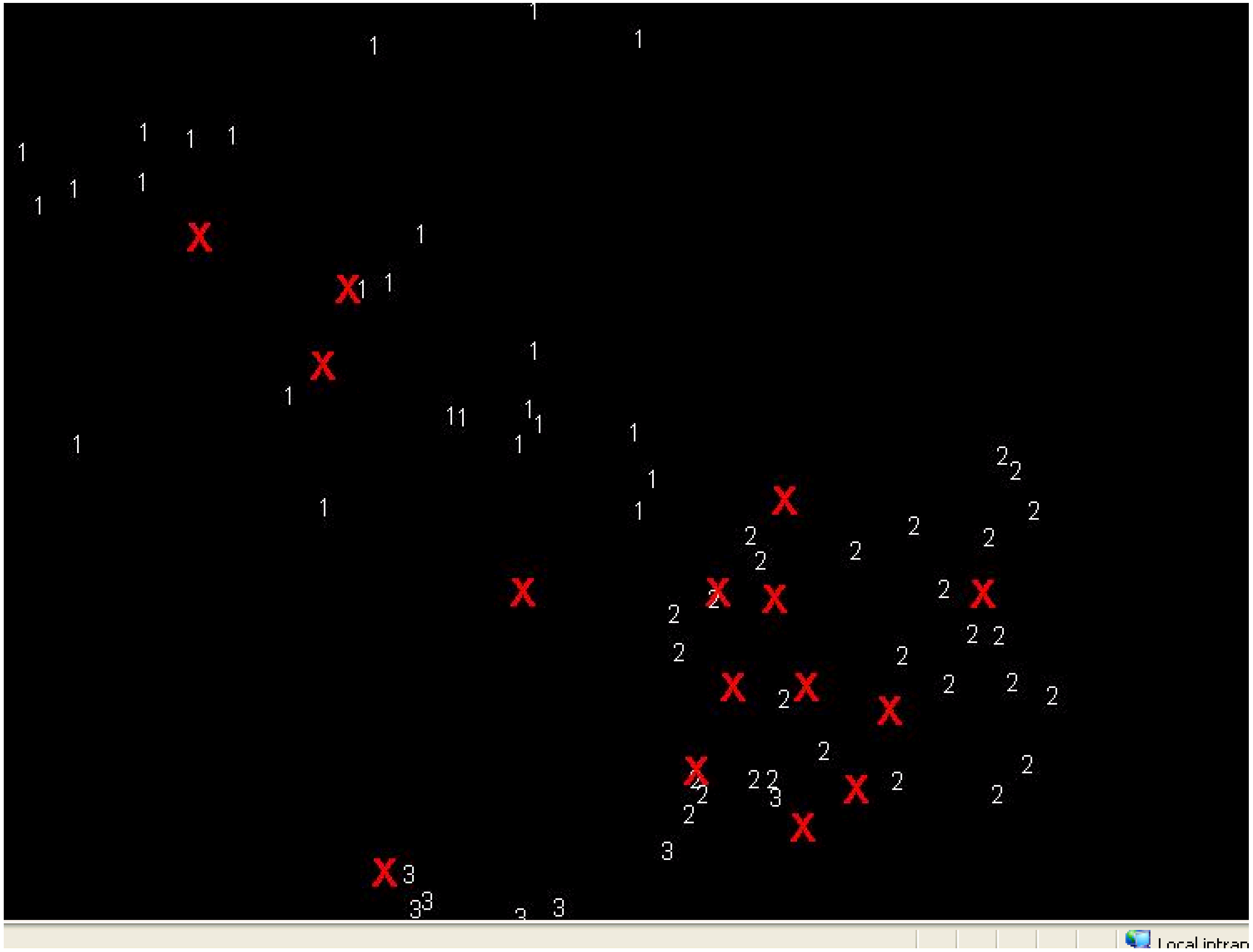
Array of weight vectors

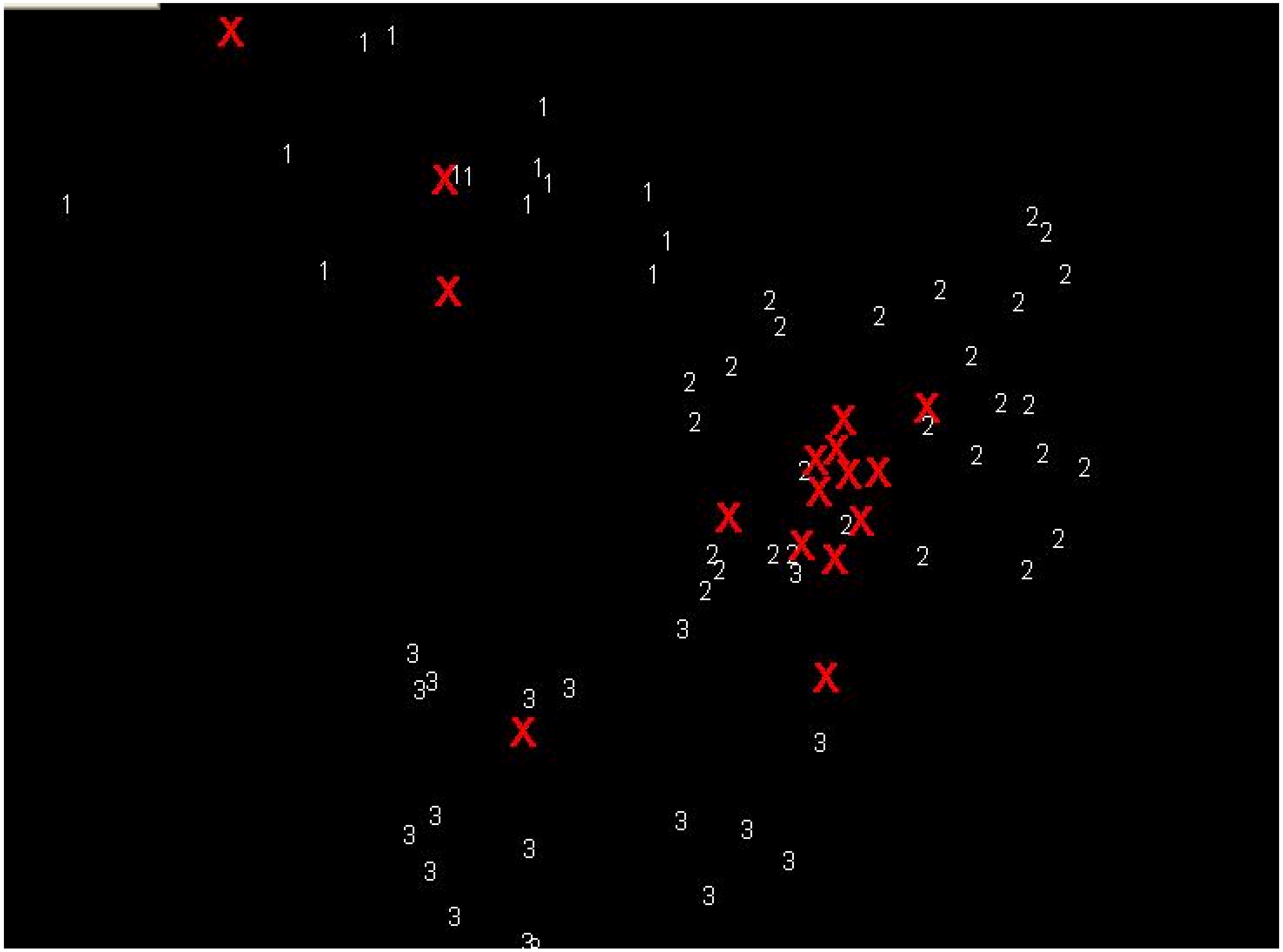

Self-Organizing Map
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
+ = SOM weight vectors
○ = Original data rows
PRINCIPLE:
SOM weight vectors come to represent a group of similar data rows
Vector migrating to look like a representative point

