BABEŞ-BOLYAI UNIVERSITY
Faculty of Computer Science and Mathematics

ARTIFICIAL INTELLIGENCE

Intelligent systems

Machine learning

Support Vector Machines

K-means

Intelligent Systems – Support Vector Machines

- Support Vector Machines (SVMs)
  - Definition
  - Solved problems
  - Advantages
  - Difficulties
  - Tools

Intelligent Systems – Support Vector Machines

- Definition
  - Developed by Vapnik in 1970
  - Popularised after 1992
  - Linear classifiers that identify the hyperplane that separates the positive and negative classes
  - Have a theoretical foundation
  - Work very well for large data (text mining, image analysis)

  - **Remember**
    - Supervised learning problem – a data set:
      - (x^d, t^d), with:
      - $x^d \in \mathbf{R}^m$ ☐ $x^d=(x^d_1, x^d_2,..., x^d_m)$
      - $t^d \in \mathbf{R}$ ☐ $t^d \in \{1, -1\}$, 1 ☐ positive class, -1 ☐ negative class
      - where d = 1,2,...,n,n+1,n+2,...,N
    - First n data (x^d and t^d are known) are used as training data
    - Last N-n data (x^d is known, t^d is unknown) are used as testing data

Intelligent Systems – Support Vector Machines

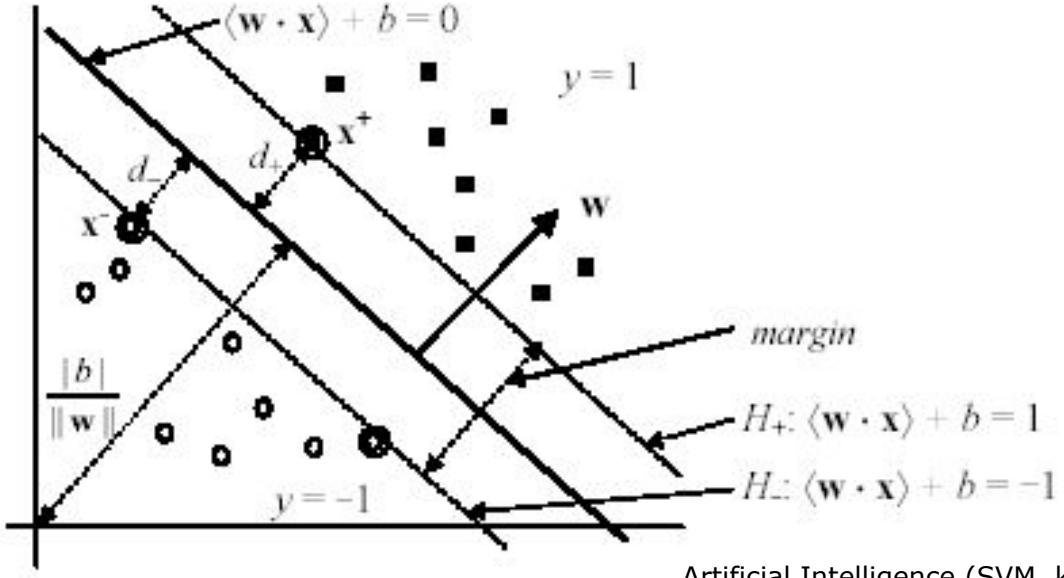
- Definition
  - SVM finds a linear function $f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b$, ($\mathbf{w}$ –weight vector) such as

