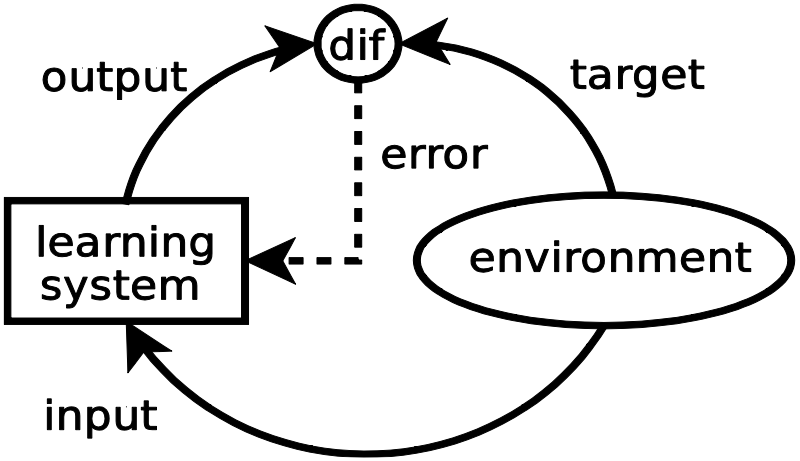
Unsupervised learning: SOM
Nonparameteric models:
K-means clustering and k-nearest neighbours

Lubica Benuskova

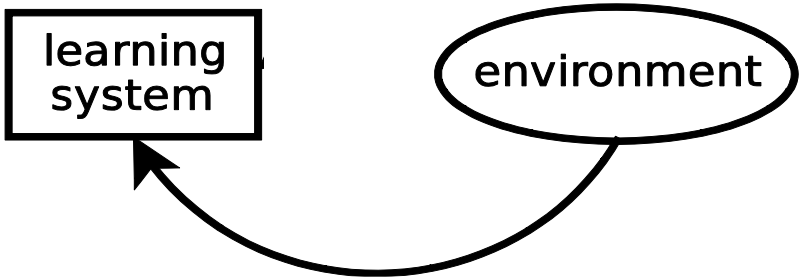
AIMA 3rd ed. Ch. 18.8 – 18.8.2

3 ways of learning in the learning module

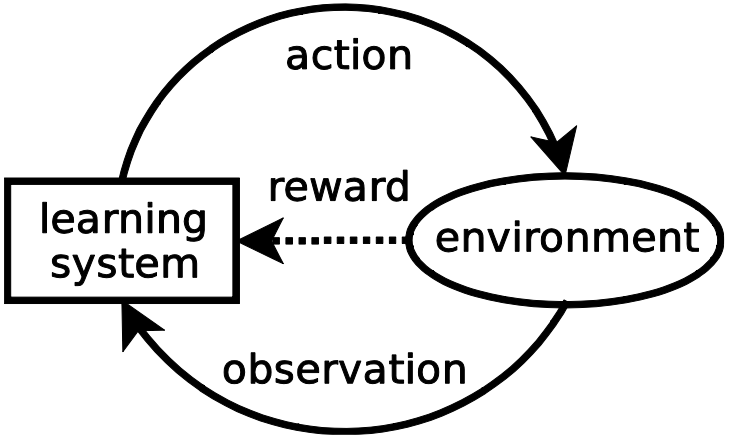
supervised (based on error)
- Target output is known



unsupervised (self-organized)
- Target output is not known



reinforcement learning (based on reward)
- Similar to supervised



Self-organizing map (SOM)

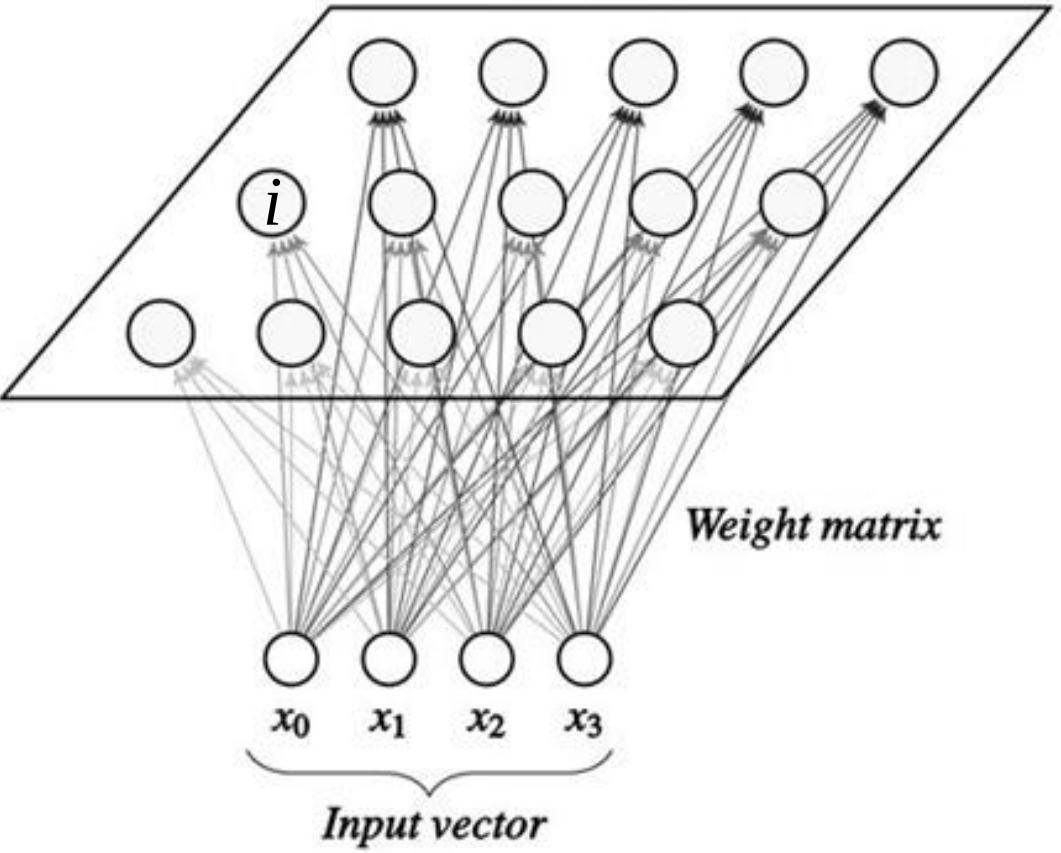
- SOM is a type of artificial neural network (ANN) that is trained using **unsupervised learning** to produce a low-dimensional (typically two-dimensional), discretized representation of the training samples, called **a map**.

- SOM differs from other ANN as it uses only the input data as opposed to error-correction learning (such as backpropagation with gradient descent).

- It uses the so-called **neighbourhood function** to preserve the **topological properties** of the input space.
  - This makes SOMs useful for visualizing low-dimensional projection of high-dimensional data.

- SOM was introduced by the Finnish professor Teuvo Kohonen in the 1980s and is sometimes called a Kohonen map or Kohonen network.

Self-Organizing Map (SOM): architecture



Neurons are not connected

m linear neurons in the single layer:

$$o_i = \sum_{j=1}^{m} w_{ij} x_j = \mathbf{w}_i \, \mathbf{x}$$

$$\mathbf{x} = (x_1, \ldots, x_j, \ldots, x_m)$$

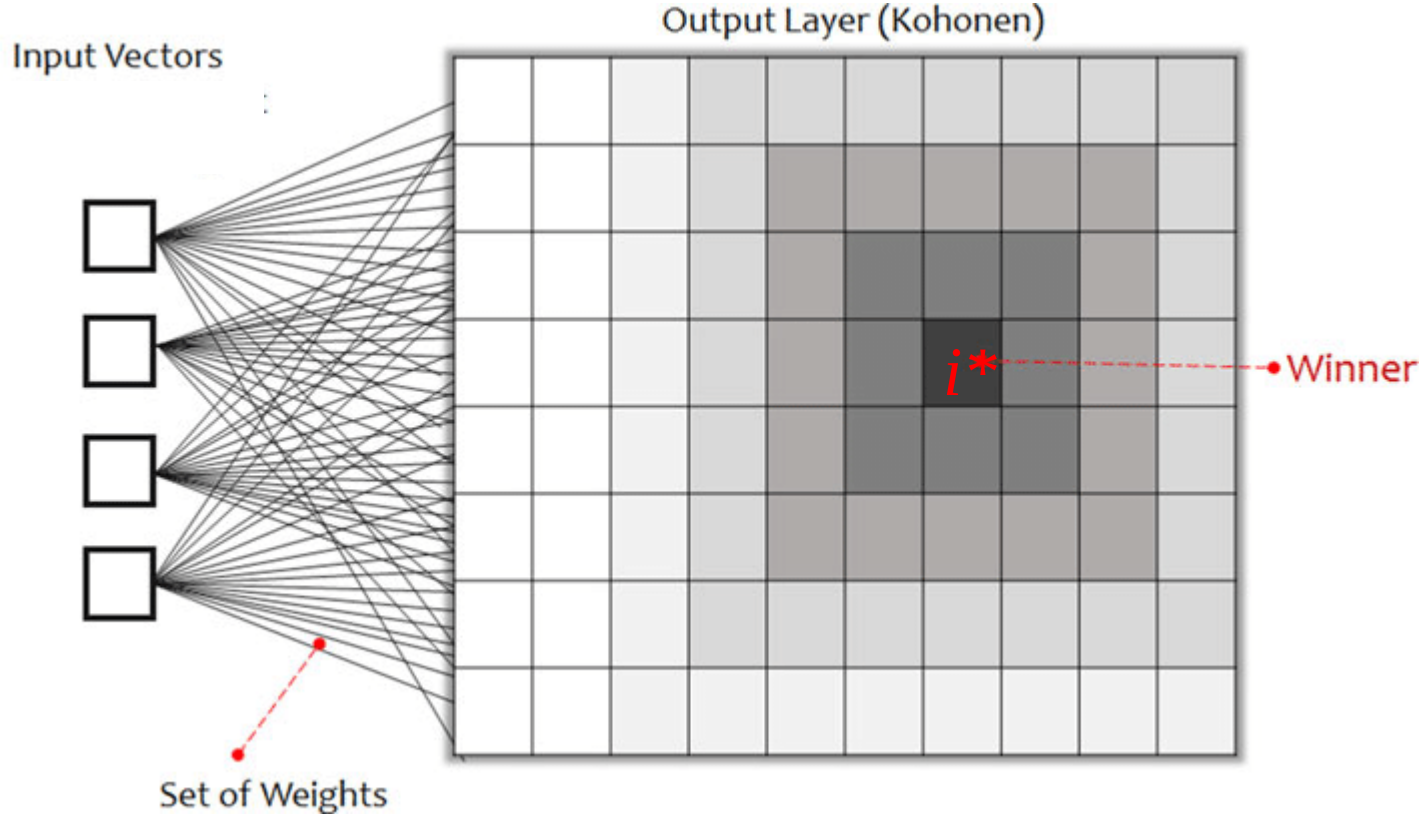
- Input patterns = vectors of real values. They feed each neuron.
- The weight vector of neuron *i* : $\mathbf{w}_i = (w_{i1}, \ldots, w_{ij}, \ldots, w_{im})$

