METU Informatics Institute
Min720

Pattern Classification with Bio-Medical Applications

Part 10: Neural Networks-2
Clustering

Back Propagation as Feature Mapping

- Consider hidden-to-output layer
- Has capability of separation for a linearly separable problem.
- That means, the hidden node outputs should lie in a linearly separable space.
- Consider the x-or problem again. See how the y_1, y_2 outputs change while learning is taking place.
- We can see from figure that the iterations move y's towards a linearly separable position (next page).
- Another example: (next next page) shows the importance of the number of hidden nodes.

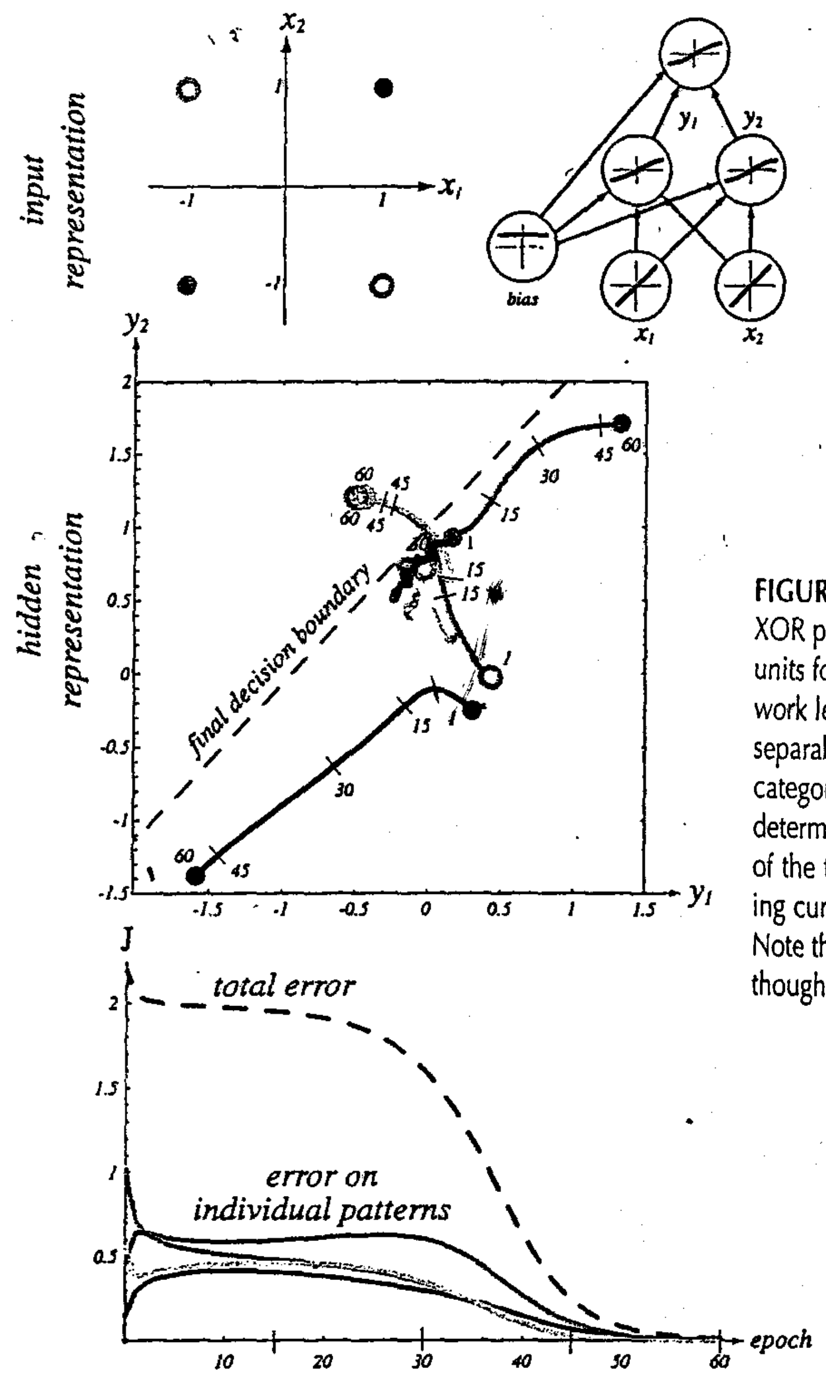


FIGURE 6.10. A 2-2-1 backpropagation network with bias and the four patterns of the XOR problem are shown at the top. The middle figure shows the outputs of the hidden units for each of the four patterns; these outputs move across the $y_1 y_2$-space as the network learns. In this space, early in training (epoch 1) the two categories are not linearly separable. As the input-to-hidden weights learn, as marked by the number of epochs, the categories become linearly separable. The dashed line is the linear decision boundary determined by the hidden-to-output weights at the end of learning; indeed the patterns of the two classes are separated by this boundary. The bottom graph shows the learning curves—the error on individual patterns and the total error as a function of epoch. Note that, as frequently happens, the total training error decreases monotonically, even though this is not the case for the error on each individual pattern.