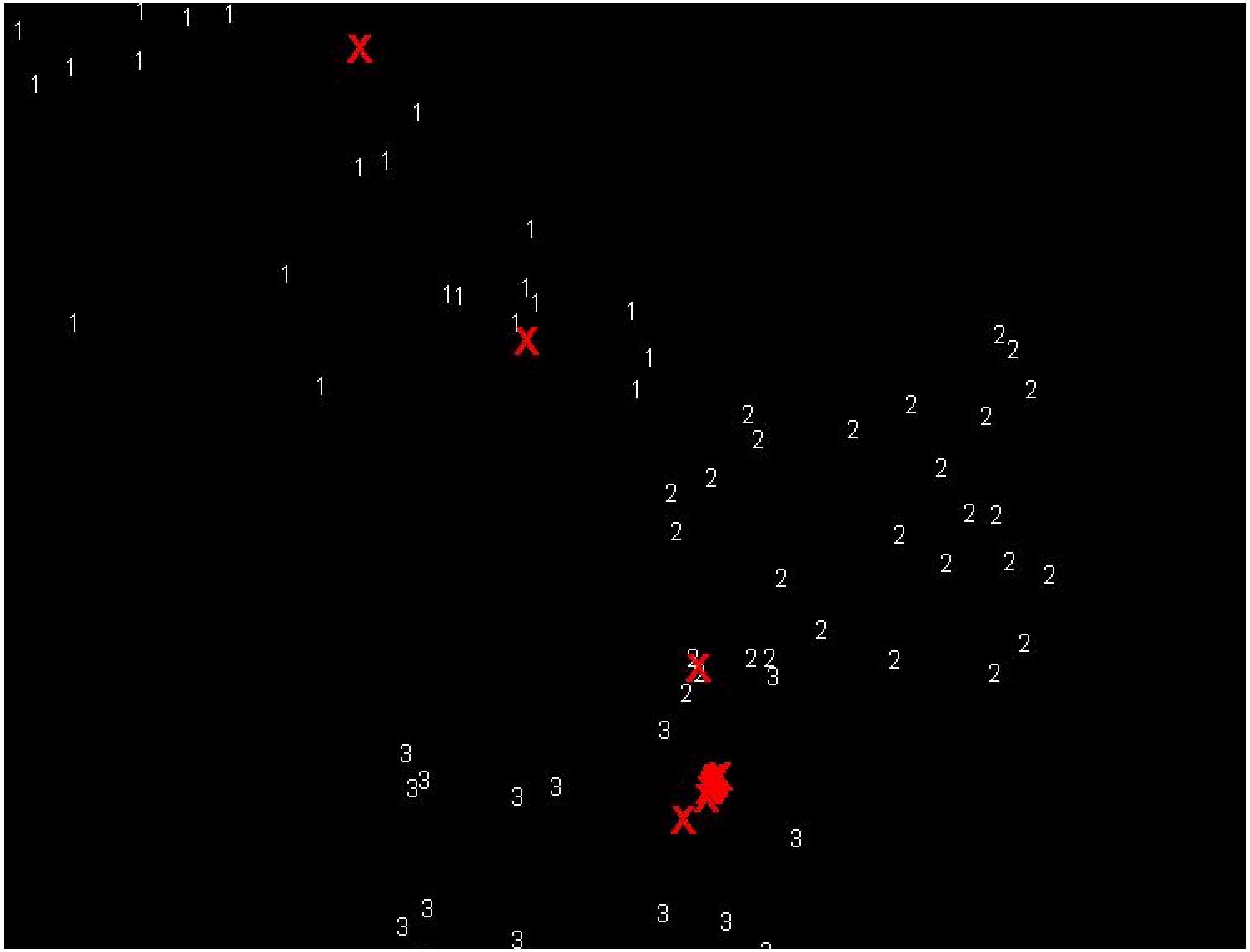
A 2-d location problem





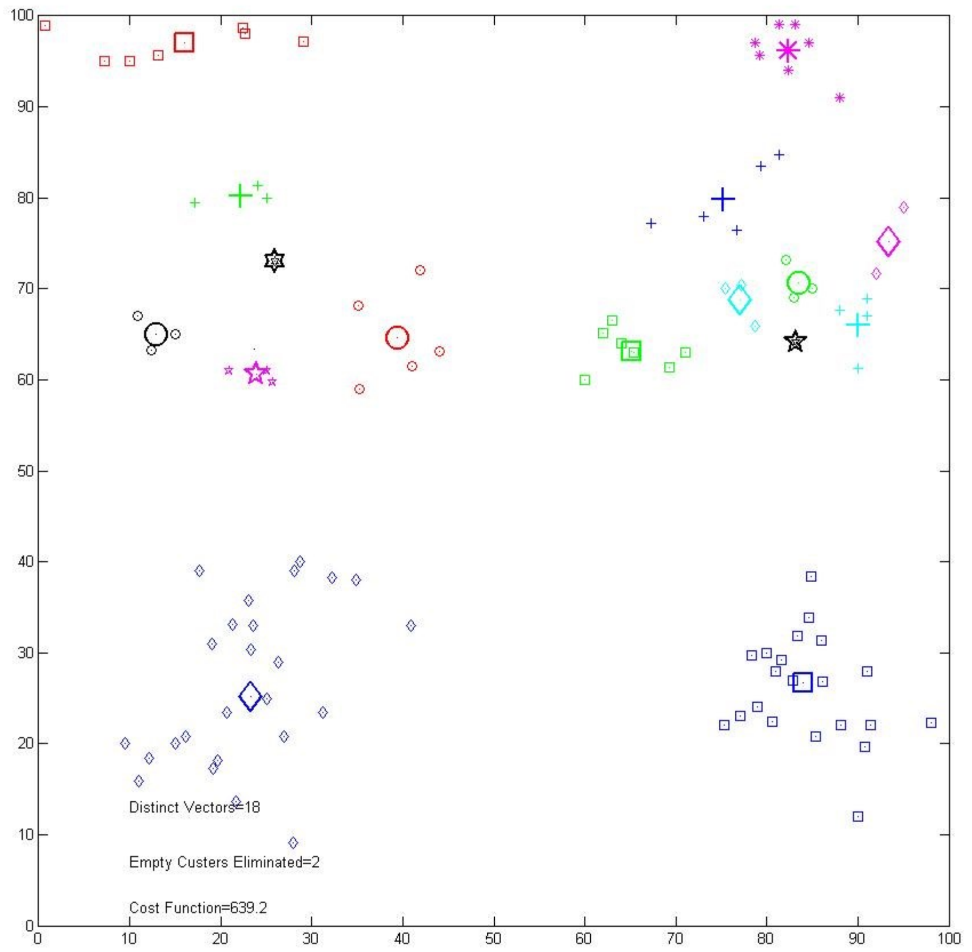

Local intran





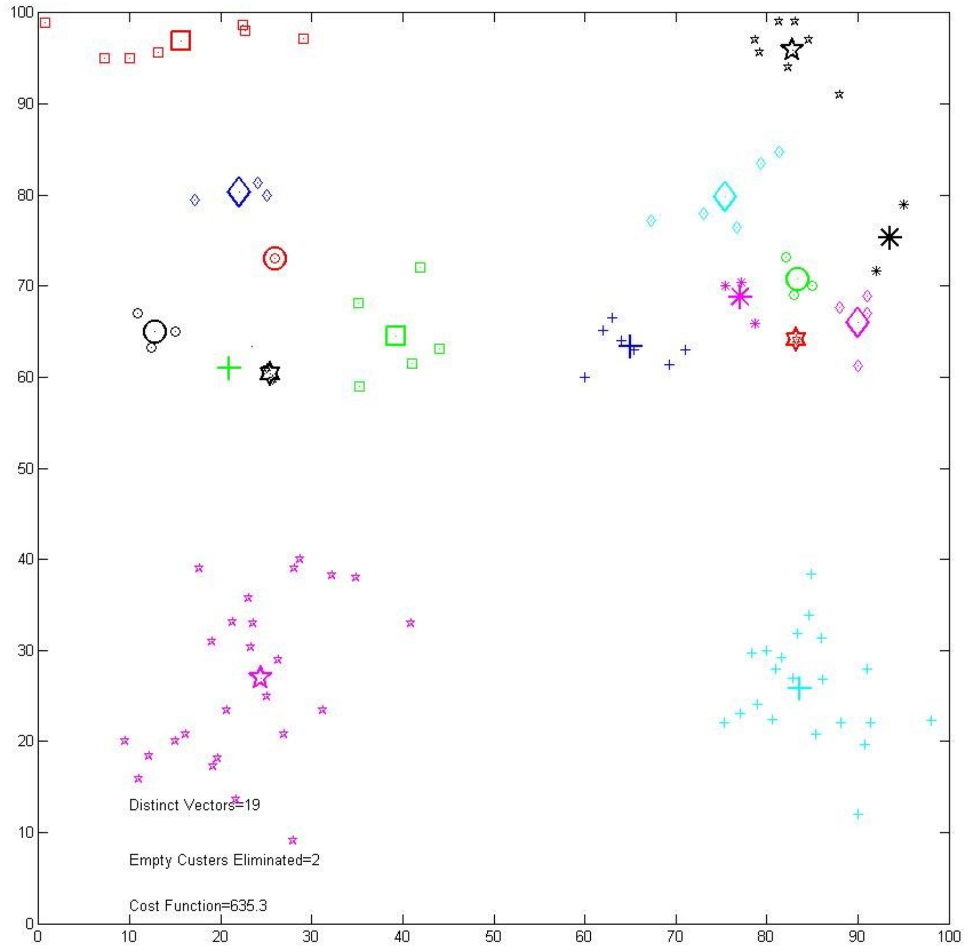
The SOM and data topology

100 training vectors, 18 distinct vectors



The SOM and data topology. Another run.

100 training vectors, 19 distinct vectors



Spectral Clustering

The spectrum of a matrix is a set of invariants, including its eigenvalues, that characterize the transformation

Those invariants are part of a profound and elegant relationship between a graph representation of a dataset and the clustering solution for that dataset

The Concept

- Cluster based on the degree of connectedness among the points
  - Nearby points strongly connected
  - Far away points weakly or not connected

- That principle enables clustering of non-convex sets
  - K-Means will perform well only on convex sets

A great reference!

For the motivation, details, illustrative examples, and related variations of Spectral Clustering, in a very readable review, look in

<u>A Tutorial on Spectral Clustering</u>

Technical Report #TR-149, von Luxburg, Ulrike

Aug 2006, Max-Planck-Institut für biologische Kybernetik

The following borrows heavily on his work.

The Ng-Jordan-Weiss Algorithm in a Nutshell

- Map each data point to a graph vertex , with the edges representing the similarity between the vertices (n data points)
- Represent the graph by a weighted adjacency matrix
- Construct the (normalized) Laplacian matrix for the graph
- Find the eigenvalues and eigenvectors of the Laplacian matrix
- Define k by a steep jump in the values of the eigenvalues
- Take the k columns of eigenvectors corresponding to the first k eigenvalues, and read across rows. The k-element row is the coordinates of a new data point
- Cluster these n new data points

The algorithm, step by step

SIMILARITY OF DATA POINTS

We need a measure of similarity (in contradistinction to distance) so that we can relate data points to one another