SOM training: competition phase

- Training set consists of input vectors only, which are presented in random order:

$$A_{train} = \{\mathbf{x}^1, \mathbf{x}^2, ..., \mathbf{x}^p, ..., \mathbf{x}^P\}$$

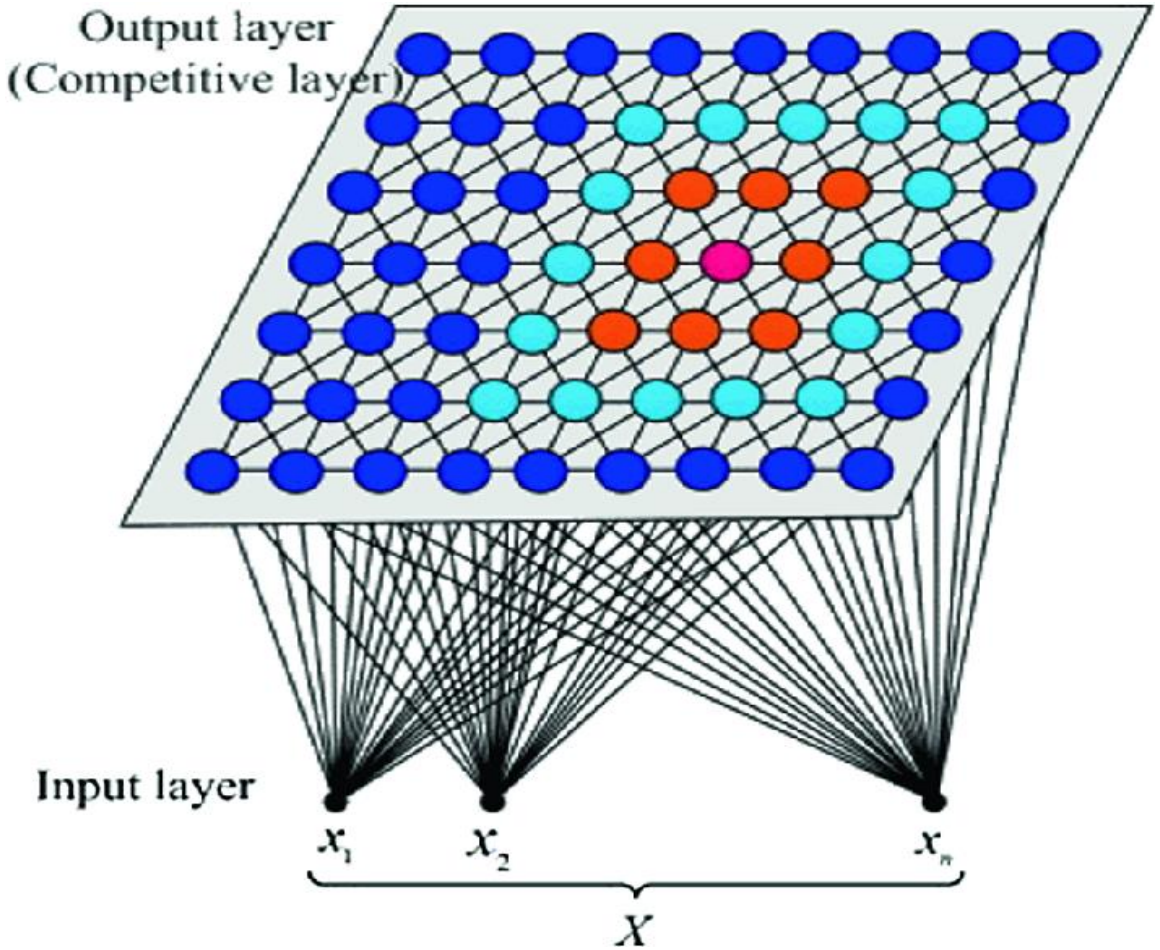
- For each input vector in the training set, we find **a winner neuron**
  - According to maximal dot product: $i^* = \text{argmax}(\mathbf{w}_i \, \mathbf{x})$
  - Or according to minimal Euclidean distance: $i^* = \underset{i}{\text{argmin}} \, d_{\text{E}}(\mathbf{w}_i, \mathbf{x})$



SOM training: weight update & cooperation

- The weights of the winner i^* and its neighbours in $N_{i^*,i}$ are updated in each iteration t after the presentation of the input pattern $\mathbf{x}$:

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) \cdot N_{i^*,i}(t) \cdot [\mathbf{x}(t) - \mathbf{w}_i(t)]$$

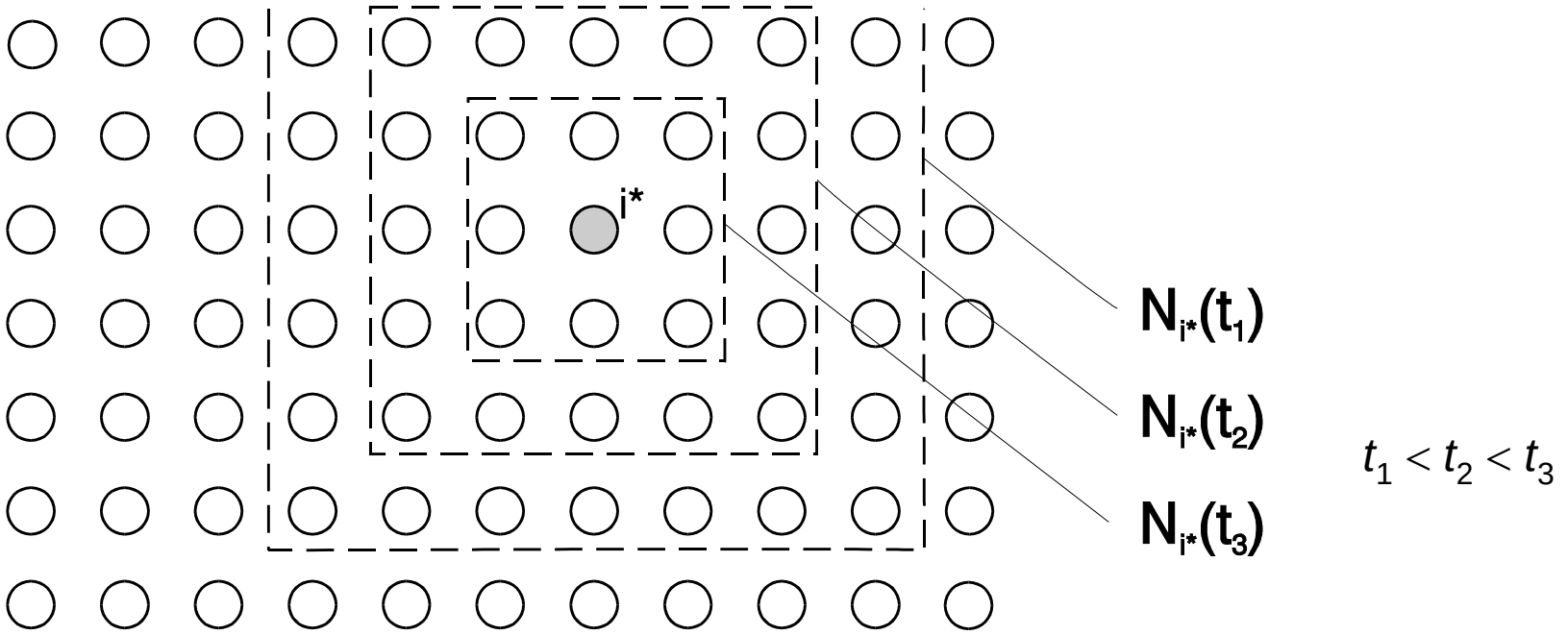


The weight vectors of each $i \in N_{i^*,i}$ move closer to the current input $\mathbf{x}$

Rectangle neighbourhood function $N_{i^*,i}$

- Rectangle neighbourhood, i.e. based on Manhattan distance, between neurons $d_{\mathrm{M}}(i^*,i)$:

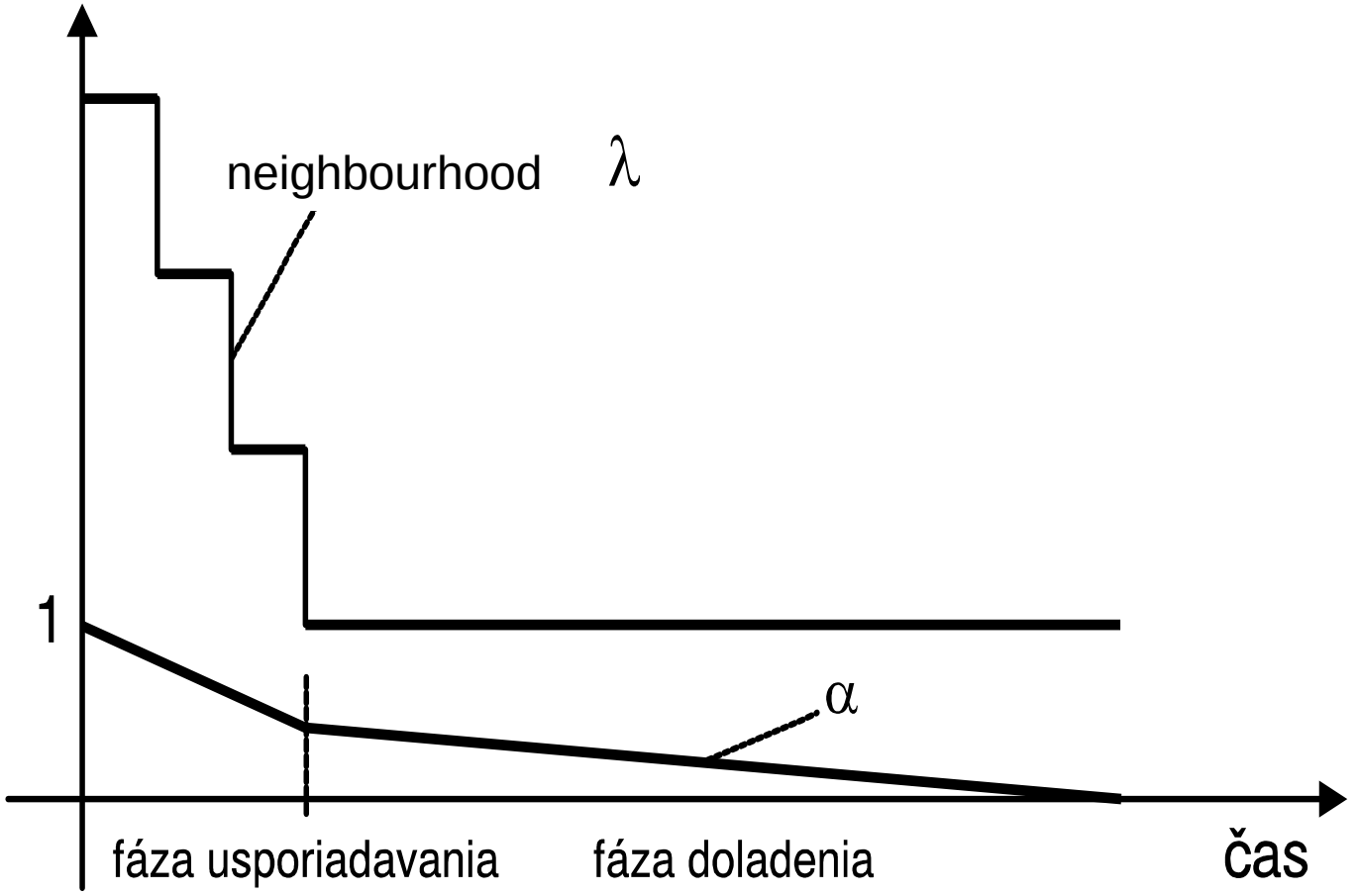
$$N_{i^*,i}(t) = \begin{cases} 1 & \text{if } d_{\mathrm{M}}(i^*,i) \le \lambda(t) \\ 0 & \text{otherwise} \end{cases}$$