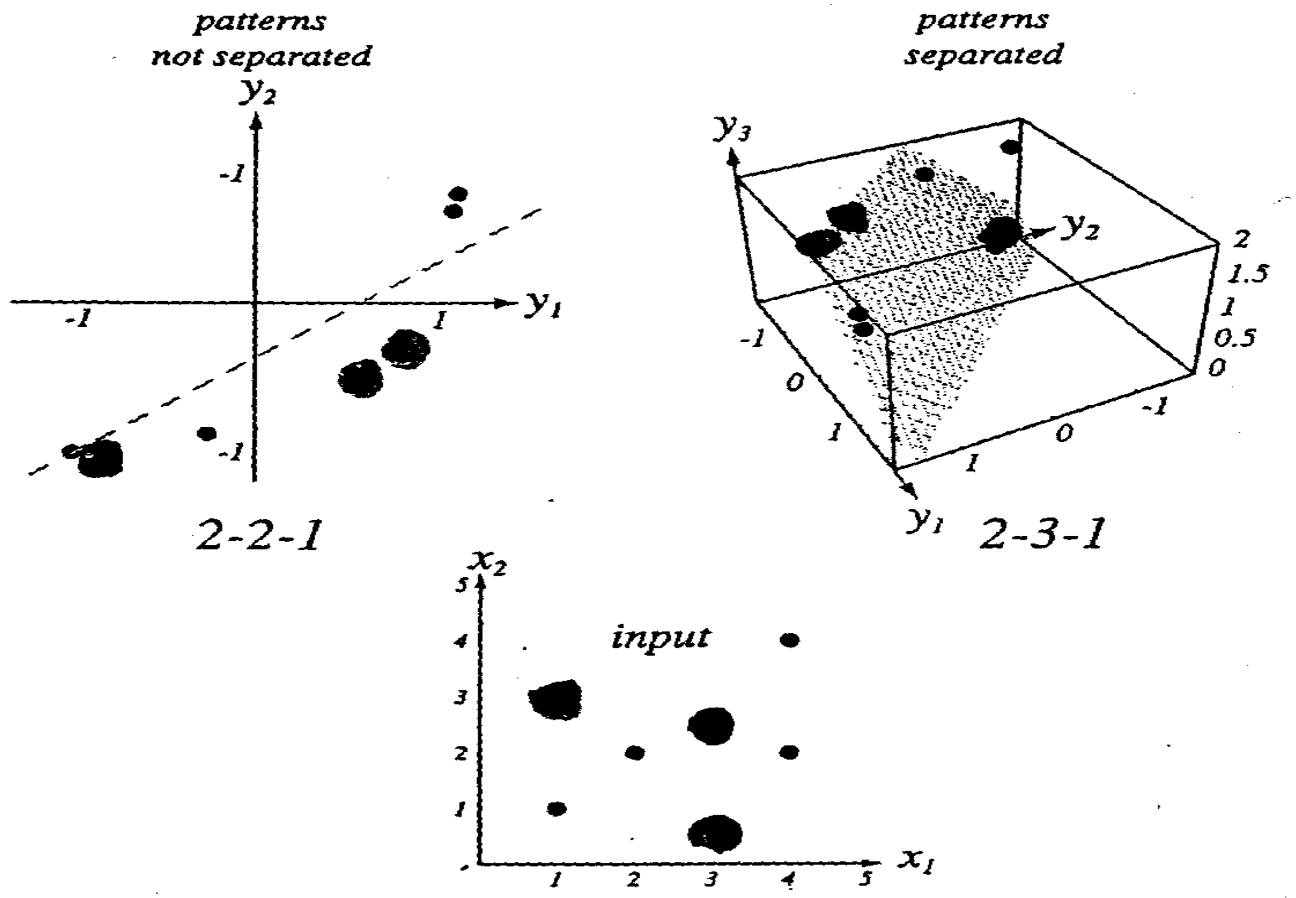


FIGURE 6.12. Seven patterns from a two-dimensional two-category nonlinearly separable classification problem are shown at the bottom. The figure at the top left shows the hidden unit representations of the patterns in a 2-2-1 sigmoidal network with bias fully trained to the global error minimum; the linear boundary implemented by the hidden-to-output weights is marked as a gray dashed line. Note that the categories are almost linearly separable in this $y_1 y_2$-space, but one training point is misclassified. At the top right is the analogous hidden unit representation for a fully trained 2-3-1 network with bias. Because of the higher dimension of the hidden layer representation, the categories are now linearly separable; indeed the learned hidden-to-output weights implement a plane that separates the categories.

Learned weights – what do they mean?

Input to hidden weights

- Such weights at a single unit describe the input pattern that leads to a maximum activation.
- Finds feature groupings
- Example : next page(Fig. 6.13 in DHS)
- Input to hidden weights for a 2-hidden nodes topology shows the features emphasized for each node.

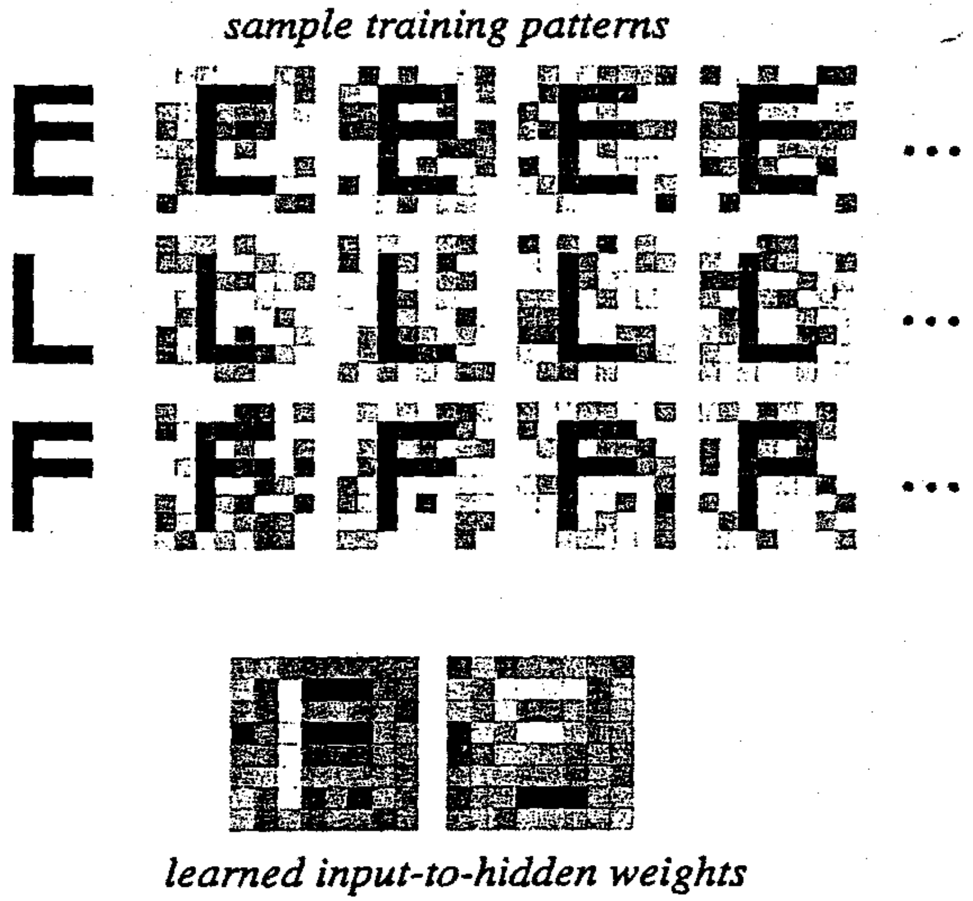
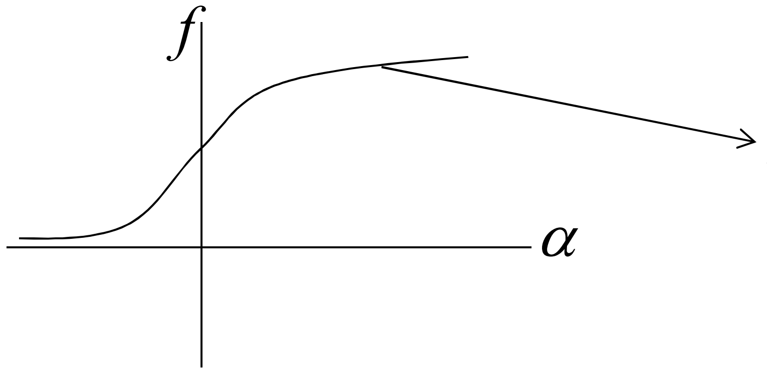


FIGURE 6.13. The top images represent patterns from a large training set used to train a 64-2-3 sigmoidal network for classifying three characters. The bottom figures show the input-to-hidden weights, represented as patterns, at the two hidden units after training. Note that these learned weights indeed describe feature groupings useful for the classification task. In large networks, such patterns of learned weights may be difficult to interpret in this way.