The Gaussian similarity $s(a,b) = e^{\frac{-\|a-b\|^2}{2\sigma^2}}$

is popular. σ controls the width of the neighborhood

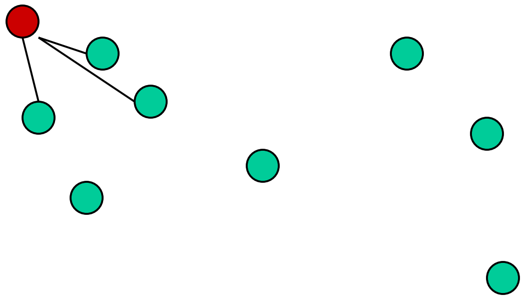
Adjacency

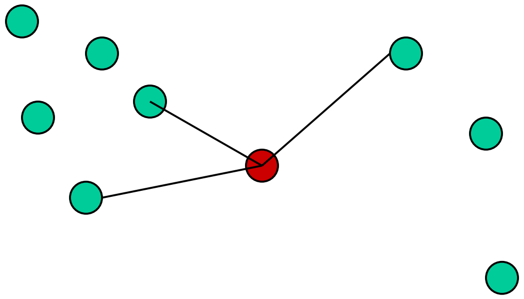
Next, we need to know which data points are adjacent to which

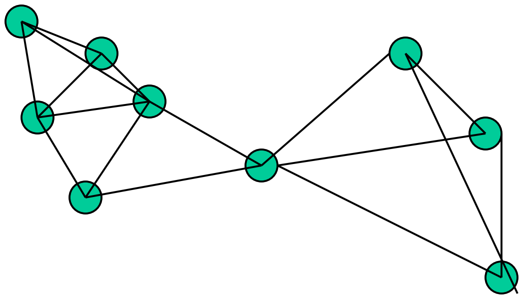
We know how similar they are by our similarity measure. Now we need to know what defines ‘adjacent’, based on the similarity measure.

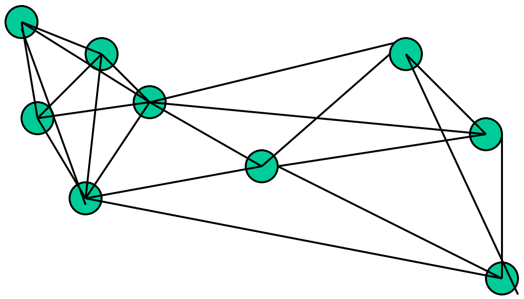
Two popular ways to define adjacency:

- ε- neighborhood: all data points in the ball of radius ε
- k-nearest neighbors: the k most similar data points to the index point


3 NN for a data point

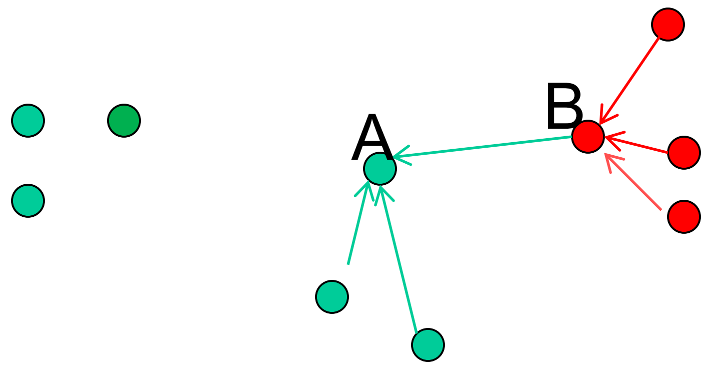

3 NN for a different data point


3 NN graph for all data points


4 NN graph for all data points

There is a gradual shift from local connectivity to global connectivity as the number of nearest neighbors increases

NN *vis à vis* mutual NN



A similarity edge is directed. In the above graph, green arrows indicate the 3 NN to the data point A. Likewise, red arrows show the 3 neighbors nearest to data point B. Note however, that while B is adjacent to A, A is not adjacent to B.

In our discussion, we will consider NN, and not mutual NN, and consider the graph to be undirected. The consequent ‘extra’ edge at vertex B will be accounted for later in the normalization process.

Adjacency Matrix

- Make an adjacency matrix A where the matrix entries are the similarities between n data points. For each point, try the similarity with every other point. If the similarity fails the adjacency test (*i.e.* it is not a k-NN or if it is not within the ε-ball), its matrix value is assigned 0
- A weight matrix differs from an adjacency matrix in that the distances on the edges are not binary, and the graph is fully connected

Choosing how many NN

- Choose k for k-NN such that data points are heavily connected locally, yet have either weak or no connections with remote neighborhoods.