During training the neighbourhood shrinks

Parameters of learning are not constant

- The neighbourhood parameter $\lambda(t)$ and the value of learning speed $\alpha(t)$ decrease in time:



Kohonen algorithm for training SOM

1. Choose initial $\alpha(0) = 1$, initial $\lambda(0)$ and initial $\mathbf{w}(0) \in [-0.5, 0.5]$. Set iteration counter $t = 0$, pattern $p = 0$, epoch counter $e = 0$ and maximal number of iterations t_{max}.
2. For randomly chosen **pattern x**(p) calculate outputs of all neurons.
3. Find the winner i^*, i.e. the neuron whose weight vector is closest to the current input vector.
4. Move the weights of the winner and its neighbours towards the current input vector:

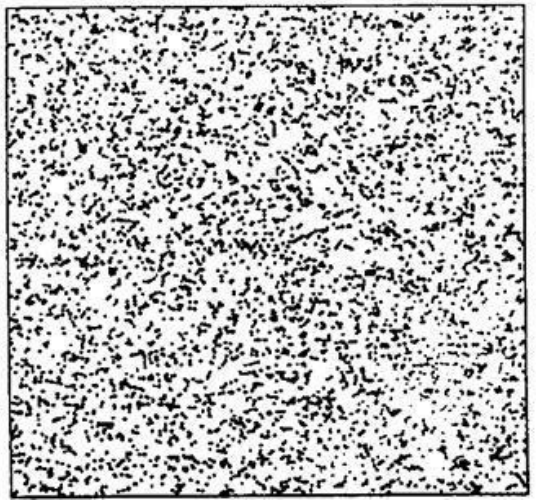
$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \alpha(t) \cdot N_{i^*,i}(t) \cdot [\mathbf{x}(t) - \mathbf{w}_i(t)]$$

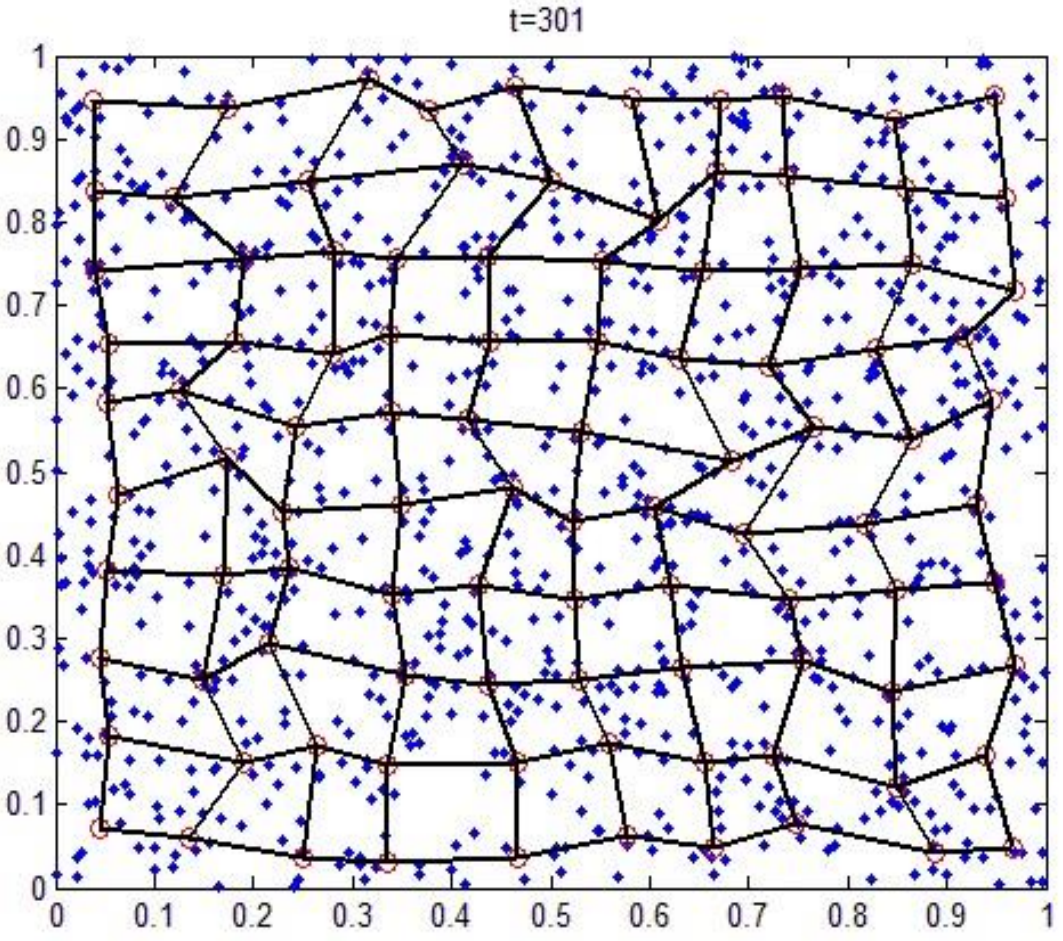
5. If $t = t_{max}$ stop training. Otherwise, put $t = t+1$, update the learning speed α and the size of the neighbourhood λ and go to step 2.

SOM: the weight plot

$$\mathbf{x} = (x_1, x_2)$$

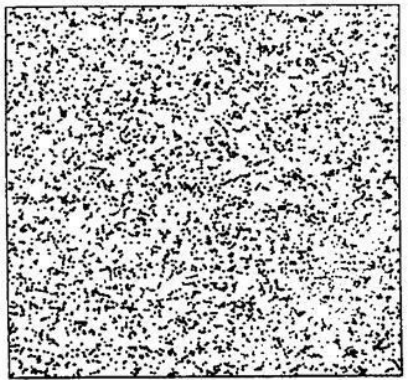
$$\mathbf{w}_i = (w_{i1}, w_{i2})$$





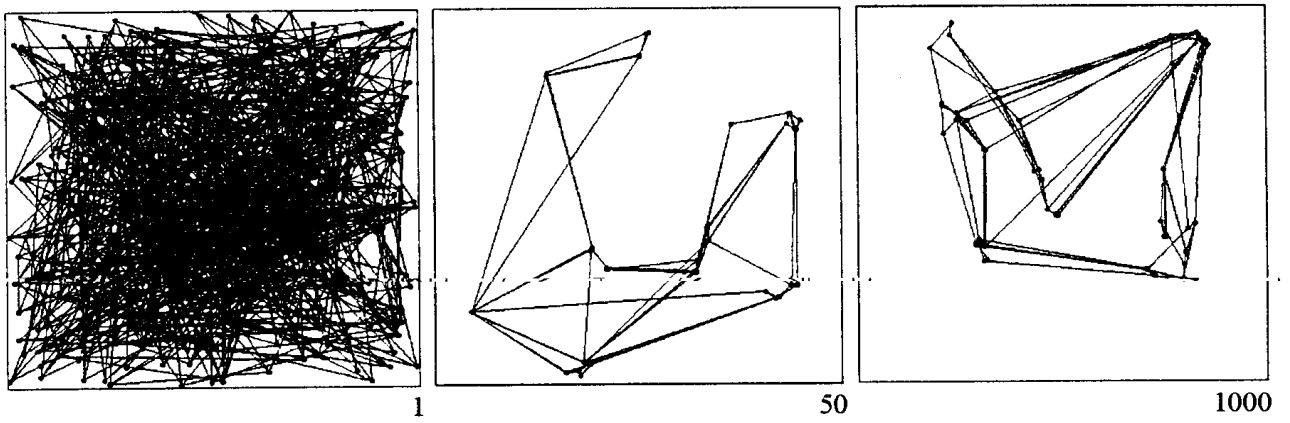
- The input is a 2D vector, the x and y coordinates of points
- The net has 10 x 10 neurons
- Each neuron has a 2D weight vector

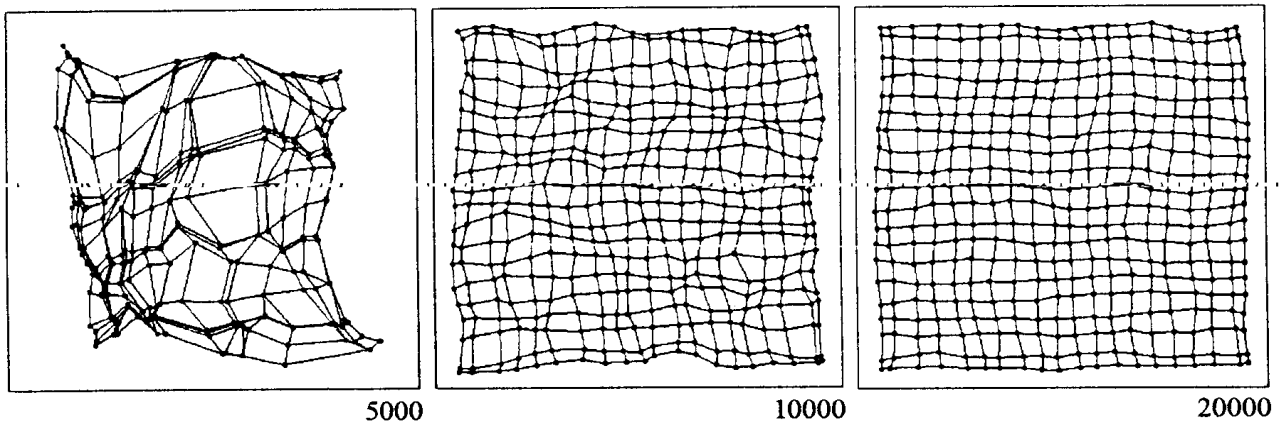
SOM lattice of 20x20 neurons: weight evolution