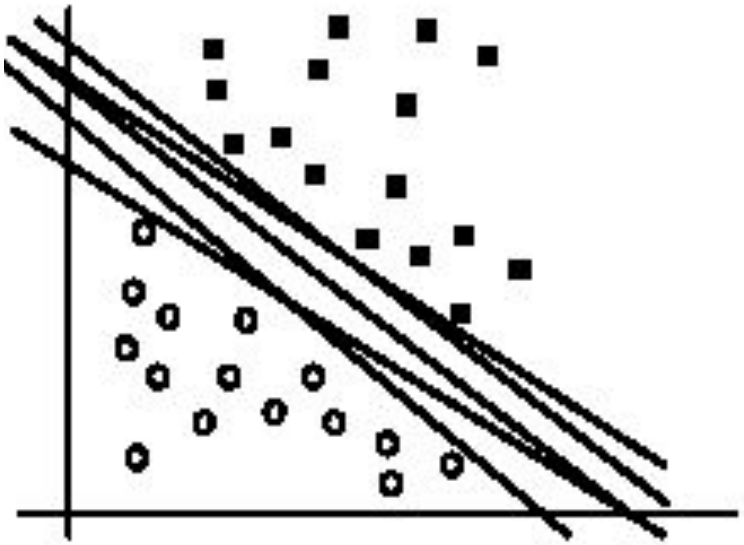
$$y_i = \begin{cases} 1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b \geq 0 \\ -1 & if \langle \mathbf{w} \cdot \mathbf{x}_i \rangle + b < 0 \end{cases}$$

  - $\langle \mathbf{w} \cdot \mathbf{x} \rangle + b = 0$ ☐ decision hyperplane that separates the two classes

Intelligent Systems – Support Vector Machines

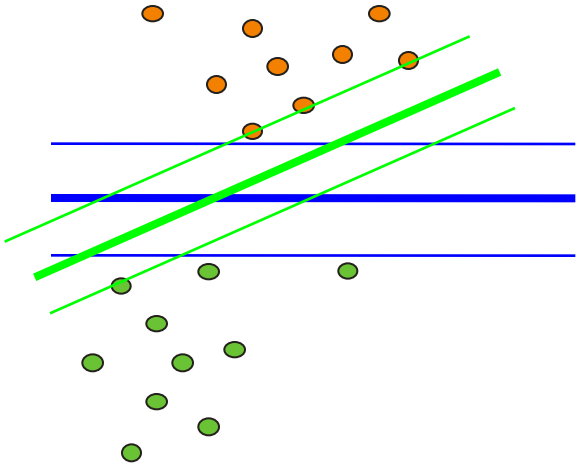
- Definition
  - There are more hyper-planes
    - Which is the best hyper-plane?
  - SVM searches the hyper-plane with the largest margin (that minimises the generalisation error)
    - SMO (*Sequential minimal optimization*) algorithm





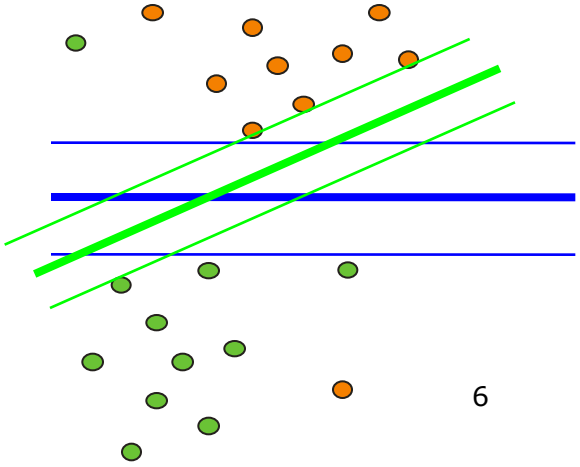
Intelligent Systems – Support Vector Machines

- Solved problems
  - Classification problems ☐ more cases (based on the data type):
    - Linear separable