- There is a trade-off between weak connection and no connection. The theoretical advantage of a weak connection (so that the entire graph is connected) is offset by the noise in the solution to the eigen problem in a very large matrix that is not sparse

Laplacian Matrix

- Next, using the values from the adjacency matrix, make a graph Laplacian matrix
  - First find the n x n diagonal degree-matrix D, that is, the sum of the weights of connected edges from each vertex. Put that on the diagonal of D
  - Next find the Laplacian, choosing from either an unnormalized graph Laplacian, a random-walk normalized graph Laplacian, or a symmetric normalized graph Laplacian

Graph Laplacians

- Unnormalized

$$L=D-A$$

- Normalized Random Walk

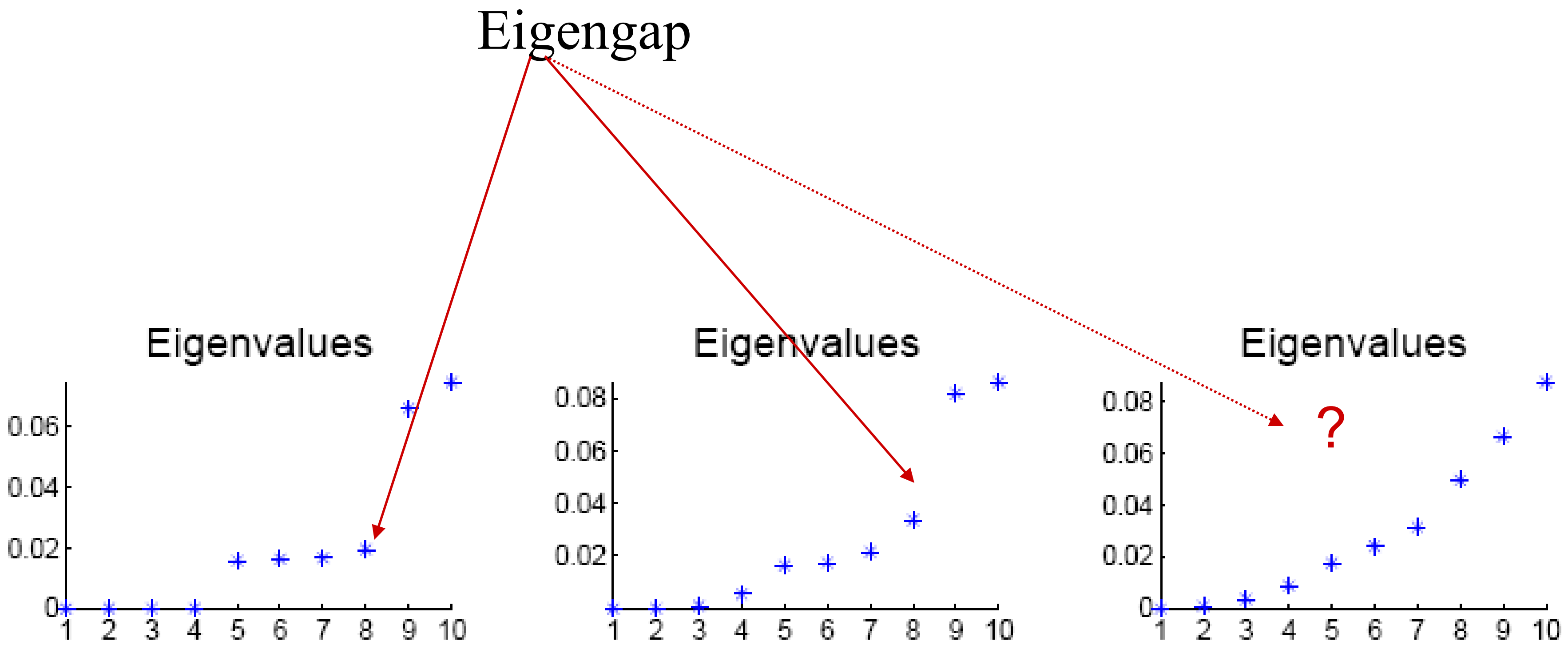
$$L=I - A^{-1} W$$

- Normalized Symmetric

$$L=I - D^{-1/2}A D^{-1/2}$$

Eigenvectors

- Compute the eigenvalues and eigenvectors of L (this is a big job; L is n x n, but is sparse)
- The eigenvalues will ascend in order, usually rising abruptly at some point.
- Choose the cardinality of the eigenvalue just before the abrupt rise (eigengap) to be k



from von Luxburg

Re-interpret the eigenvectors

- The eigenvector elements are in n rows of n columns
- Evaluate only the first k columns up to the cardinality of the eigengap, so there is an n×k sub-matrix
- Recast the n×k sub-matrix data such that the ith data point is given by the coordinates 1 thru k in the ith row, for i=1..n data points (eigenvector elements). Restated, the coordinates of the ith original number are given by reading across the ith row of the eigenvector sub-matrix
- If the graph Laplacian is symmetric-normalized, the rows need to be normalized
- The transformed data are now clustered in an k-dimensional space.

New clusters

- These ‘new’ data points will follow a natural clustering
- Cluster these. This should be trivial- almost visual. If not, most often, k-Means is the algorithm of choice for that.
- The k from the eigengap (*ie* the number of column eigenvectors kept) is the k is for the k-Means clustering .

Notes/Issues

- For a variety of statistical and intuitive reasons, we used the symmetric Laplacian although the random-walk Laplacian gave the same results

- Key properties of the Laplacian
  - 0 is an eigenvalue of L, with eigenvector of 1's
  - All n eigenvalues are positive, semidefinite, real valued with $0=\lambda \leq \lambda_1 \leq \lambda_2 \leq \lambda_3 \ldots \leq \lambda_n$
  - The more the eigenvalues are closer to 0, the more distinct the new clusters will be. Ideally there would be k-eigenvectors, each with an eigenvalue of 0
  - The matrix is sparse so there are efficient eigenvalue algorithms to deal with a huge matrix

Why does this work? Explanation #1: Intuitive

Consider that a demon wanders randomly, for a very long time, from vertex to vertex

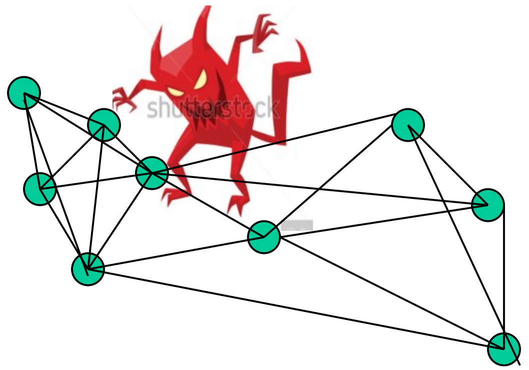
The vertices are originally weighted with transition probabilities.

Assume that the vertices are connected (*ie* the transition matrix is ergodic). After the demon has made a very large number of transitions, the edge weights will become the stationary transition probabilities, with the vertices corresponding to states.

Why does this work? Explanation #1: Intuitive

It makes intuitive sense that the demon will ultimately end up in the neighborhood where she ‘belongs’ in the limit.

This notion of finding her ‘rightful’ (*sci* limiting) place is tantamount to identifying a cluster to which the demon belongs



Why does this work?
Explanation #1: Intuitive

Of course, by now we know that the way to find the stationary transition probabilities is simply to find the eigenvectors of the original transition matrix.

This overview is somewhat complicated by the fact that the original graph might be in m-dimensional space, but the math takes care of that.

Why does this work? Explanation #1: Intuitive

So, how do we get from the graph to a probability transition matrix?

By the graph Laplacian! If, for each node, the probability of leaving the node is the specific edge weight / weight of all edges leaving the node, then, for the entire matrix, that probability is **A** / **D**

Now recall the definition of the Random Walk Laplacian:

$$L_{rw} = I - \frac{A}{D} \; or \; I - D^{-1}A$$

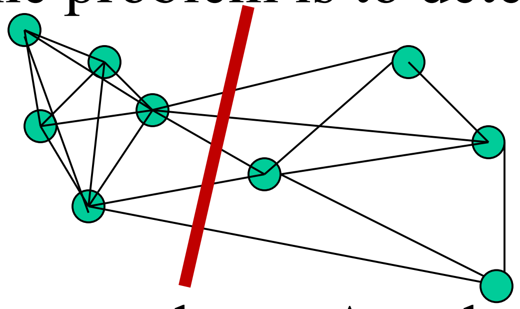
So, the graph Laplacian is simply the complement of the probability matrix

Intuitive Explanation #2

Points close to each other will aggregate around the diagonal of the graph Laplacian. Clusters will tend to aggregate separately in this fashion. These ‘subclusters’ along the diagonal will each have an eigenvalue that is zero if the clusters are disconnected, and some small number if the clusters are connected. The number of these small eigenvalues determines the k-arity of the clustering.