Input $\mathbf{x} = (x_1, x_2)$ is uniformly random distribution of points covering the 0-1 square

Initial random weights $\mathbf{w}=(w_1,w_2)$

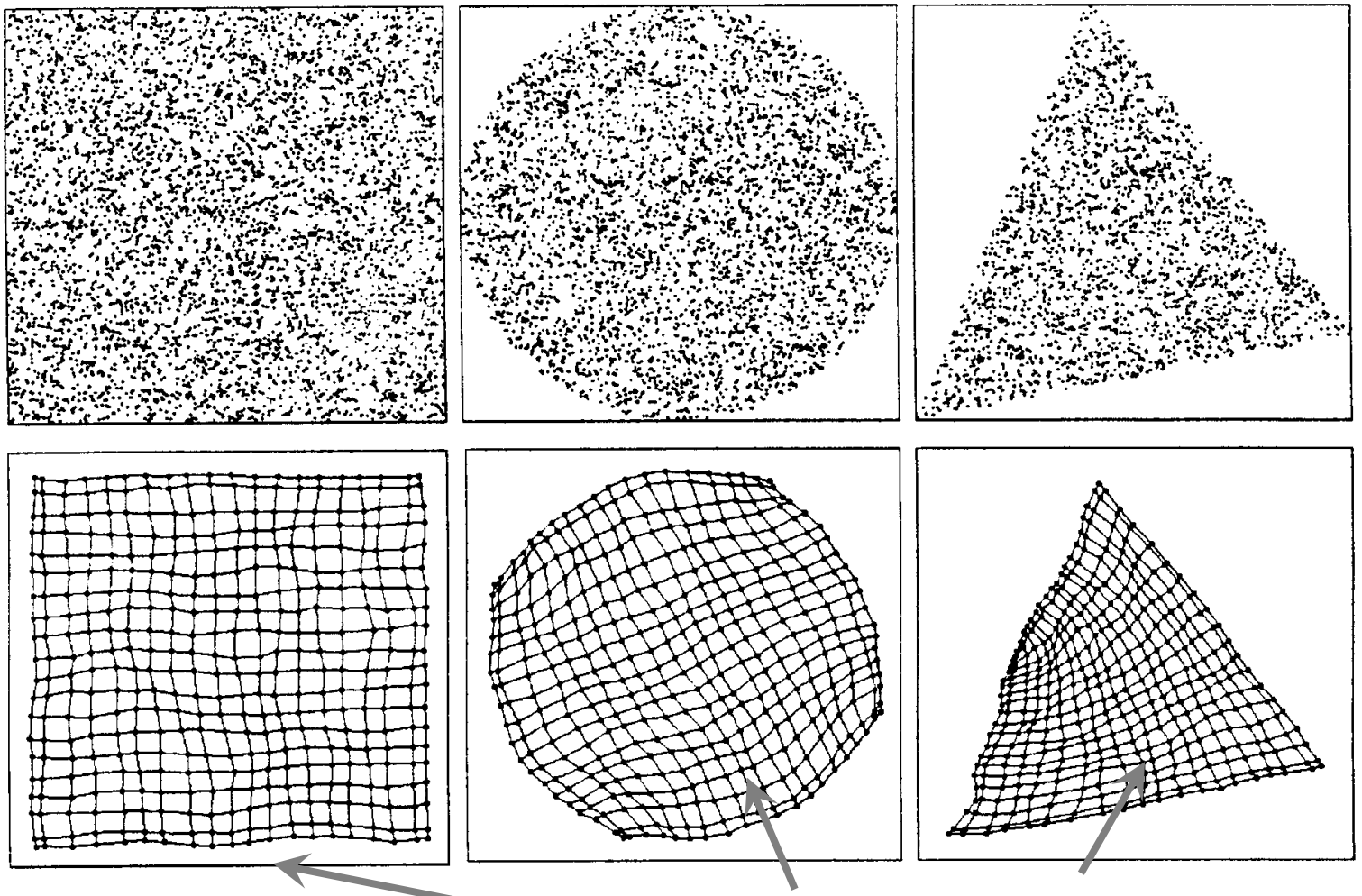




Final weights coordinates

Examples of square, circular and triangle 2D distributions

Input $\mathbf{x} = (x1, x2)$

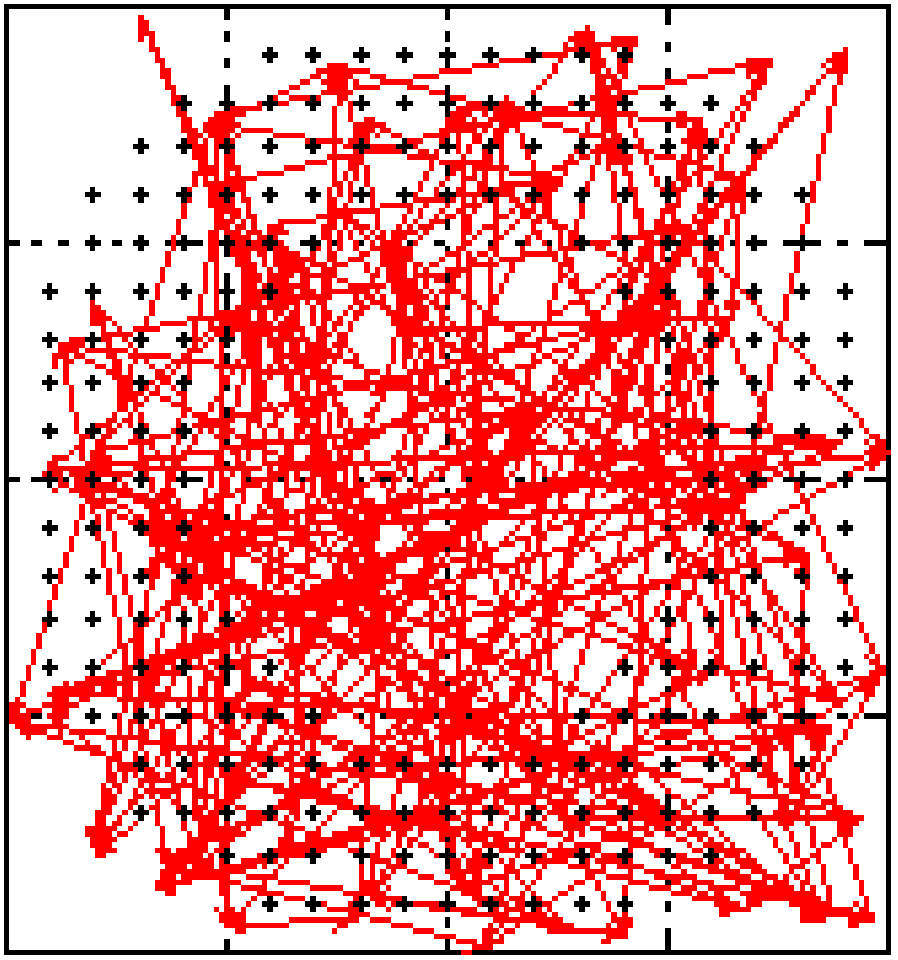


Final weights $\mathbf{w} = (w1, w2)$

SOM animation

- The goal is to learn a topologically faithful mapping from input data points to regular lattice of weights of 10x10 neurons.

Black = input data points



Red = weight coordinates

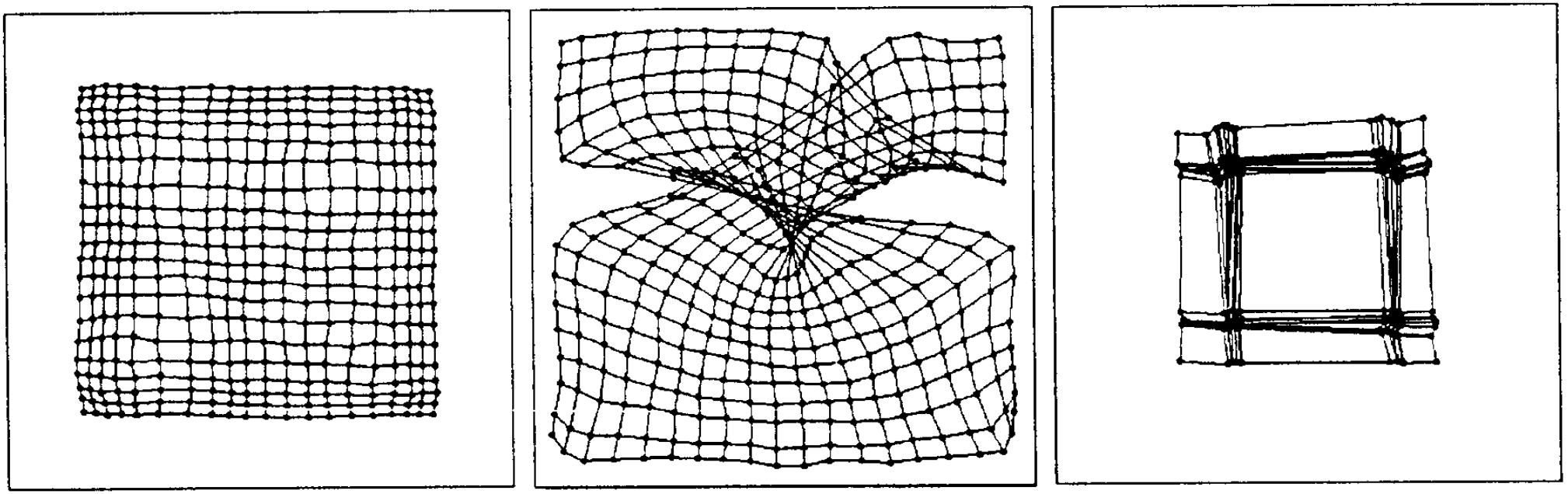
Source: B. Hammer, U. of Osnabrueck, Germany

Examples of bad training

a) Insufficient weight expansion (α decreases too fast)

b) Butterfly effect (α decreases too fast compared to the decrease of the diameter of neighbourhood λ)