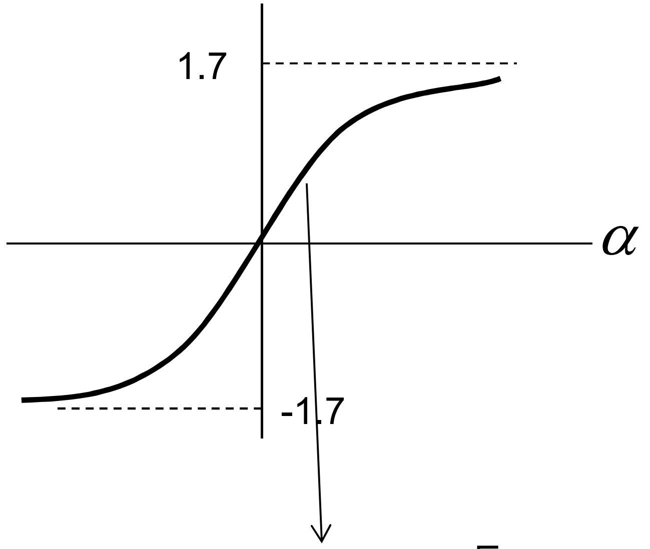
General rules to follow when implementing MLP and Back Propagation Algorithm (Practical Implications)

- *Activation Function* – works with any continuous function with derivative f' = f(1-f)
- Gaussian activation functions are also used, especially if it is guessed that the data is generated by gaussian distributions.

Proper Activation Functions:



$$f = \frac{1}{1+e^{-\alpha}}$$



-Should saturate
-Should be nonlinear
-Should be smooth
-<u>"Odd" function</u> preferred because they lead to faster convergence(f-a)=-f(a).
-<u>Monotonicity</u>

$$f(\alpha) = a\tanh(b\alpha) = a\left[\frac{e^{+b\alpha} - e^{-b\alpha}}{e^{b\alpha} + e^{-b\alpha}}\right]$$

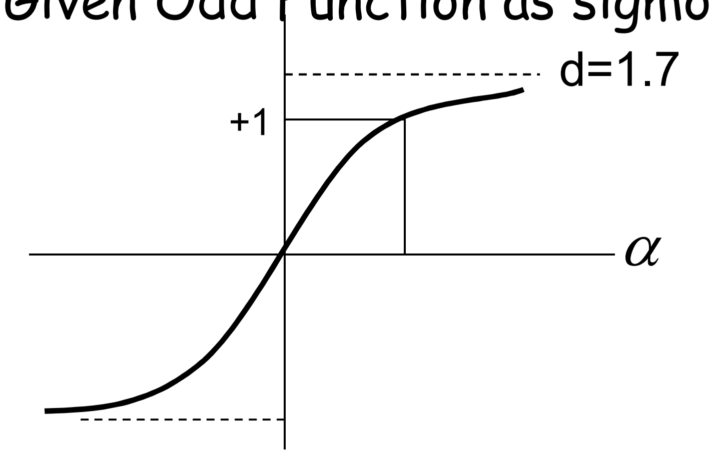
-<u>is a good sigmoid function</u>

Scaling inputs

- If the input features have different units, then they should be normalized to have zero mean and standard deviation=1. Otherwise, the network will train with higher values. Same is true even with same unit but different sizes.

Target Values

Given Odd Function as sigmoid



- Should the target values be selected as d?
- No, since it's never reached, the algorithm will not terminate.
- Use +1, -1 instead.

Number of Hidden Units(n_H)

- Complicated boundaries require higher n_H
- Nearly linear boundaries require small no. of n_H
- But if you don't have enough samples, no. use of using high n_H
- Rule of thumb: $n_H = n/10$ where n- total no. of training samples

Initializing Weights

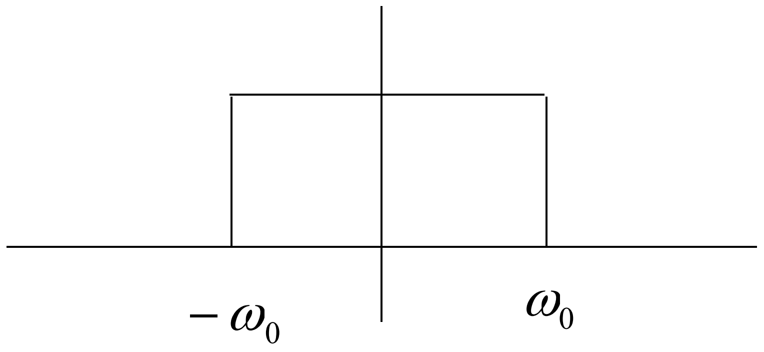
- Starting with zero weights cause "no learning" since

- All zero weights lead to all-zero outputs
- Updating weights at hidden nodes cause no change; remember the formula

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j Xi$$

- Uniform learning: all weights reach their fixed values at the same time. We want fast and uniform learning.
- Rule of thumb: Choose weights randomly from a uniform distribution

- Rule of thumb for initializing hidden to output weights:



$$-\frac{1}{\sqrt{n_H}} < \omega_0 < -\frac{1}{\sqrt{n_H}}$$

- Rule of thumb for initializing input to hidden weights:

$$-\frac{1}{\sqrt{d}} < \omega_0 < -\frac{1}{\sqrt{d}}$$

- Calculated output to be in correct range in sigmoid.