Graph Cuts in Brief

From graph theory (and heavily implemented in VLSI design) is the graph-cut problem The problem is to determine a cut that will partition a graph.



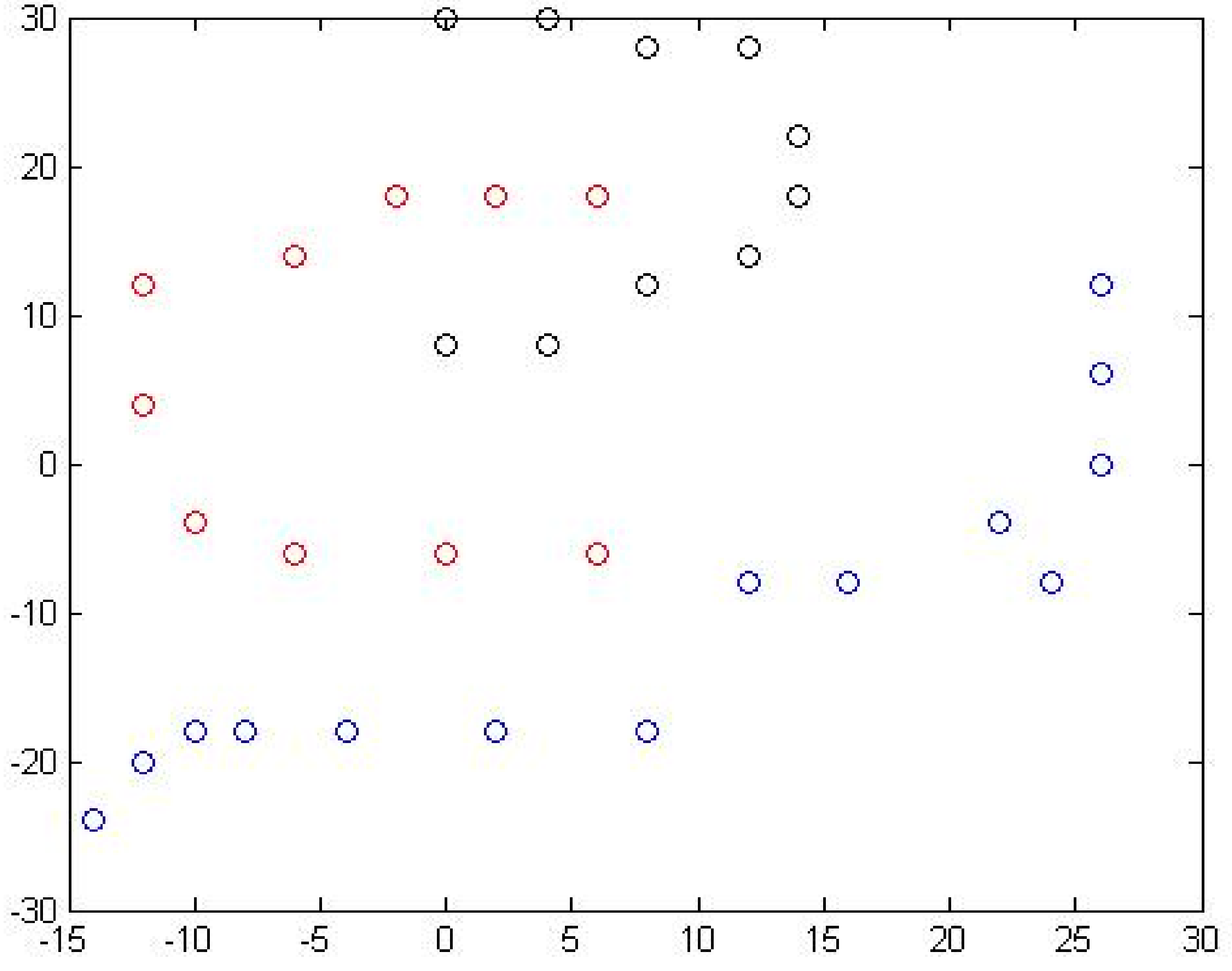
Normalized cut(A,B) between cluster A and cluster B:

$$cut(A,B)\left(\frac{1}{vol(A)}+\frac{1}{vol(B)}\right)$$

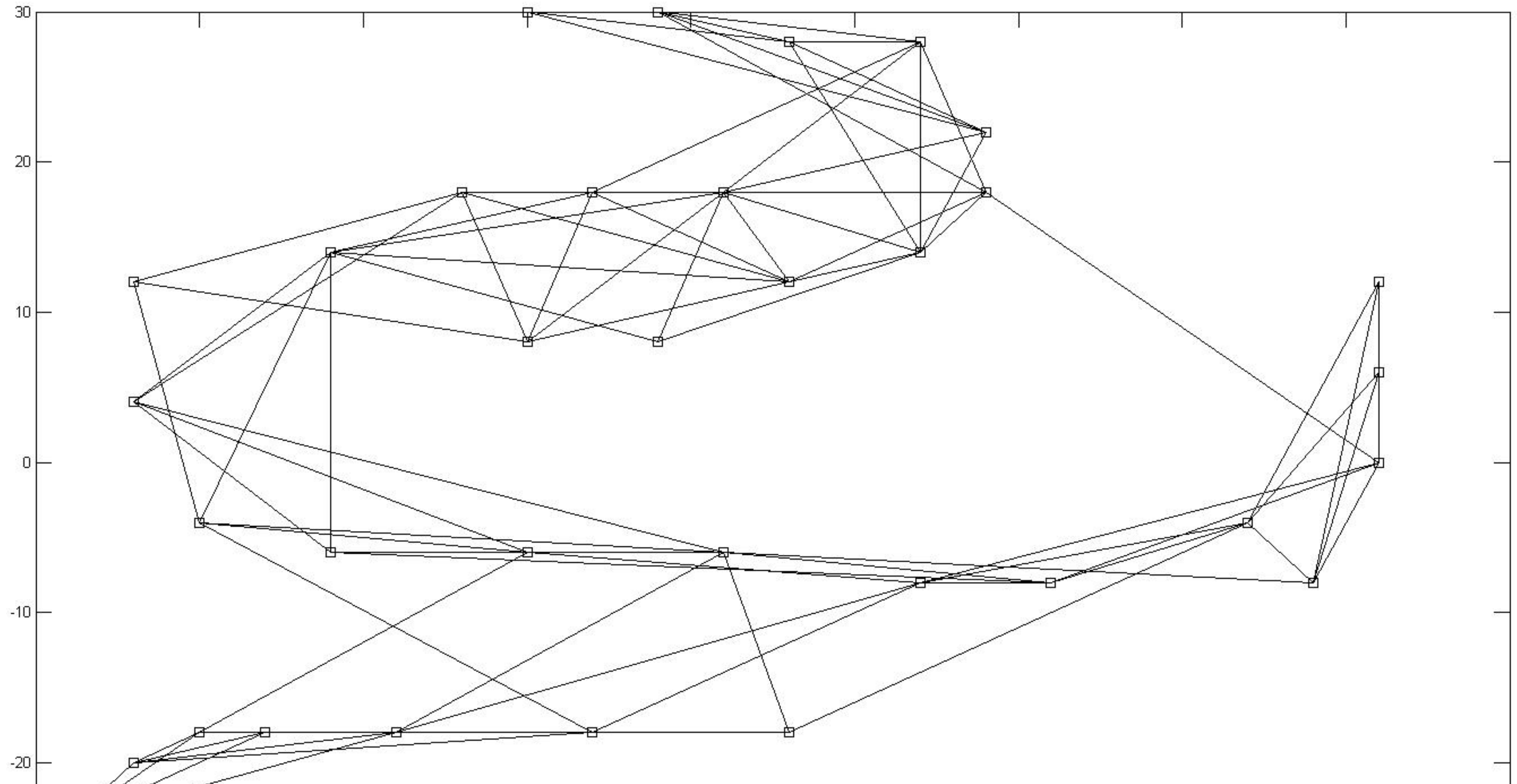
where cut(A,B) is the sum of the weights of all the connections between A and B and vol(A) is the sum of the weights of all edges that come from all data points in A