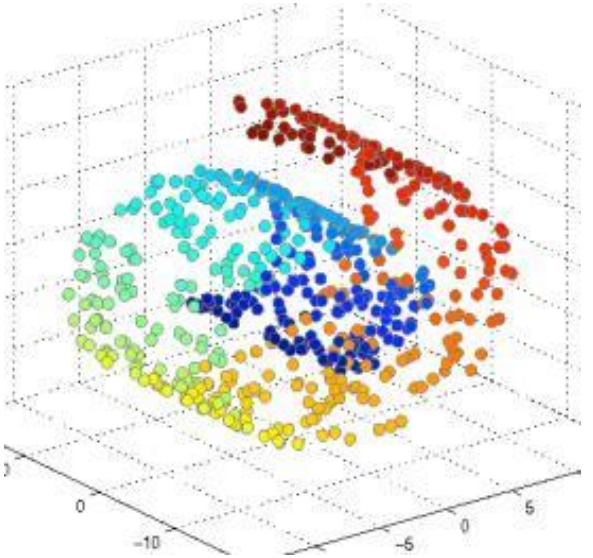
c) Pinch effect (α decreases too slowly)

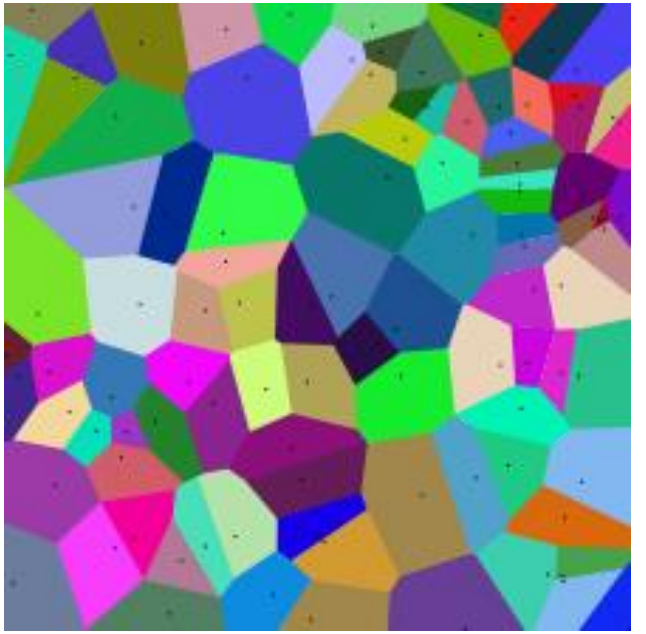


(a) (b) (c)

Applications of SOM

- Data compression and data dimensionality reduction (data projected to the 2D or 3D lattice of weights).
- *Visualization* of the data in the data mining.
- ***Clustering***, vector quantisation: clusters of data points are replaced by prototype vectors (winner weights) – in 2D the centers *C* of the so-calledVoronoi mosaic.
- *Voronoi cell V(C)* consists of all points closer to *C* than to any other center. The borders of the Voronoi mosaic are all the points in the plane that are equidistant to two centers.





Nonparametric versus parametric models

- Regression and neural networks use the training set to estimate a fixed set of parameters (i.e. coefficients of the function, weights) . Afterwards, all we need for generalization is this set of parameters, training data can be discarded.

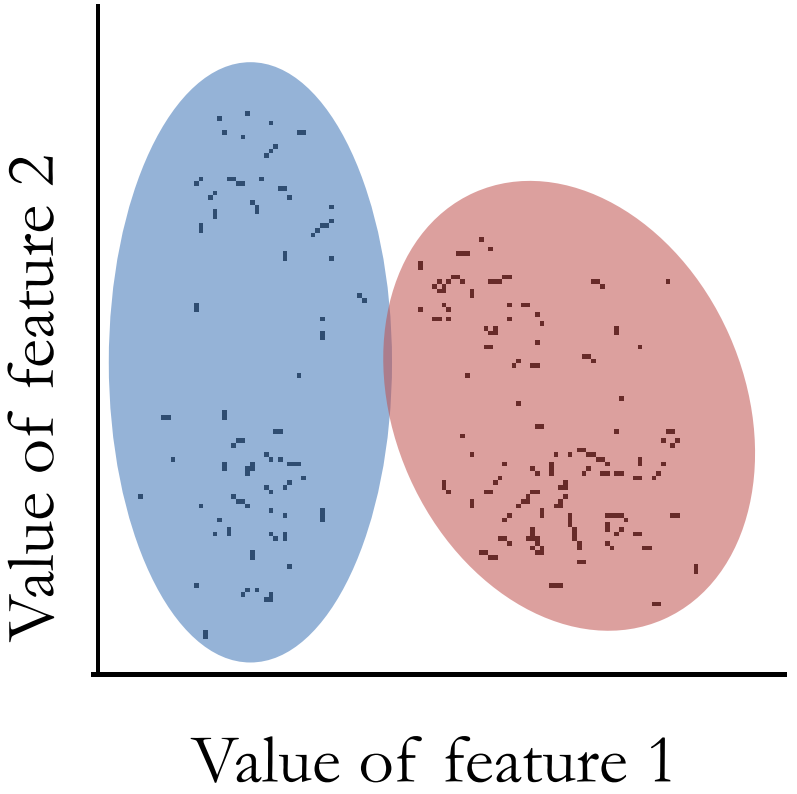
- A learning model that summarizes data with a set of parameters is called a **parametric model.**

- Suppose we can pose a hypothesis that retains within itself all the training examples and uses all of them to predict the next example. Such a hypothesis (model) would be **nonparametric** because the effective number of parameters is unbounded – it grows with the number of training examples.

Clustering: definition

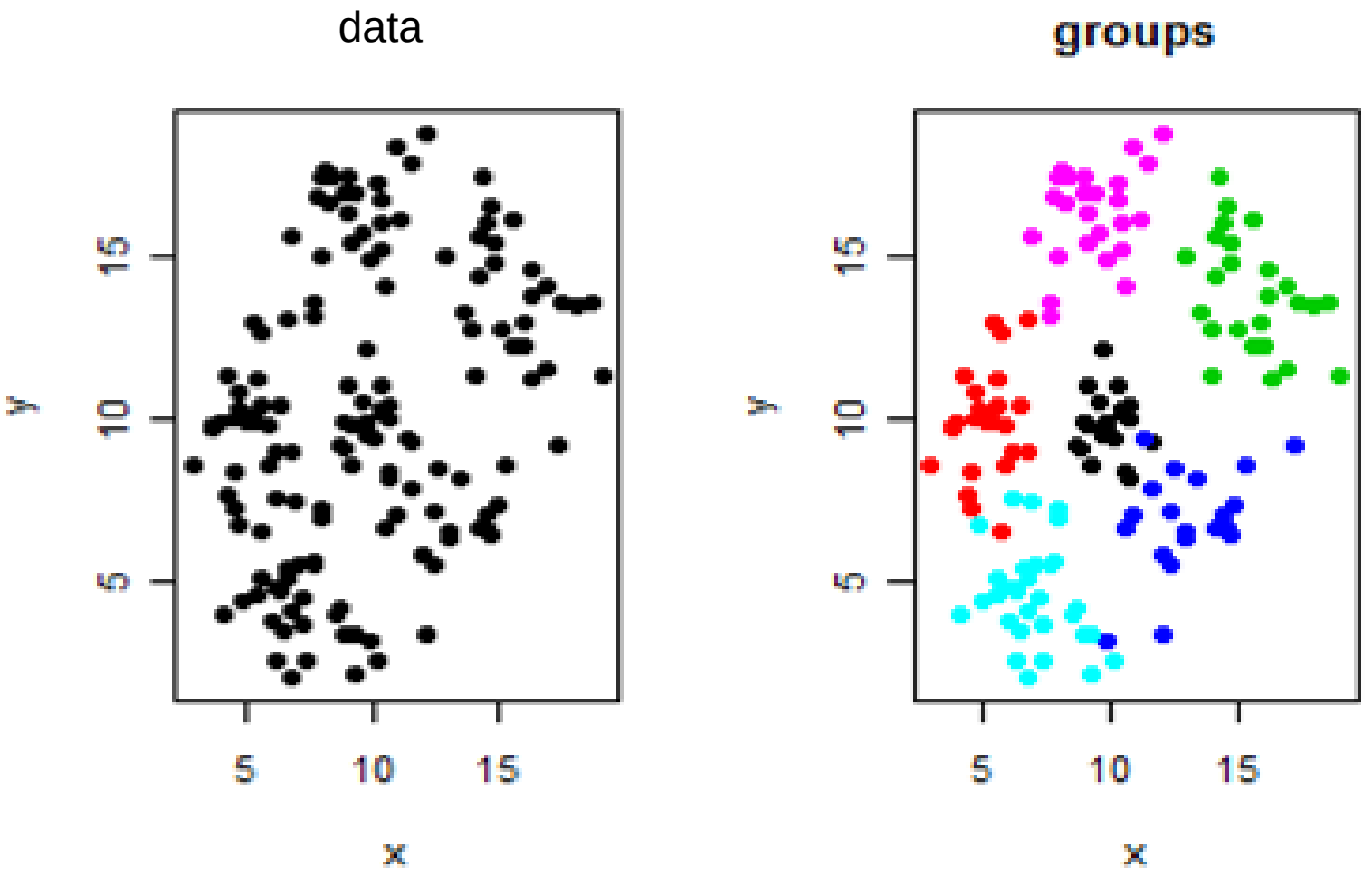
- Clustering is a nonparametric method, based only on the data themselves.
- Clustering provides a method for grouping objects based on some measure of similarity.
- The goal is to find groups of samples that represent a class of similar objects, e.g. cluster of all non-spam emails versus a cluster of all spam emails.

Each point (vector) represents 1 sample.
Number of features $n = 2$.



In other words…

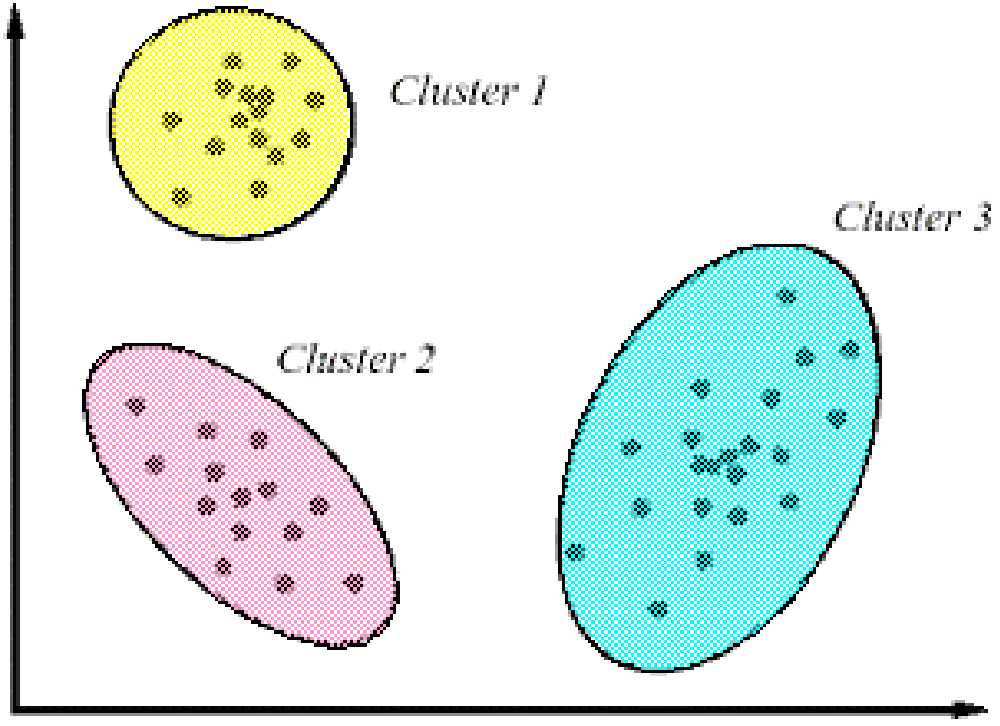
- Clustering means automatically dividing a collection of objects (samples, examples) into clusters (groups, classes), such that objects in the same cluster are *similar* in some sense.