Learning Curves

J/n= average error per pattern plotted against number of epochs (iterations) in the backprop algorithm

Learning set: samples used in training
Test set: measures the performance
Validation set: used to decide when to stop

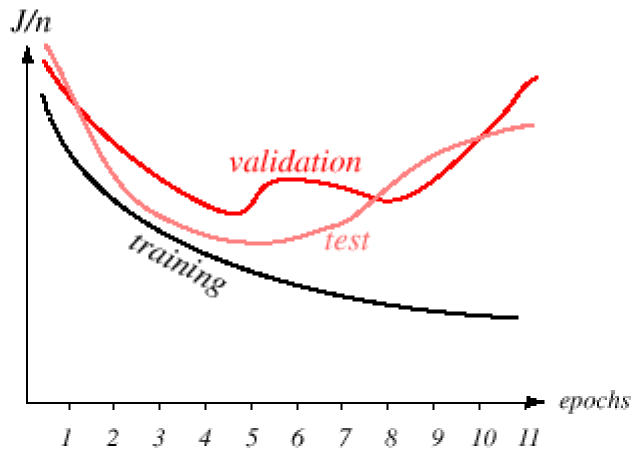


FIGURE 6.6. A learning curve shows the criterion function as a function of the amount of training, typically indicated by the number of epochs or presentations of the full training set. We plot the average error per pattern, that is, $1/n \sum_{p=1}^{n} J_p$. The validation error and the test or generalization error per pattern are virtually always higher than the training error. In some protocols, training is stopped at the first minimum of the validation set. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

Validation and Cross-validation in training

- In general, the labeled data is randomly divided into two : learning and validation sets (they should not overlap)
- The training should stop when validation set passes a minimum
- *M-fold cross-validation:* The training set is divided into m randomly selected disjoint sets of almost equal size
- System is trained m times, each time one of these being used as validation set
- Overall performance is measured as the average of m errors found.
- Approach can be used at different places: for example, optimal k in k-nn can be found using cross validation.

METU Informatics Institute

Min720
Pattern Classification with Bio-Medical Applications

Part 11:
UNSUPERVISED LEARNING AND CLUSTERING

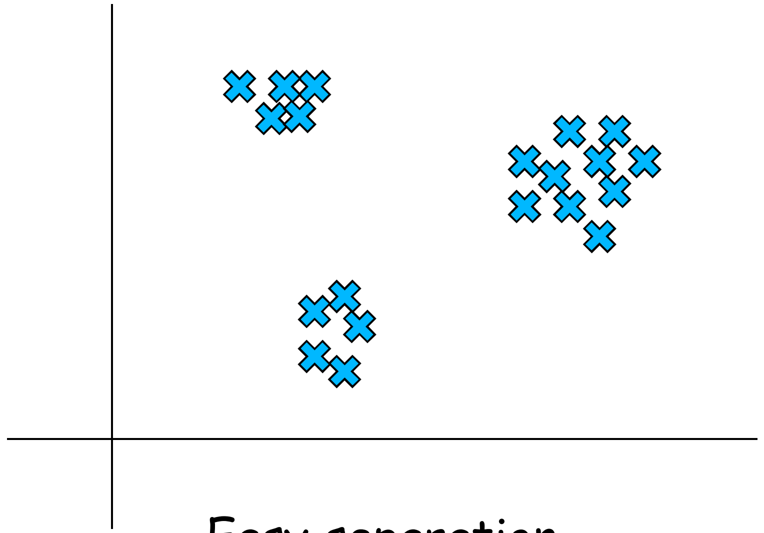
UNSUPERVISED LEARNING AND CLUSTERING

No class labels for learning samples.

We need additional means to label and classify – can be done separately (first label then classify) or together.

PARAMETRIC APPROACH- Estimation of class conditional densities

NONPARAMETRIC – CLUSTERING

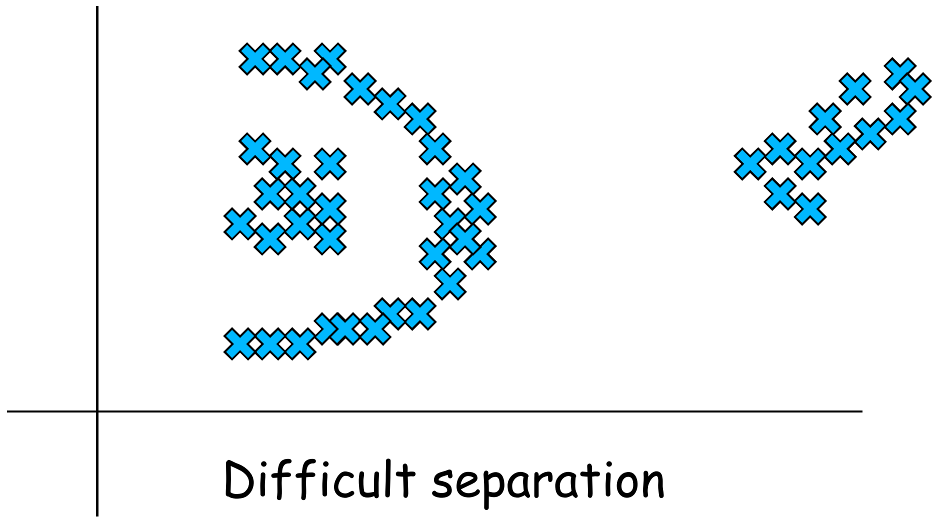


Easy separation example

Problem:

Given samples $X_1, X_2, \ldots\ldots\ldots\ldots, X_n$

Group them into clusters so that samples in same cluster are "similar"