For these 2 clusters, the solution is the first two eigenvectors. The notion is generalizable to higher k

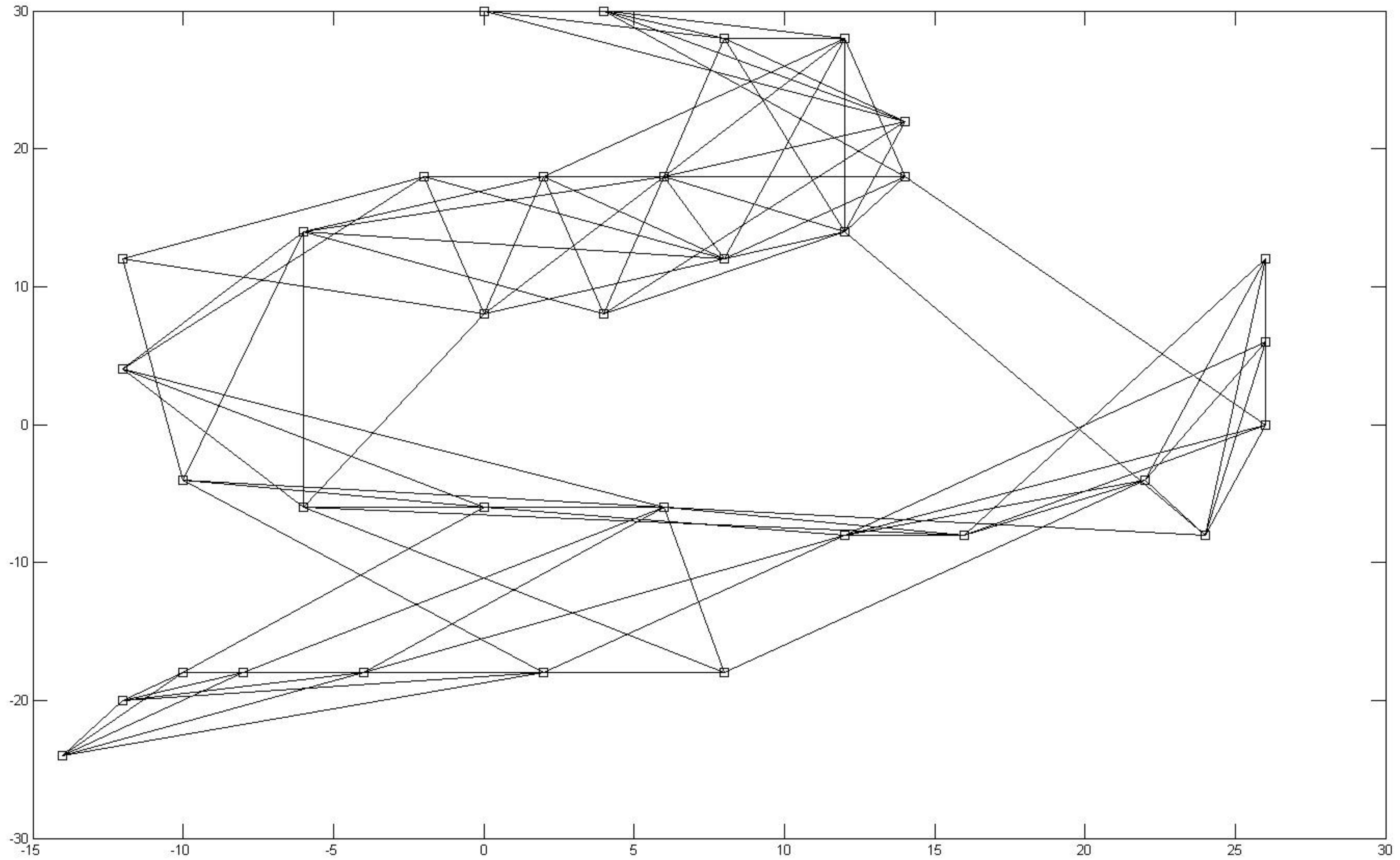
Example of NJW Algorithm results
Raw data colored to reflect intuitive clusters