Partitional clustering

- A **partition** of a set X is a division of X into K non-overlapping and non-empty partitions (clusters, blocks, groups) that cover all of X.

- Here is an example of partitional clustering. Note: the same data can be clustered into ***different number of clusters*** based on the clustering method and distance measure we use (more on this later):



Clustering: assumptions

- Objects are represented as vectors (i.e. arrays) of values (symbols).
- Vectors of values are represented as points in n-dimensional space where n is the vector dimension (= size of an array of characters).
- A quantitative scale (metric) is used to measure the closeness or similarity between vectors of values.
- Grouping is done on the basis of similarity (closeness) between objects.

Metric = distance or similarity measure

- Crucial step in most clustering is to select a distance or similarity measure, i.e. the **metric**, which will determine how the similarity of two elements is calculated.

- Distance or similarity measure will influence the shape of the clusters, as some elements may be close to one another according to one distance and farther away according to another.

- The choice depends on the measurement scale, and our heuristic knowledge. We can experiment with different measures to see, which one yields better results. The metric is always symmetric.

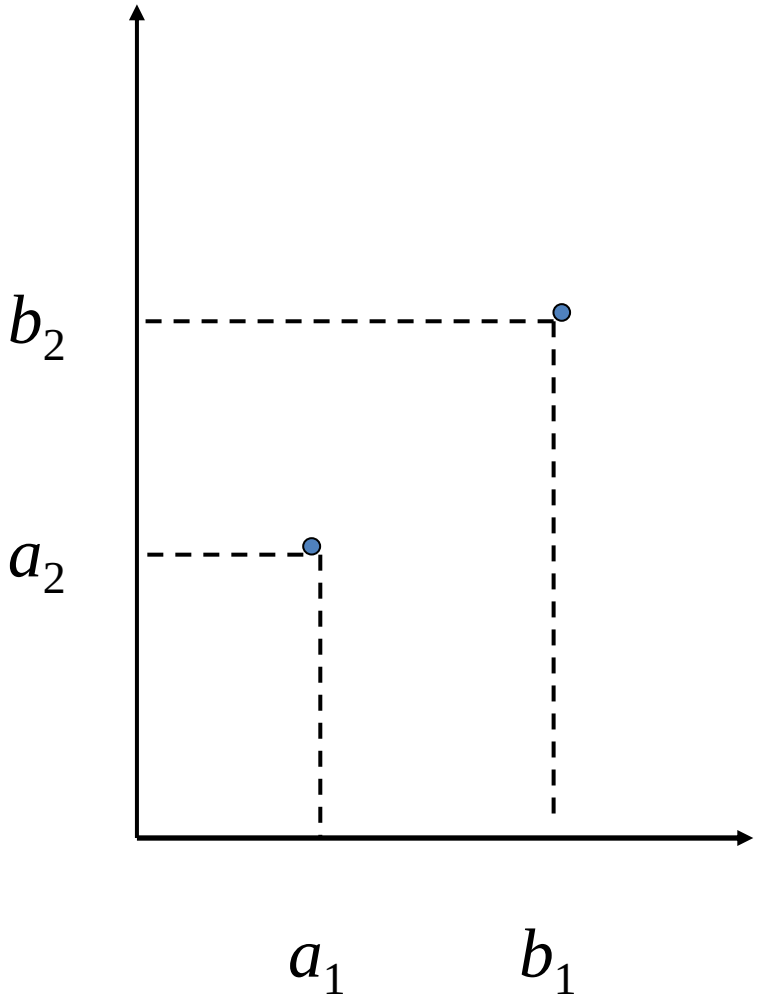
- Let's have two vectors (arrays) of character values a and b.
  - Vector $a = (a_1, a_2, \ldots, a_n)$ and vector $b = (b_1, b_2, \ldots, b_n)$.

Data points

- Let's have two vectors (arrays) of character values a and b.
  Vector $a = (a_1, a_2, \ldots, a_n)$ and
  vector $b = (b_1, b_2, \ldots, b_n)$, etc.

- So, we represent and treat our data as points in n-dimensional space.

- An example on the right features two data points in 2D space such $a = (a_1, a_2)$ and $b = (b_1, b_2)$.



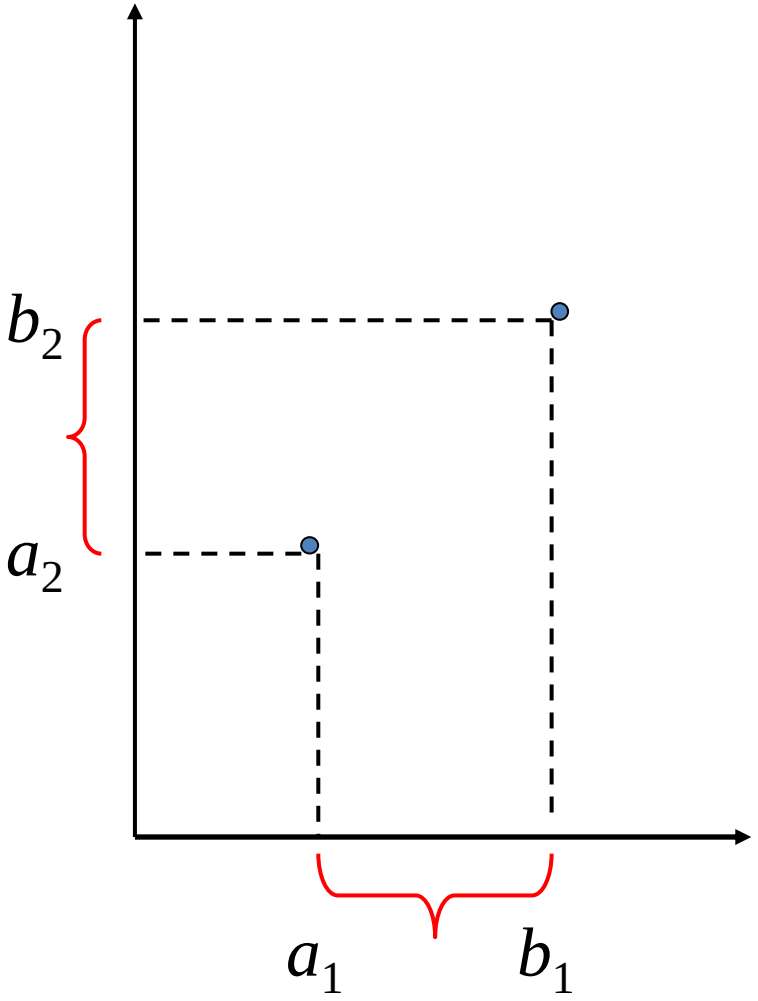
Manhattan distance

- ***Manhattan distance***: distance between two points or vectors is the sum of the absolute differences of their coordinates:

$$d_M(a,b) = |a_1 - b_1| + \ldots + |a_n - b_n|$$

- As an example, let $a = (a_1, a_2)$ and $b = (b_1, b_2)$. Then:

$$d_M(a,b) = |a_1 - b_1| + |a_2 - b_2|$$



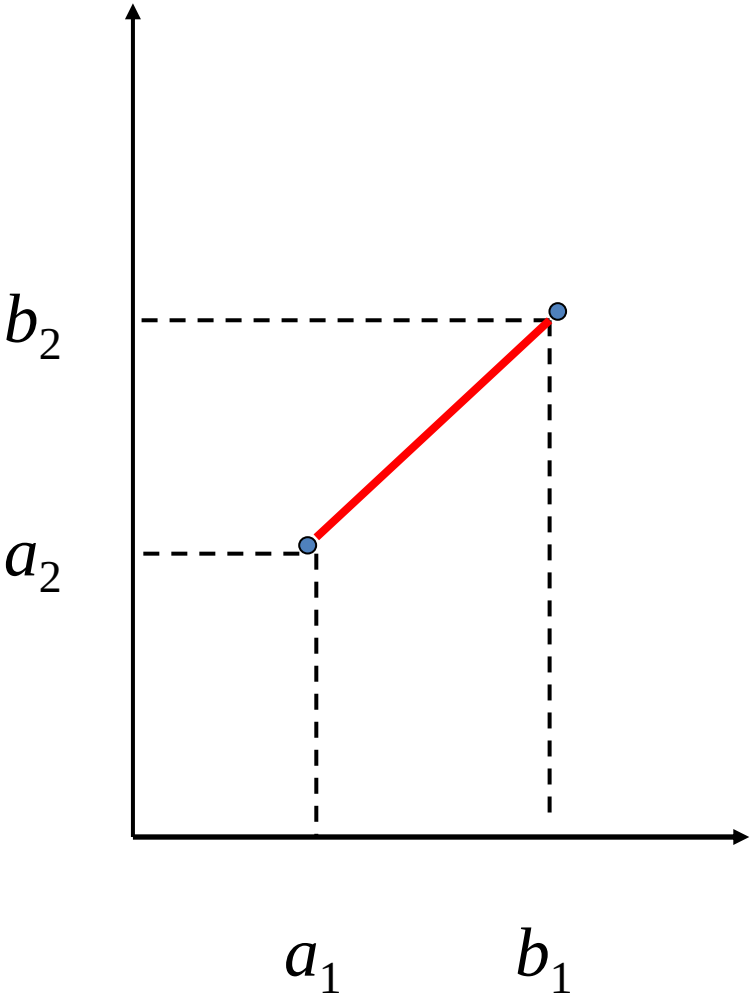
Euclidean distance

- ***Euclidean distance*** is the distance between two points in the Euclidean space:

$$d_E(a,b) = \sqrt{(a_1 - b_1)^2 + \ldots + (a_n - b_n)^2}$$

- As an example, let $a = (a_1, a_2)$ and $b = (b_1, b_2)$. Then (recall the Pythagoras theorem):

$$d_E(a,b) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2}$$



Hamming distance

- ***Hamming distance*** between two strings of equal length is the number of positions at which the corresponding symbols are different, i.e.:

$$d_H(a,b) = \sum_{k=1}^{n} count(a_k \neq b_k)$$

- Hamming distance measures the minimum number of substitutions required to change one string into the other, i.e. the number of changes that transformed one string into the other.