Difficult separation

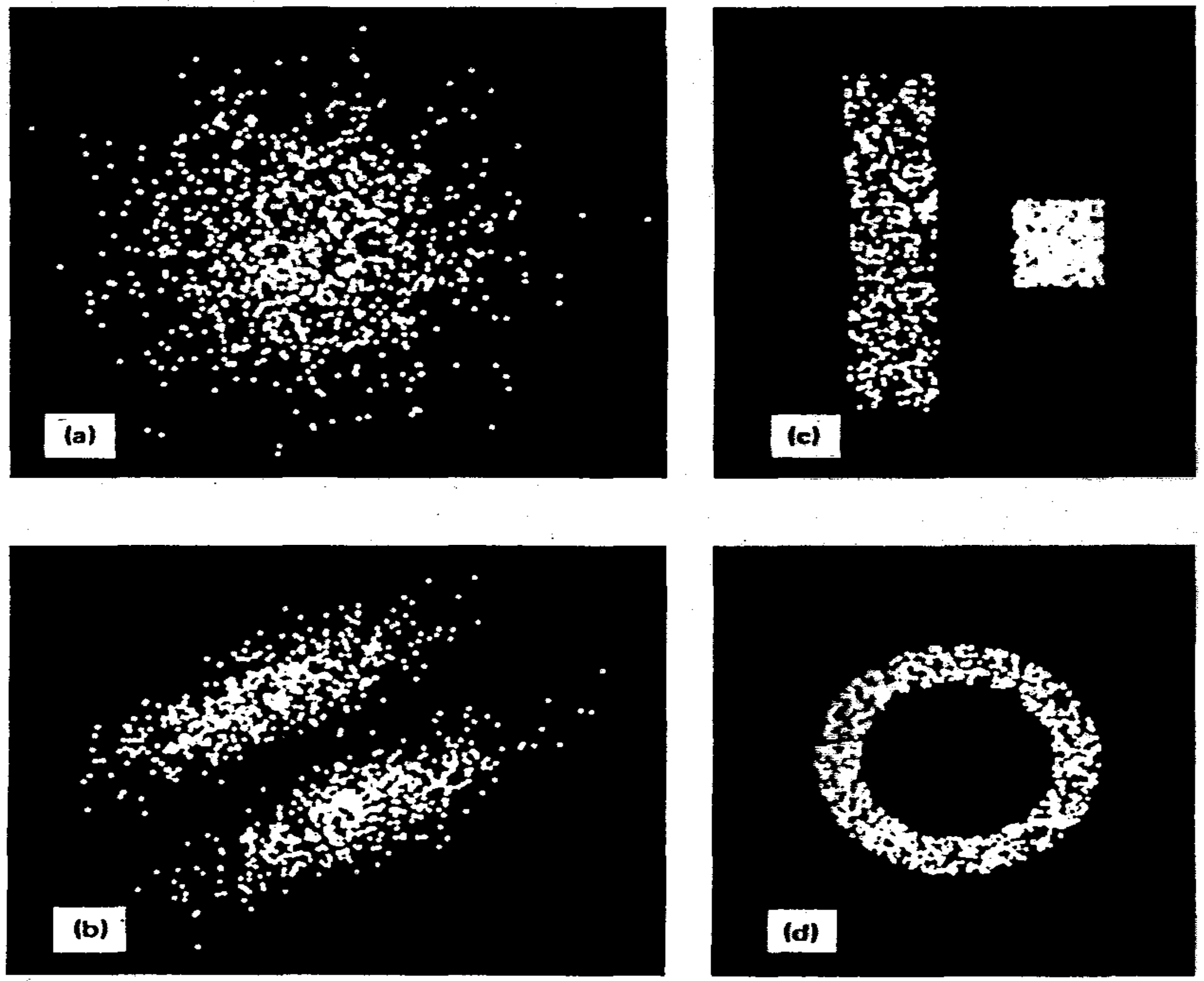


FIGURE 6.7. Data sets having identical second-order statistics.

Similarity Measures

We would like to group the learning samples so that the samples that fall in the same cluster are "similar" and others are "non-similar".

$$S(x_i, x_i) = 0$$

$$S(x_i, x_j) > 0 \quad if \quad i \neq j$$

$S(x_i, x_j)$ - small if $x_i \,\&\, x_j$ belong to same cluster.

$S(x_i, x_j)$ - large if $x_i \,\&\, x_j$ are at different clusters.

$$S(x_i, x_j) = S(x_j, x_i)$$

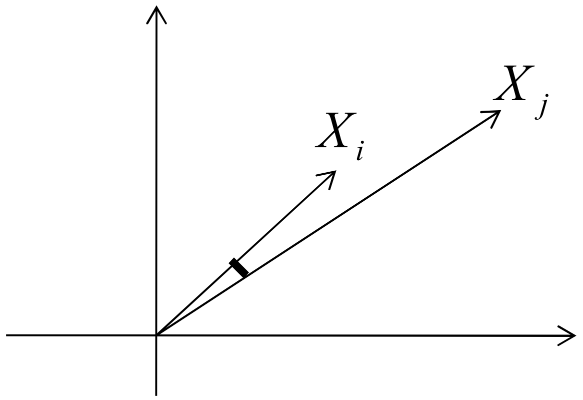
Euclidian Distance is such a measure.

Mahalanobis, cosine distances

Cosine distance: angle wrt origin

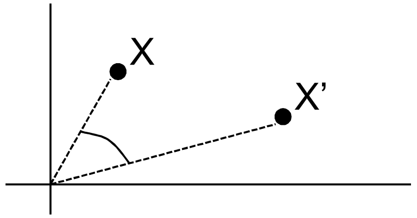
Non-metric :

$$S(X_i, X_j) = \frac{X_i^T X_j}{\|X_i\| \|X_j\|}$$



cosine of the angle between X_i and X_j, so larger if they lie on the same line.

Cosine distance – measures the angle between two vectors X and X’



When the features are binary, assume xi, xj binary vectors

$x_i^t x_j$ - number of ‘1’ ‘s shared

$x_i^t x_j$ / $x_i^t x_i + x_j^t x_j - x_i^t x_j$ <u>tanimoto distance</u>

Number of shared features (being 1),
normalized with number of features that are 1 in x_i and x_j.
Used in information retrival.