3-Nearest Neighbors

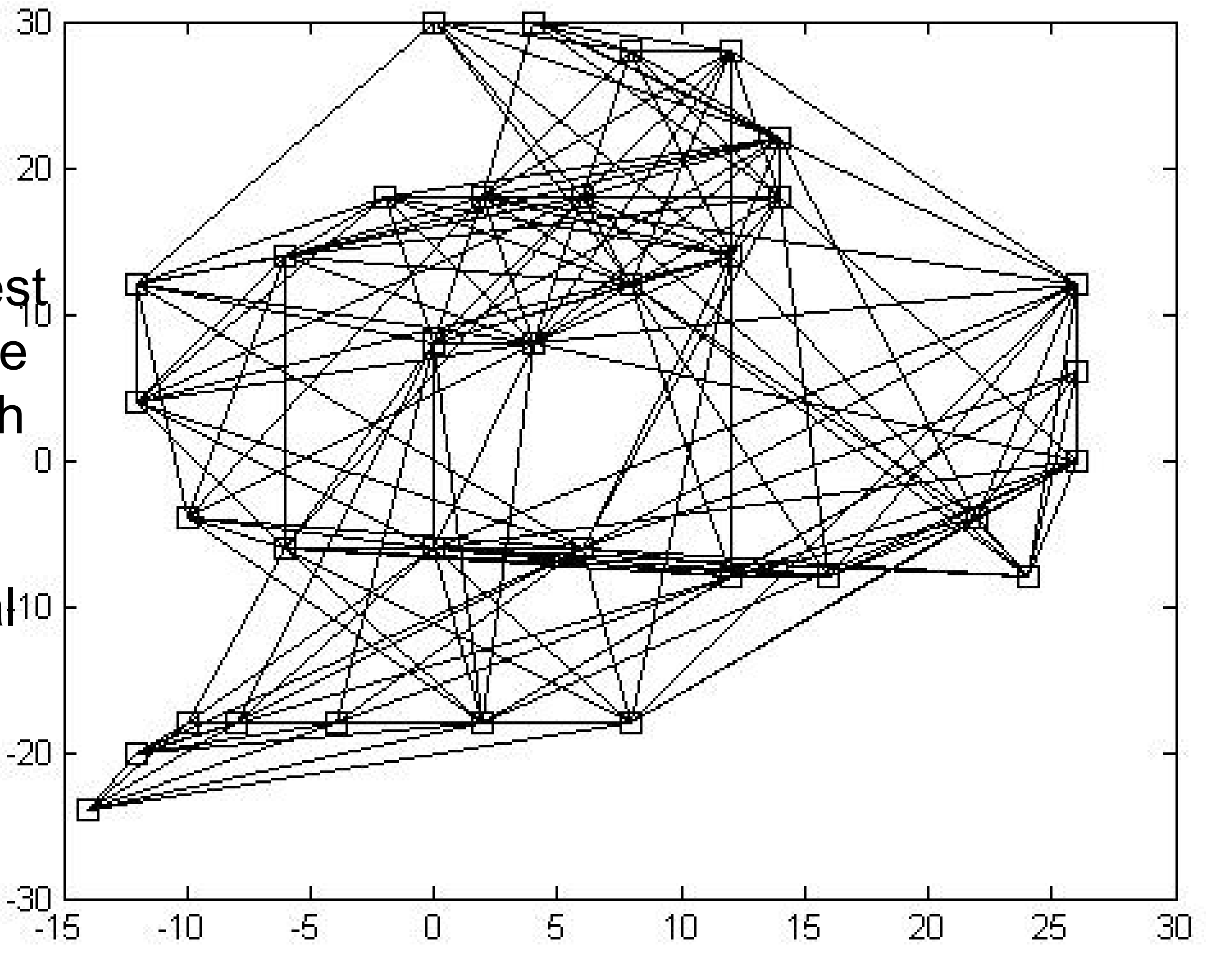


4-Nearest Neighbors

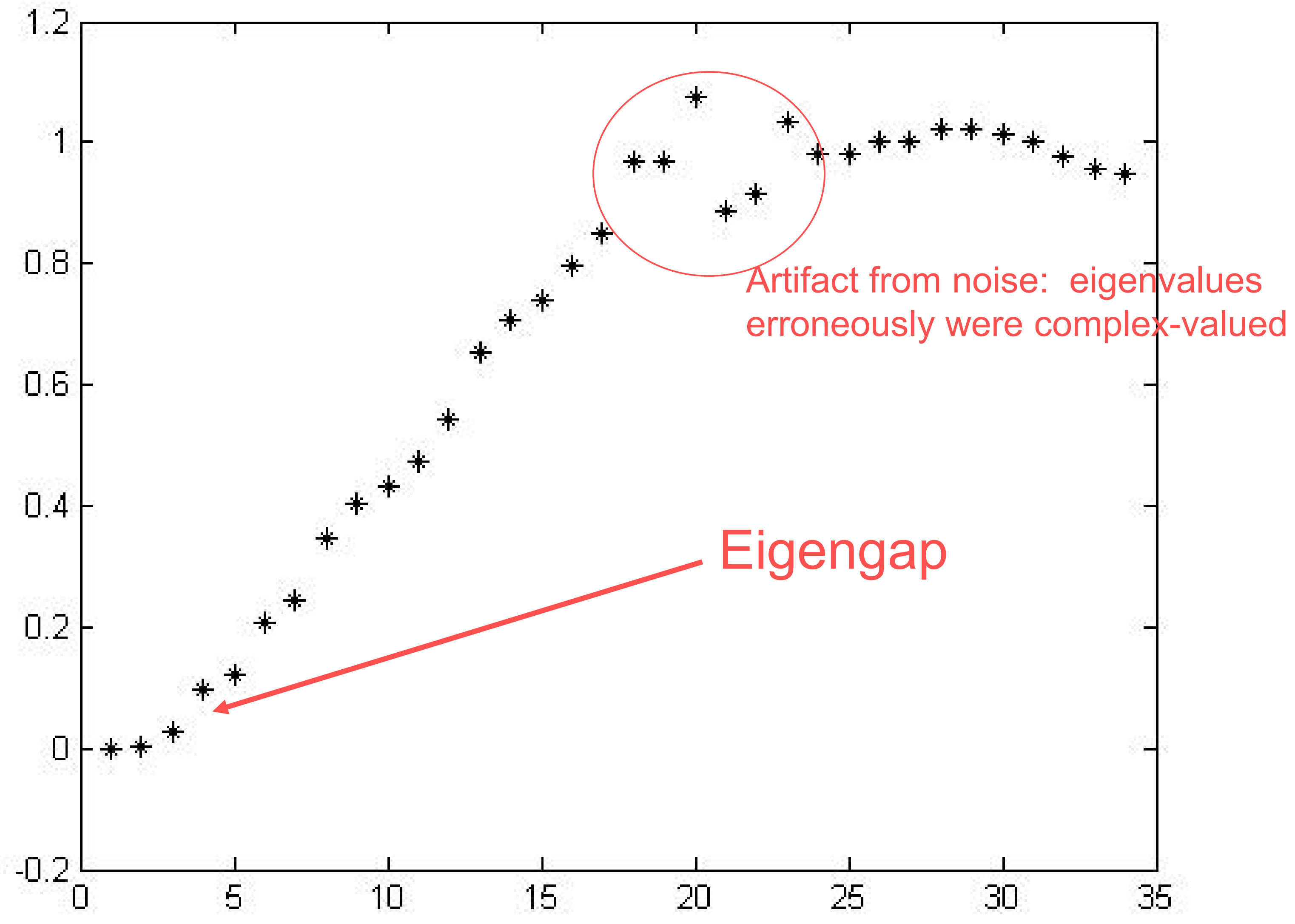


KNN-8

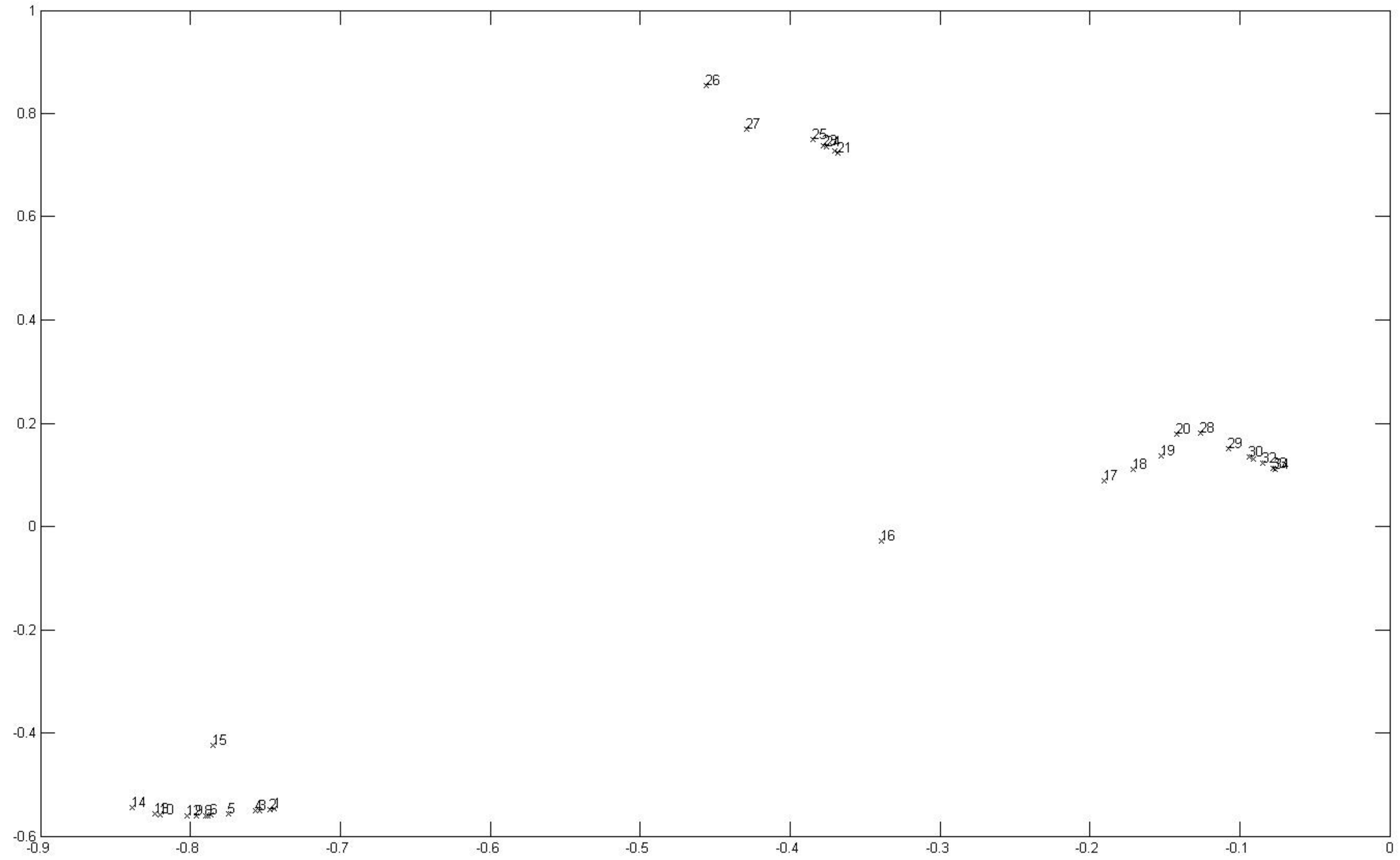
As the number of nearest neighbors increases, the connectivity of the graph is changing from an emphasis on local neighborhoods to global influence.