- Suitable for character strings (arrays) and for binary strings.

K-means clustering

- *K*-means clustering is a method of cluster analysis, which aims to partition N vectors into K clusters, in which each vector belongs to the cluster with the nearest centroid (i.e. center of a cluster).

- It attempts to find the centers of natural clusters in the data by the iterative refinement.

- It is the most popular "bottom-up" clustering algorithm.

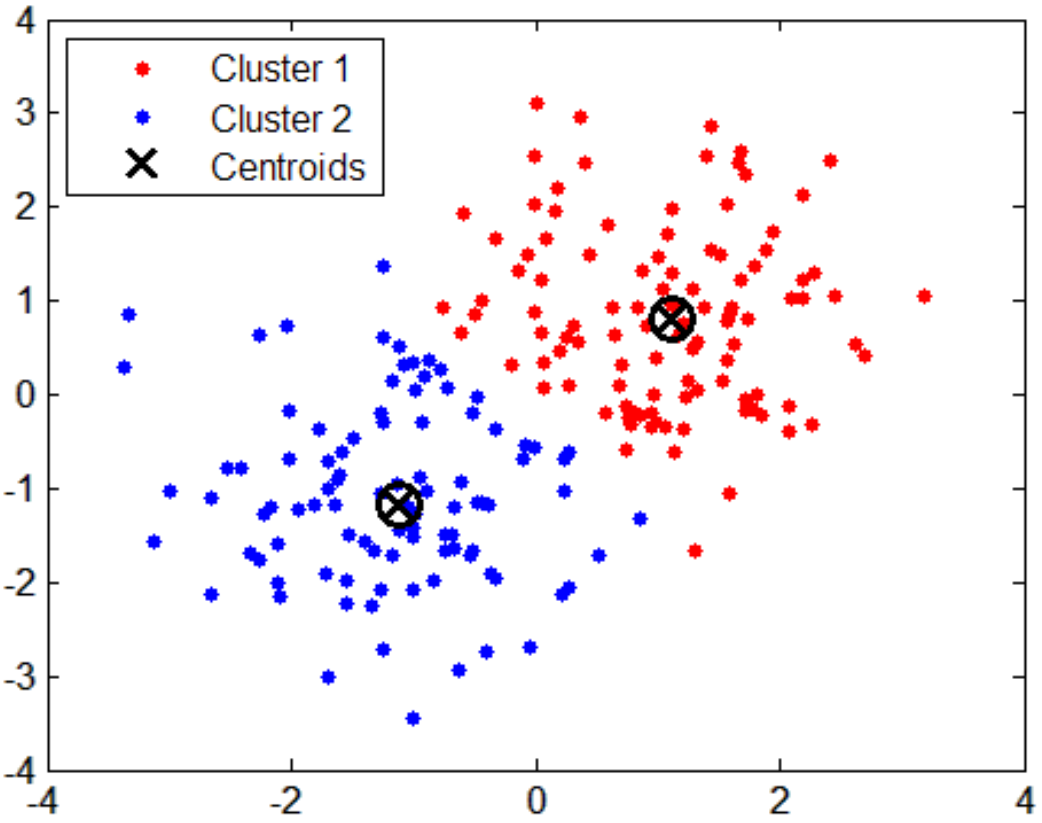
- If K is the number of clusters, n is the dimensionality of vectors, and N the number of vectors, the problem can be exactly solved in time $O(N^{nK+1} \log N)$.

K-means clustering

- Vector clustering for ***K*** centers:
  - *K*-means clustering partitions the data points into *K* clusters $\mathbf{C}=\{C_1,C_2,\ldots,C_K\}$ so as to minimize the sum of distances between all vectors belonging to the cluster and the centroid of the cluster

$$\underset{\mathbf{C}}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{\mathbf{x}_p \in C_k} d(\mathbf{x}_p, \mathbf{c}_k)$$

  - Cluster C_k has the centre $\mathbf{c}_k$



K-means clustering algorithm

- Randomly choose *K* input vectors from the data, which will be the starting centres of clusters (more on "randomly" in the lab).

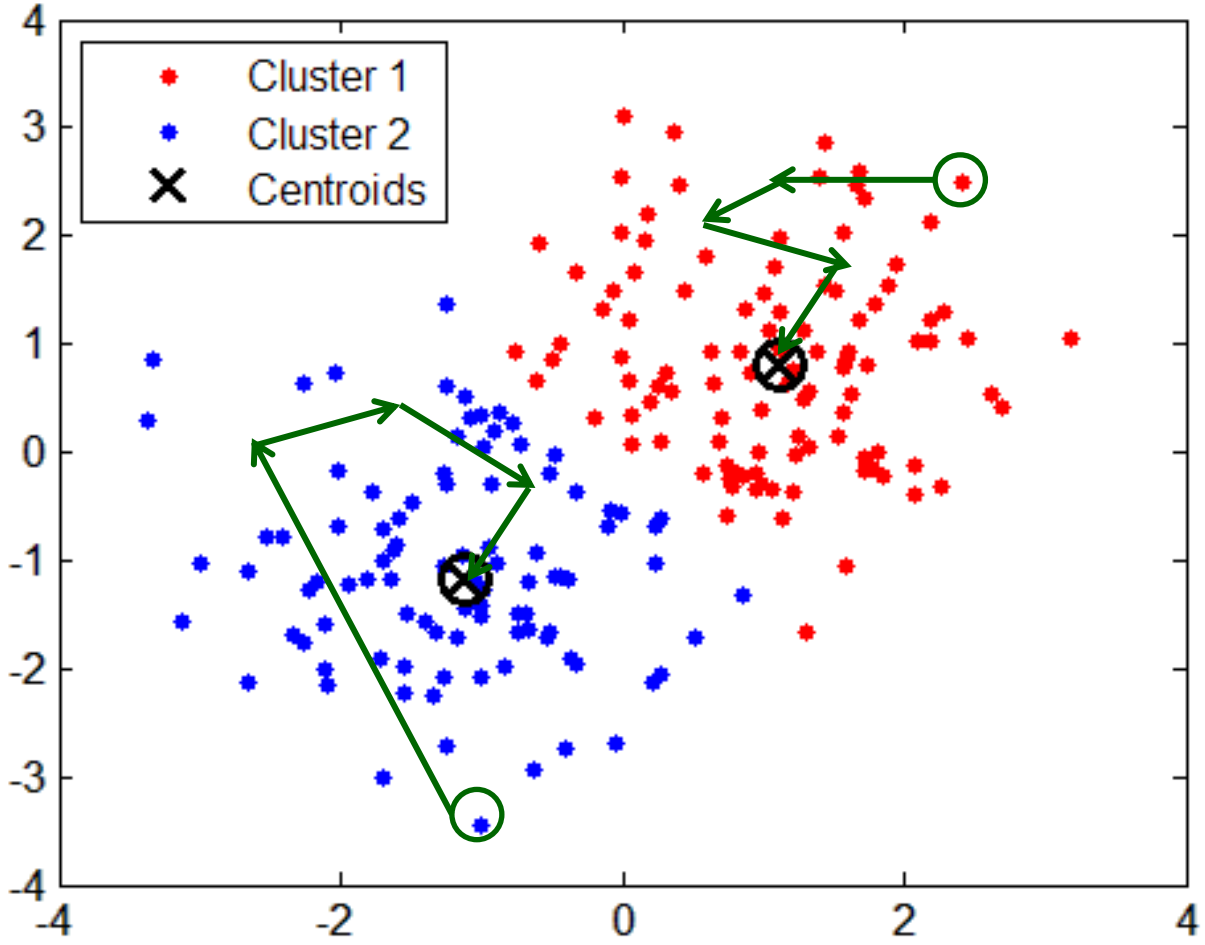
- Then for each input vector $\mathbf{x}_p$ do:
  - Find the centre $\mathbf{c}_k$, which is closest to $\mathbf{x}_p$
  - Create a new centre $\mathbf{c}_k$ by changing each coordinate i, so that:

$$c_{ki} \leftarrow c_{ki} + \alpha\left(x_{pi} - c_{ki}\right)$$

  - Stopping criterion: either we can make α linearly decreasing to zero or we can stop when centres do not move significantly.
  - In general, we choose initial $\alpha \in (0, 1]$.

Illustration of iterative refinement

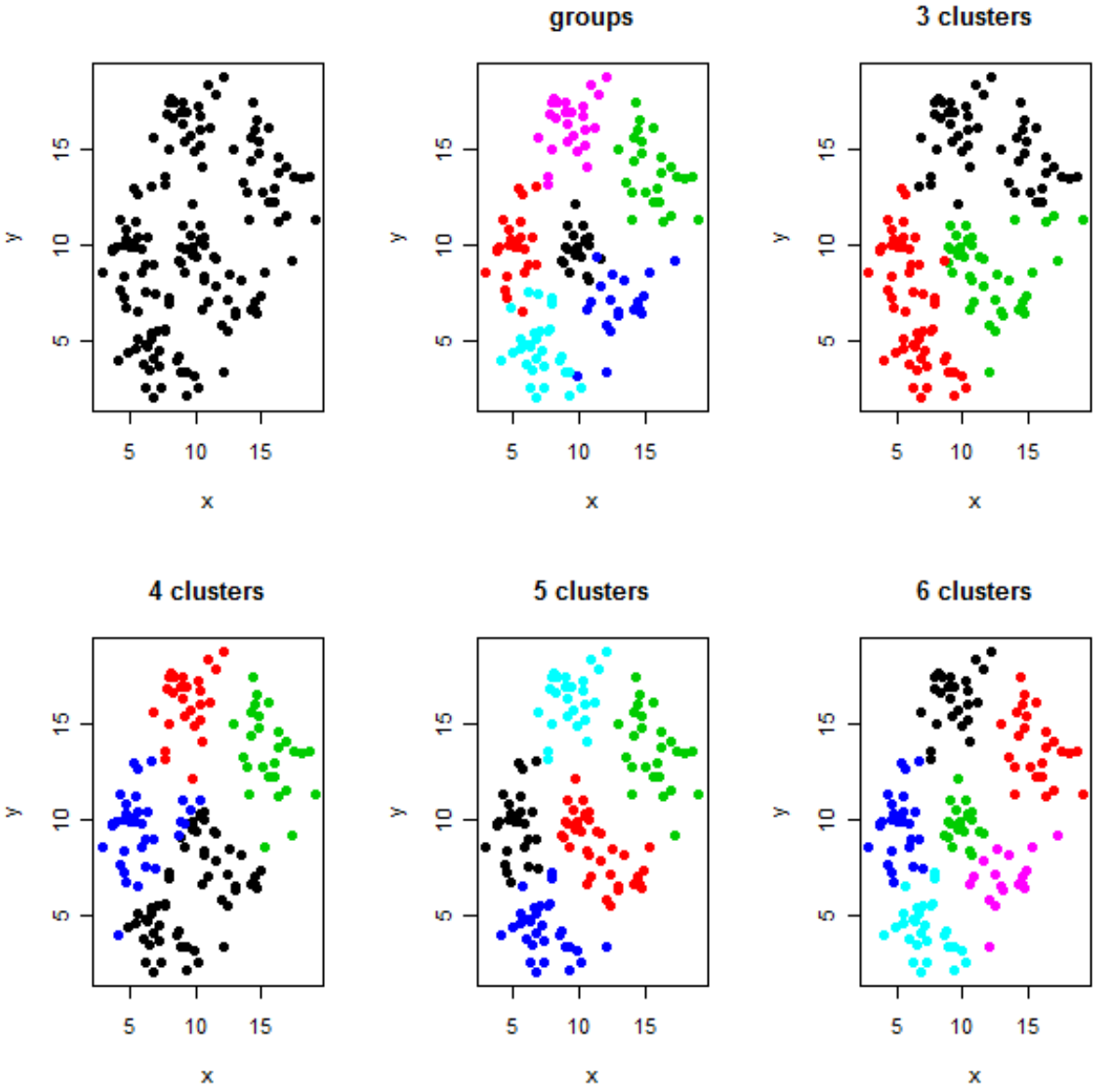
- We start with K randomly chosen initial centers.
- Then we randomly pick up **any** remaining data point and move the center towards it a bit. We do it for each of the K centers, until the centers do not move around the data anymore.