Scaling Effects

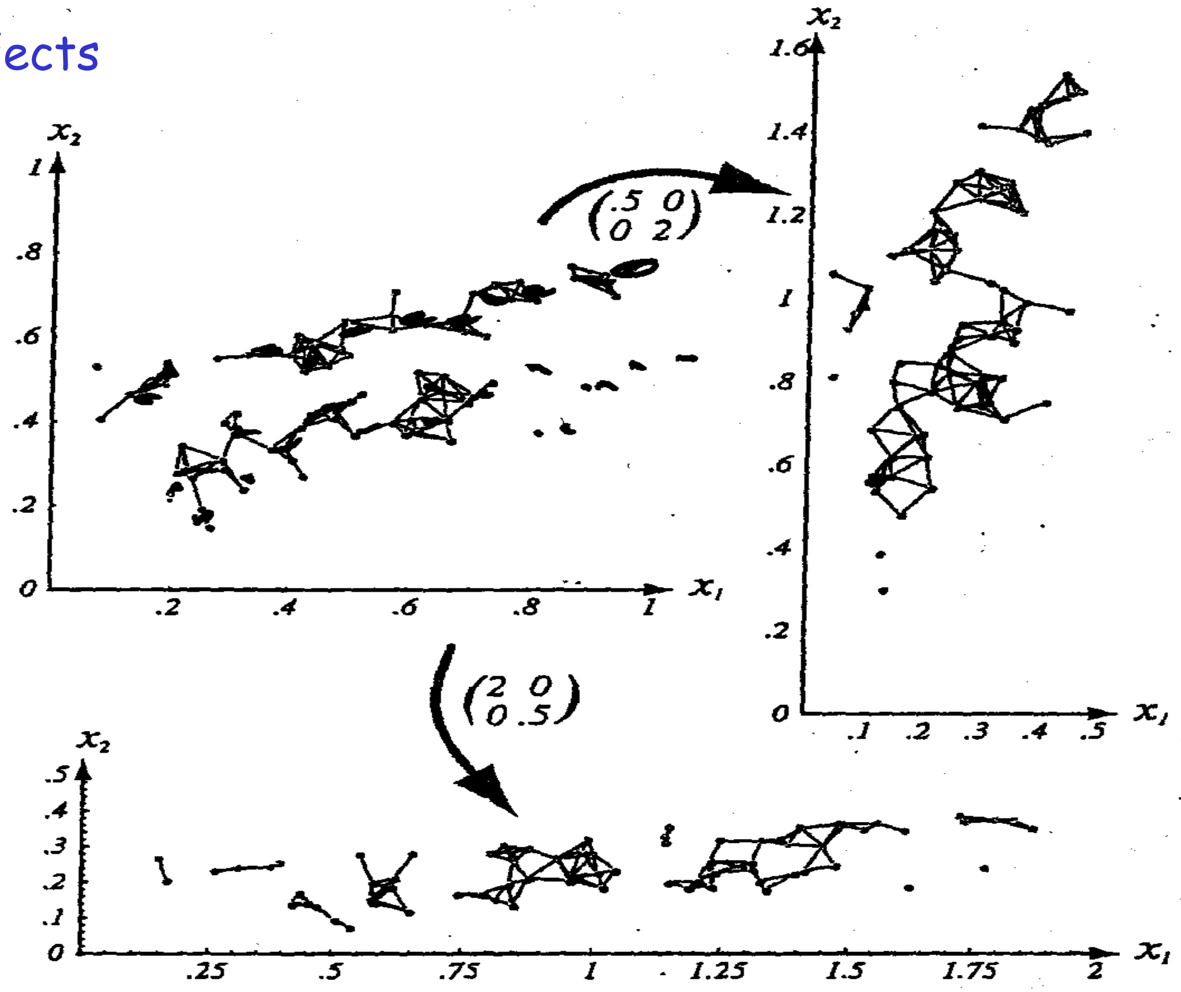


FIGURE 10.8. Scaling axes affects the clusters in a minimum distance cluster method. The original data and minimum-distance clusters are shown in the upper left; points in one cluster are shown in red, while the others are shown in gray. When the vertical axis is expanded by a factor of 2.0 and the horizontal axis shrunk by a factor of 0.5, the clustering is altered (as shown at the right). Alternatively, if the vertical axis is shrunk by a factor of 0.5 and the horizontal axis is expanded by a factor of 2.0, smaller more numerous clusters result (shown at the bottom). In both these scaled cases, the assignment of points to clusters differ from that in the original space.

Scaling Effects

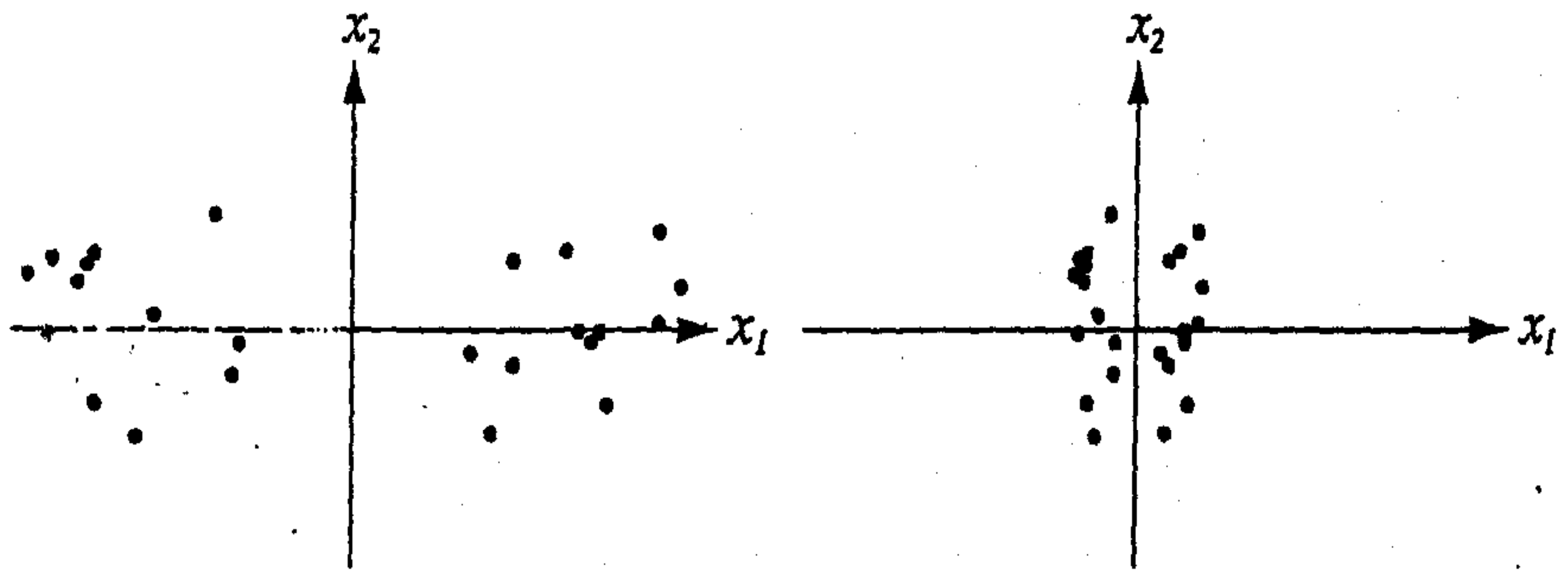


FIGURE 10.9. If the data fall into well-separated clusters (left), normalization by scaling for unit variance for the full data may reduce the separation, and hence be undesirable (right). Such a normalization may in fact be appropriate if the full data set arises from a single fundamental process (with noise), but inappropriate if there are several different processes, as shown here.

Criterion Functions for Clustering

$$J_e = \sum_{i=1}^{c} \left(\sum_{x \in C_i} \| x - M_i \|^2 \right) = \frac{1}{2} \sum_{i=1}^{c} n_i \overline{S}_i$$

$$\overline{s}_i = \frac{1}{n_i^2} \sum_{x \in C_i} \sum_{x' \in C_i} \| x - x' \|^2$$

c- number of clusters

M_i – mean of the samples in the same cluster

<u>Aim: </u>determine the partition that will minimize J.