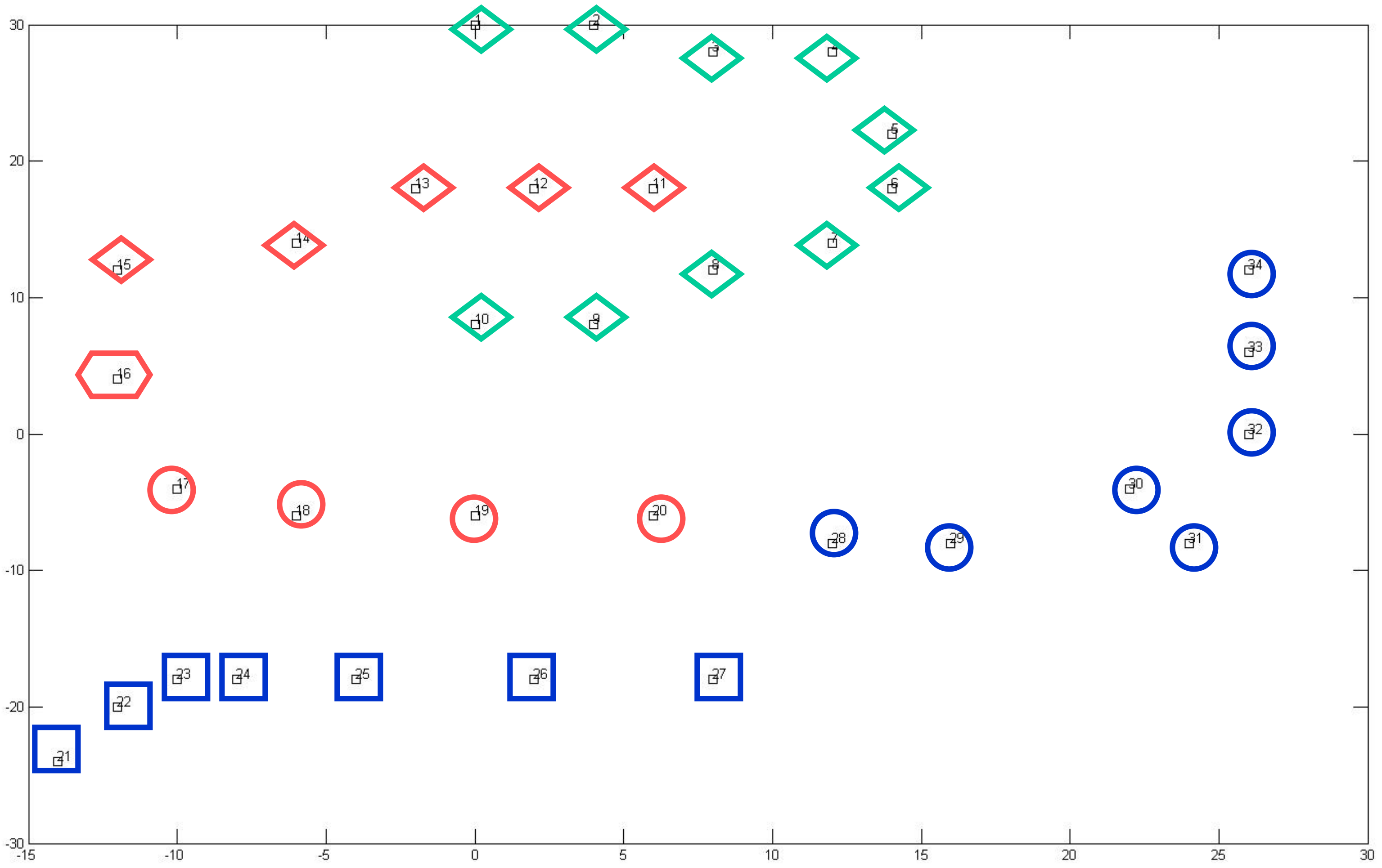
Eigenvalues of the Laplacian for 4-NN



First 2 eigenvectors
3 obvious clusters
Labels are revealed

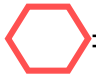


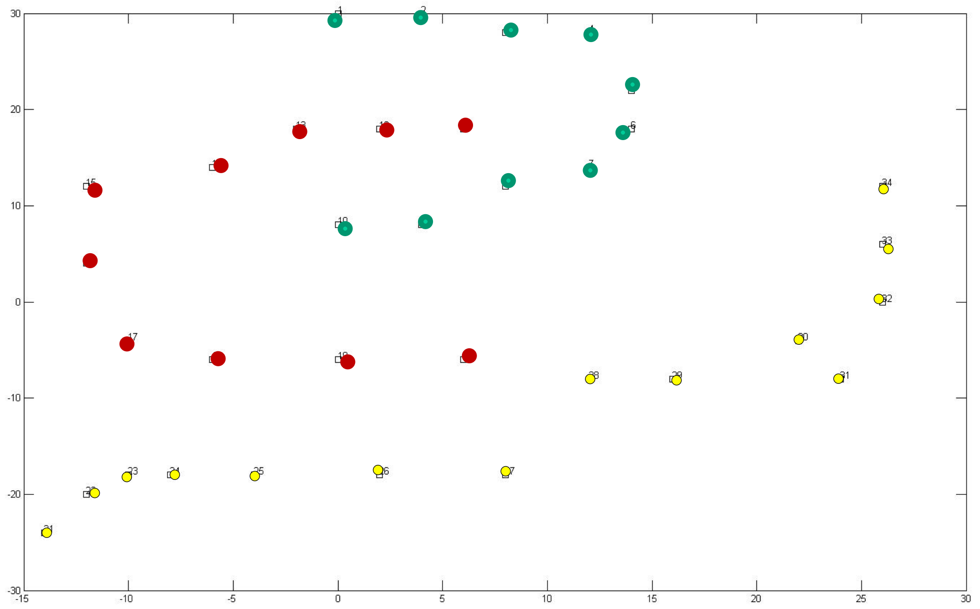
Labels Revealed



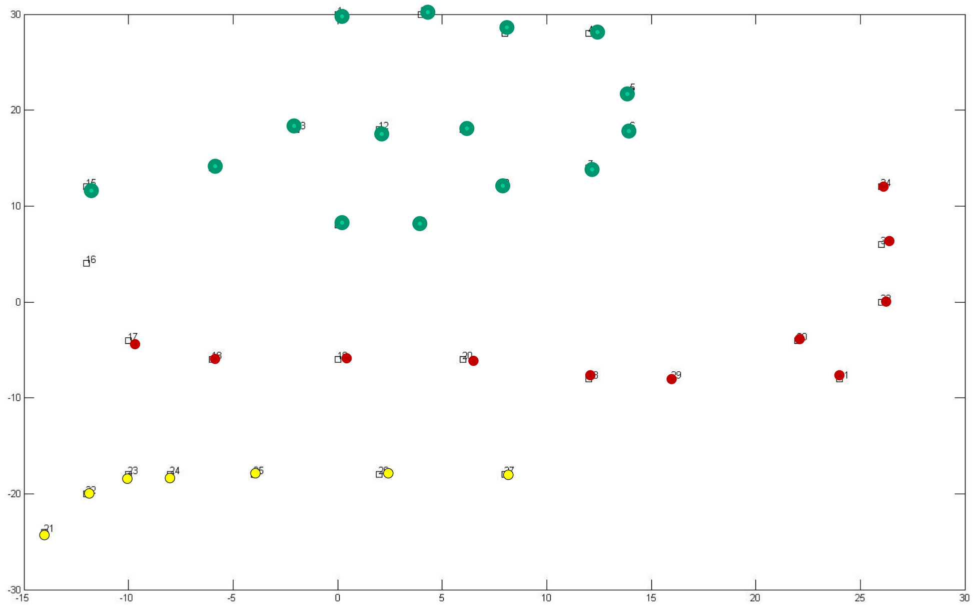
COLOR=ORIGINAL INTENT

SHAPE=CLUSTERING REULT


=POSSIBLE SEPARATE CLUSTER FOUND



Intuitive clusters (Experimental)



Spectral clusters

More Sophisticated Steps

Algorithms can build on the idea that data to be clustered have attributes of both density and connectedness.

Algorithms that exploit these attributes simultaneously are called multi-parameter clustering algorithms. Typically they are agglomerative strategies but not always.