K-means clustering: evaluation

- *K*-means clustering is run for different number of clusters *K* as we do not know ahead how many natural clusters are in the data.
- There is no guarantee it will converge to the global optimum.
- The result depends on the initial random choice of centers of clusters. (There is an improved method called the K-means++)
- As the algorithm is usually very fast, it is common to run it multiple times with different starting centers.
- Particular value of $\alpha \in (0, 1]$ must be also found by experimentation.

How to choose *K* ?

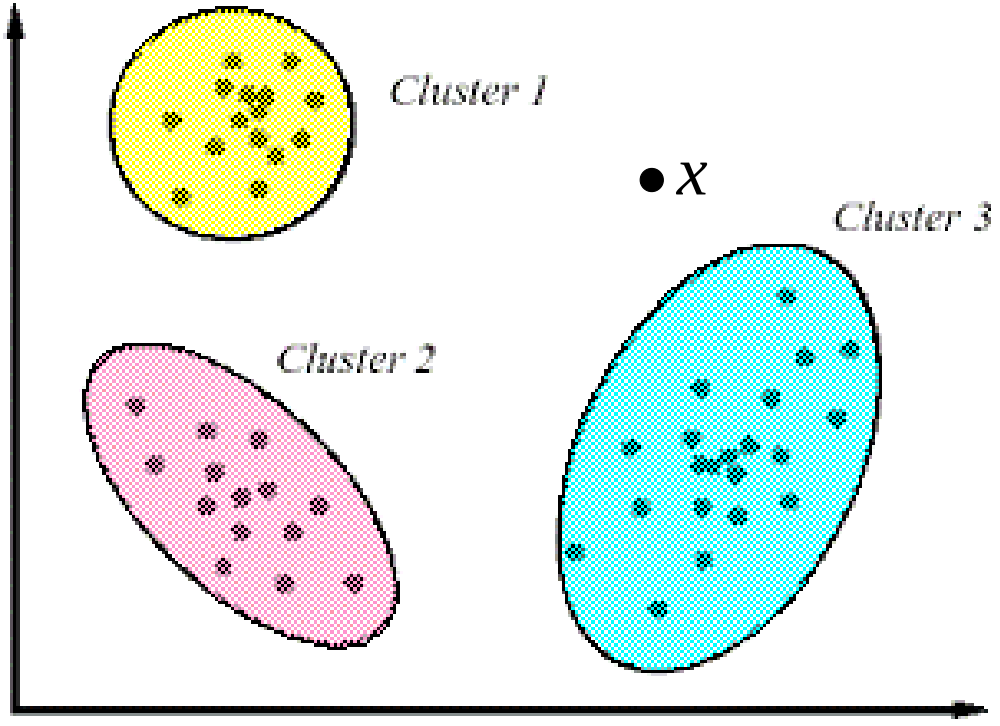


- How many clusters are there? We do not know.
- We have to apply some heuristics to choose *K* or experiment with different *K*s.
- The best K is the one that gives minimum of

$$D = \sum_{k=1}^{K} \sum_{\mathbf{x}_p \in C_k} d(\mathbf{x}_p, \mathbf{c}_k)$$

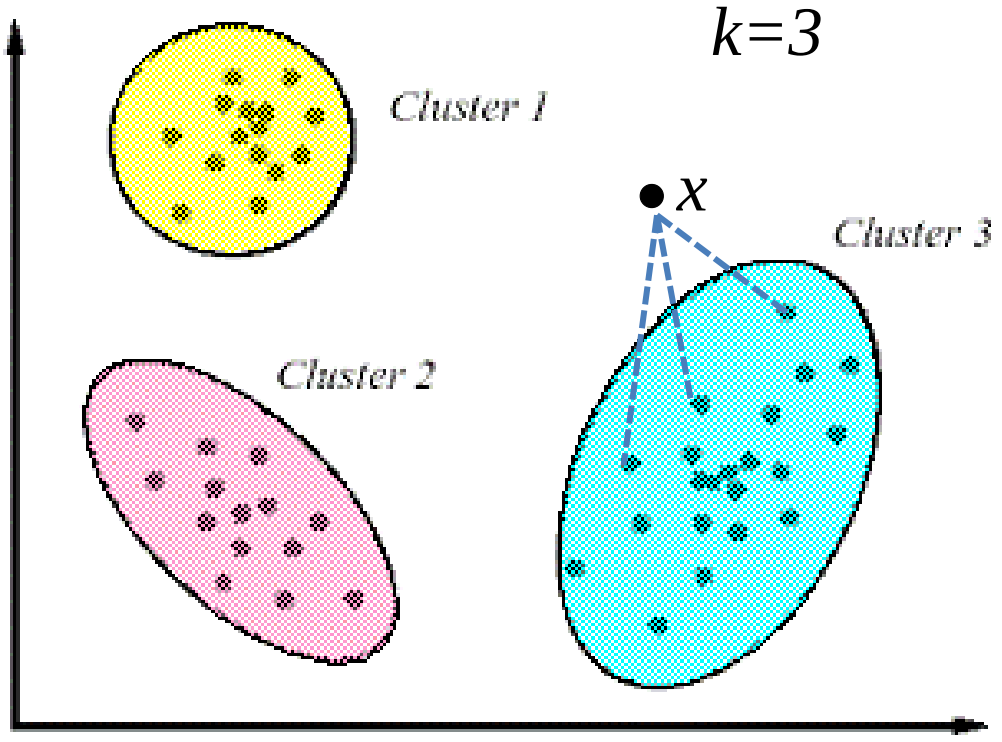
New data point

- Let us suppose that we have successfully clustered our data into 3 clusters.
- What will happen if a new observation (data point) *x* appears?
- How do we know which cluster (class) it belongs to?



k- nearest neighbours

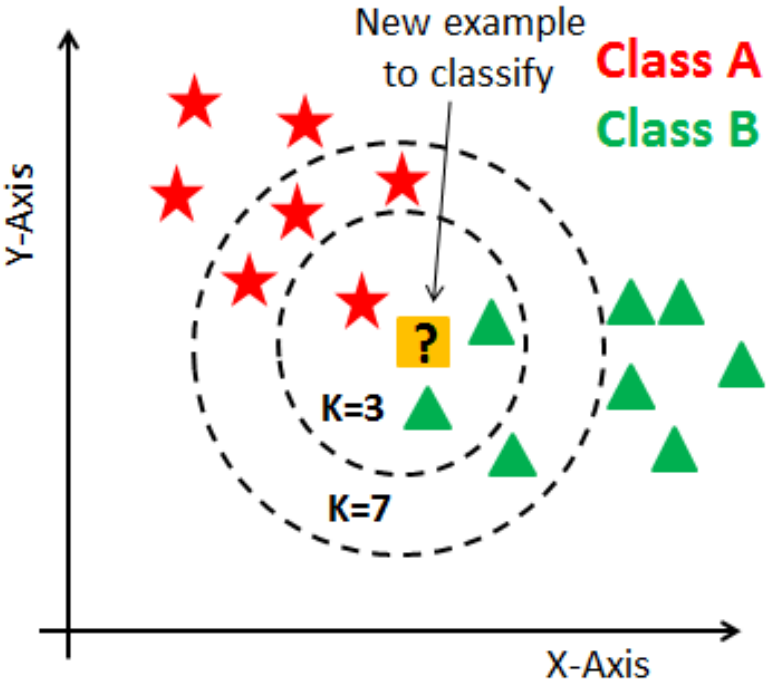
- The way to go about classifying a new sample (i.e. assigning it a class label) is called the method of *k*-nearest neighbours.
- It is a very simple method: **using a given metric**, find *k* nearest (already classified) data points and take the majority vote of these neighbours. To avoid ties, *k* is always an odd number (1, 3, 5, etc).



kNN classification: the role of k

- kNN can be applied even without clustering, as long as we have some data with class labels, we can apply the same voting principle for the new data point with an unknown class label.

- The test sample will be classified either to the class of red stars or to the class of green triangles.
  - If $k = 3$, the sample is classified to the triangle class because there are 2 triangles and only 1 star inside the inner circle of 3 nearest neighbours.
  - If $k = 7$, the sample is classified to the star class (4 stars vs. 3 triangles inside the outer circle of 7 nearest neighbours)



Application of SOM for classification

- Let the training data are labelled with a class label.

- Let the cluster center is the neuron with such a weight vector that the total distance between all vectors belonging to the cluster and the center of the cluster is minimal

$$\operatorname{argmin} \sum_{\mathbf{w}_p \in C_k} d(\mathbf{w}_p, \mathbf{c}_k)$$

- Then we can clasiffy a new samples according to several criteria
  - According to the closest center
  - According to the *k* nearest neighbours

Conclusions

- Clustering and SOM are exploratory data mining techniques, and common techniques for statistical data analysis, used in many fields, including pattern recognition, image analysis, information retrieval, bioinformatics, computer graphics.

- Self-organized map
  - There is no "teacher" to tell the neurons how they should change their synaptic weights
  - The input itself provides the necessary information to adjust the synaptic weights between neurons

- There are number of SW packages for SOM, see e.g.
  - http://www.scholarpedia.org/article/Kohonen_network