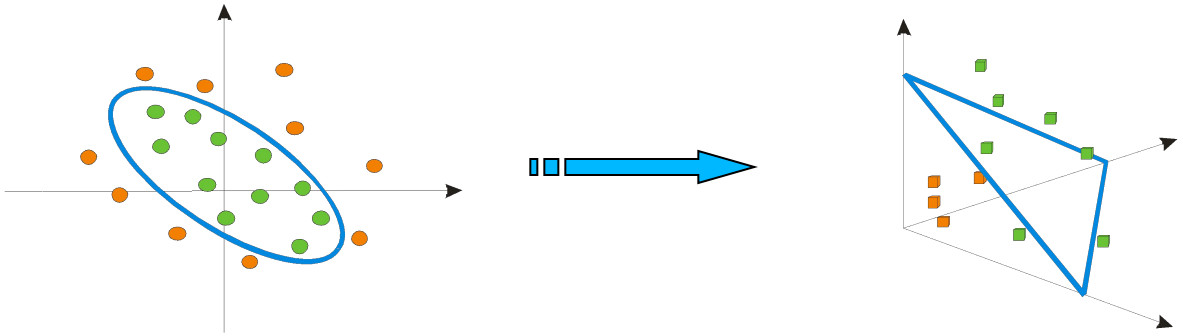
      - Separable
        - Error = 0

      - Non-separable
        - Constraints are relaxed ☐ some error are allowed
        - C – penalisation coefficient



Intelligent Systems – Support Vector Machines

- Solved problems ☐ classification problems ☐ data cases:
  - Non-linear separable
    - Input space is transformed (mapped) into a space of more dimensions (feature space) by using kernel function – in this new space the data becomes linear separable
    - In SVMs the kernel function computes the distance among 2 points
      - ☐ kernel ~ similarity function



Intelligent Systems – Support Vector Machines

- Solved problems ☐ classification problems ☐ data cases:
  - Non-linear separable ☐ possible kernels
    - Classic kernels
      - Polynomial kernel: $K(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) = (\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2} + 1)^p$
      - RBF kernel: $K(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) = exp(- ||\boldsymbol{x}^{d1} - \boldsymbol{x}^{d2}||^2/2\sigma^2)$
    - Multiple Kernels
      - Linear : $K(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) = \sum w_i K_i (\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2})$
      - Non-linear
        - Without coefficients: $K(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) = K_1(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) + K_2(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) * exp(K_3(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}))$
        - With coefficients: $K(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) = K_1 (\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) + c_1 * K_2 *(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) exp(c_2 + K_3(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}))$
    - Kernels for strings
    - Kernels for images
    - Kernels for graphs

Intelligent Systems – Support Vector Machines

SVM parameters setting:

- Penalisation coefficient C
  - C – small slowly convergence
  - C – large fast convergence

- Kernel parameters
  - If *m* (# of attributes) is larger than *n* (# of data)
    - SVM with a linear kernel (SVM without kernel)

$$K(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) = \boldsymbol{x}^{d1} \cdot \boldsymbol{x}^{d2}$$

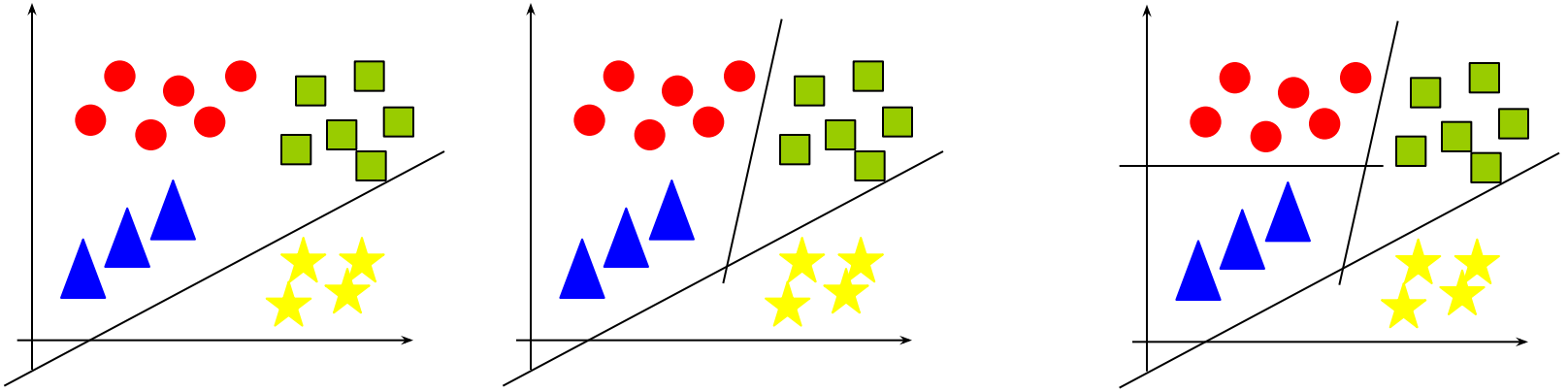
  - If *m* (# of attributes) is large and *n* (# of data) is medium
    - SVM with Gaussian kernel