❖ Minimum variance partition (sum of squared error criterion)

Sum of squared error criterion

$$M_i = \frac{1}{n_i} \sum_{x \in D_i} x$$ Sample mean for cluster D_i

Sum of squared errors:

$$J_e = \sum_{i=1}^{c} \sum_{x \in D_i} \| x - M_i \|^2$$

Using J_e results well for compact clusters.

Basic ISODATA Algorithm (k-means)

Assume that there are k categories

1. Choose k arbitrary points in space as cluster centers.

$$M_{10}, M_{20}, \ldots\ldots\ldots\ldots, M_{k0}$$

2. Assign samples to their nearest cluster.

3. Update M's. If any means changed value, go to 2. Otherwise stop.

May fall into local minimum. $\min_i \|x - M_i\|^2$

Versions:

If the number of clusters is not specified, a criterion (threshold) for eliminating some clusters are used, such as:

If $Ji > \theta$ eliminate cluster i and merge it with another

- Distances between cluster centers may be computed to determine how to merge..

Iterative Optimization

Trying all combinations to find minimum Je – Exhaustive search is expensive

Iteratively – remove samples from one cluster to the other until a minima is obtained.

1. Arbitrarily select an initial grouping into a clusters. Find sample means

$$M_{10}, M_{20}, \ldots\ldots\ldots, M_{c0}$$

2. Select an arbitrary point and move it to another cluster if that change reduces J_e.
3. Iterate until no change in J_e occurs.

For the sample X^* to be removed from c_i and go to c_j, the mean m_j changes as follows:

$$m_j^* = \frac{1}{n_j+1}\left(\sum_{x\in C_j} x + x^*\right) = \frac{1}{n_{j+1}}\left(n_j m_j + x^*\right) = \frac{1}{n_j+1}\left(n_j m_j + m_j - m_j + x^*\right)$$

$$= \frac{1}{n_j+1}\left[\left(n_j+1\right)m_j + x^* - m_j\right]$$

$$\boxed{m_j^* = m_j + \frac{x^* - m_j}{n_j+1}}$$

Similarly

$$\boxed{m_i^* = m_i - \frac{x^* - m_i}{n_i-1}}$$

Here, J's can be updated accordingly

$$J_i^* = J_i - \frac{n_i}{n_i - 1}\|x - M_i\|^2$$

$$J_j^* = J_j + \frac{n_j}{n_j + 1}\|x - M_j\|^2$$

Iterative Optimization Cont.

Step 1: Start with arbitrary clusters.

Step 2: Move a sample from one cluster to another one with minimizing S.

$$S = \sum_{i=1}^{n} s_i \qquad s_i / n_i$$ sample variance

Sample X goes from cluster i to j if:

$$\frac{n_i}{n_i - 1}\|X - M_i\|^2 > \frac{n_j}{n_{j+1}}\|X - M_j\|^2$$

For many clusters, step 2:

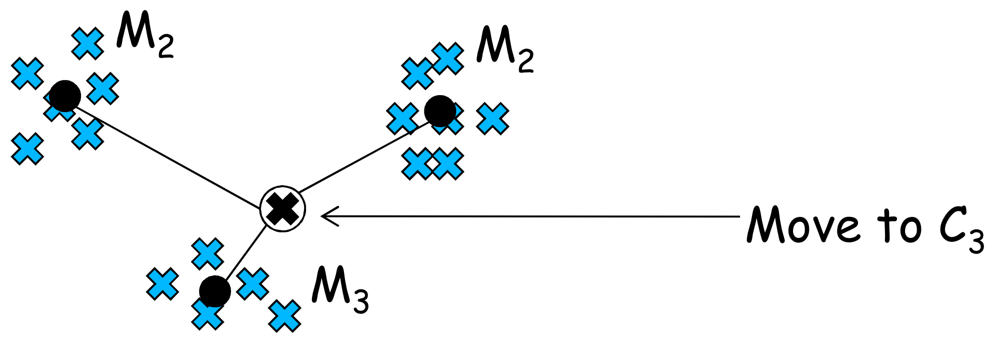
Find

$$\rho_j = \begin{cases} \dfrac{n_j}{n_j + 1} \left\| x - M_j \right\|^2 & i \neq j \\ \dfrac{n_i}{n_j} \left\| x - M_i \right\|^2 & j = i \end{cases}$$

Transfer X to C_k if $\rho_k \leq \rho_j$ for all j.

This turns out to be:

Measure the normalized distances to other sample means. Assign it to the cluster with shortest distance.



When n_i, n_j large, converts into so called "k-means" or "ISODATA" algorithm where sample counts are not taken into account.

■ **Algorithm 3.** **(Basic Iterative Minimum-Squared-Error Clustering)**

```
 1 begin initialize $n, c, \mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$
 2         do randomly select a sample $\hat{\mathbf{x}}$
 3            $i \leftarrow \arg\min_{i'} \|\mathbf{m}_{i'} - \hat{\mathbf{x}}\|$     (classify $\hat{\mathbf{x}}$)
 4            if $n_i \neq 1$ then compute
 5                $\rho_j = \begin{cases} \frac{n_j}{n_j+1}\|\hat{\mathbf{x}} - \mathbf{m}_j\|^2 & j \neq i \\ \frac{n_j}{n_j-1}\|\hat{\mathbf{x}} - \mathbf{m}_i\|^2 & j = i \end{cases}$
 6                if $\rho_k \leq \rho_j$ for all $j$ then transfer $\hat{\mathbf{x}}$ to $\mathcal{D}_k$
 7                   recompute $J_e, \mathbf{m}_i, \mathbf{m}_k$
 8           until no change in $J_e$ in $n$ attempts
 9        return $\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_c$
10   end
```

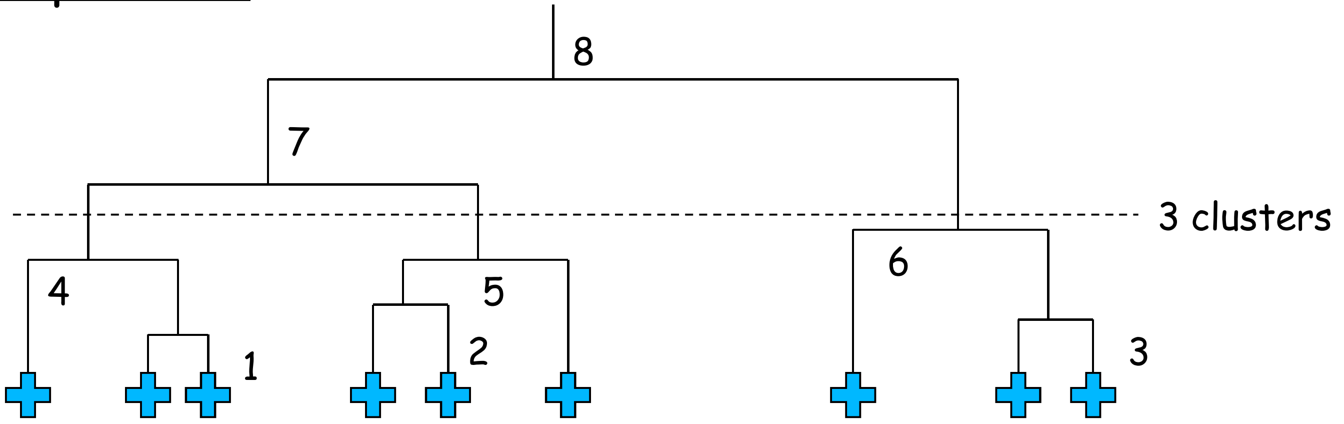
Hierarchical Clustering

A different approach to clustering.

Hierarchy in living species

- Each species is a class by itself.
- Combine the ones that are closest
- Continue combining until the number of clusters are what is desired or a criterion is satisfied.

1-d problem



Issues:

- How do we measure distance between clusters?
- When do we stop?
- Do we start from bottom or from top?

Basic Bottom-up Hierarchical Clustering Algorithm

1. Start taking each sample as a cluster. n=m (n of samples)
2. Measure $d_{i,j}$ – distance between clusters D_i and D_j. Join two clusters D_i and D_k for which

$$d_{i,k} = \min d_{i,j}$$

n=n-1;

3. If n<k stop. k:number of desired clusters

Else go to 2.

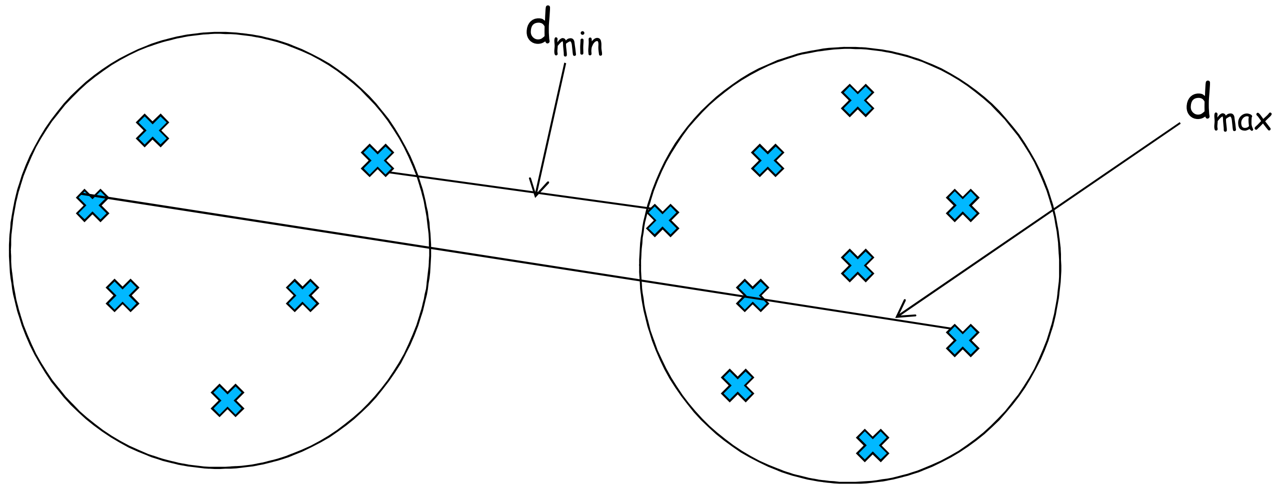
d_{ij} can be defined in many ways.

$$d_{\min_{i,j}} = \min \| x_s - x_z \|$$

where s is in i and z is in j.

$$d_{avg_{i,j}} = \frac{1}{n_i n_j} \left(\sum \sum \| x_s - x_z \| \right)$$

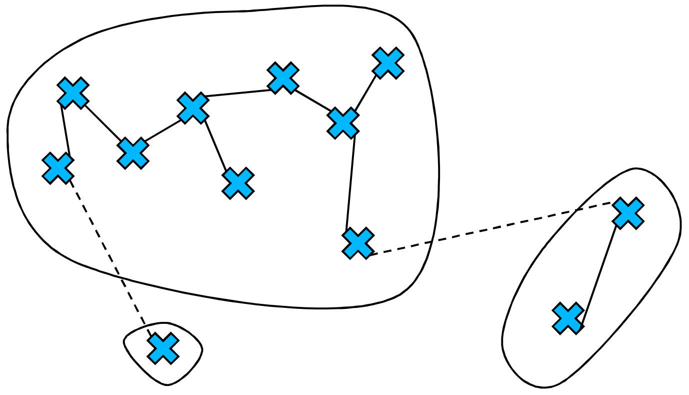
$$d_{\max} = \max \| x_s - x_z \|$$



<u>Example</u>

Apply hierarchical clustering with d_{min} to below data where c=3.

Nearest neighbor clustering



will form elongated clusters!

Computational Complexity of Hierarchical Clustering

Assume

n – samples

d – dimension

c – clusters to be formed.

Use $d_{\min}(D_i, D_j)$

- To find the nearest clusters at level $\hat{c}$,
- $\theta(d)$ for each pair
- n(n-1) pairwise calculations, $\theta(n^2)$ (each sample with all others except itself)
- Finding the minimum distance pair

overall $\propto \theta(cn^2d)$

NN clustering algorithm and minimal spanning trees

If we continue with NN to end up with a single cluster, we obtain "minimal spanning tree".

Spanning tree: a graph with no loops that will contain a path between any two points.

Minimal spanning tree: a spanning tree with a minimum total length (length : sum of path lengths)

Top-down Clustering: Graph theoretical approach so a top-down approach will

- Obtain the minimal spanning tree using a fast algorithm
- Remove the longest edge
- Continue removing until desired number of clusters are reached.

Solution with iterative clustering algorithm

Step 2 of Bottom-up hierarchical clustering algorithm:

Join clusters i and j if that will result with smallest increase in

$$d(Di,Dj) = \sqrt{\frac{n_i n_j}{n_i + n_j}} \left\| M_i - M_j \right\|^2$$

This is the same as joining clusters with closest means, normalized with the number of samples in each cluster.

One way to use above algorithm is: to obtain an initial partitioning for iterative optimization.