$$K(\boldsymbol{x}^{d1}, \boldsymbol{x}^{d2}) = exp(-||\boldsymbol{x}^{d1} - \boldsymbol{x}^{d2}||^2 / 2\sigma^2)$$

    - σ – dispersion of training data
    - Attributes must be normalised (scaled to (0,1))
  - If *m* (# of attributes) is small and *n* (# of data) is large
    - Add new attributes and use SVM with linear kernel

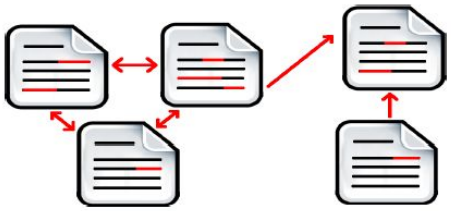
Intelligent Systems – Support Vector Machines

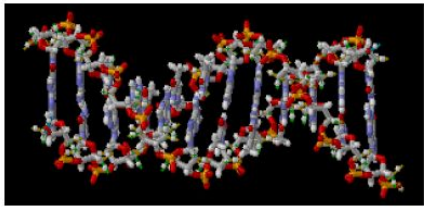
- SVM for multi-class classification problems (more than 2 classes)
  - one *vs.* all

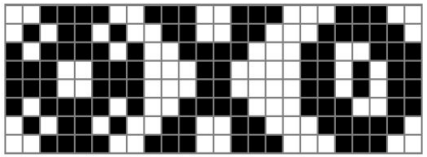


Intelligent Systems – Support Vector Machines

- Structured SVMs
  - Machine Learning
    - Simple SVM f: $\mathcal{X} \rightarrow \mathbb{R}$
      - Any type of inputs
      - Numerical outputs (natural numbers, integers, real numbers)
    - Structured SVM: $\mathcal{X} \rightarrow \mathcal{Y}$
      - Any type of inputs
      - Any type of outputs (numerical or structured outputs)
  - Structured information
    - Texts and hyper-texts
    - Molecules and molecular structures
    - Images







Intelligent Systems – Support Vector Machines

- Structured SVMs
  - Applications
    - Natural Language Processing
      - Automatic translation (outputs ☐ sentences)
      - Syntactic and/or morphologic analysis of sentences (outputs ☐ syntactic and/or morphologic tree)
    - Bioinformatic
      - Prediction of secondary structures (outputs ☐ bi=partite graphs)
      - Prediction of enzyme function (outputs ☐ paths in trees)
    - Speech processing
      - Automatic transcriptions (outputs ☐ sentences)
      - Transformation of texts in voice (outputs ☐ audio signal)
    - Robotics
      - Planning (outputs ☐ sequences of actions)

Intelligent Systems – Support Vector Machines

- Advantages
  - Can work with any type of data (linear or non-linear separable, uniform distributed or not, with known or unknown distribution)
    - Kernel function that creates new attributes (features) ☐ hidden layers of an ANN
  - If the problem is convex SVM finds a unique solution ☐ global optima
    - ANNs can associates more solutions ☐ local optima
  - Automatic selection of the learnt model (by support vectors)
    - In ANNs hidden layers have to be configured apriori
  - Avoid overfitting
    - ANNs have overfitting problems even the cross-validation is involved

- Difficulties
  - Real attributes only
  - Binary classification problems only
  - Difficult mathematical background

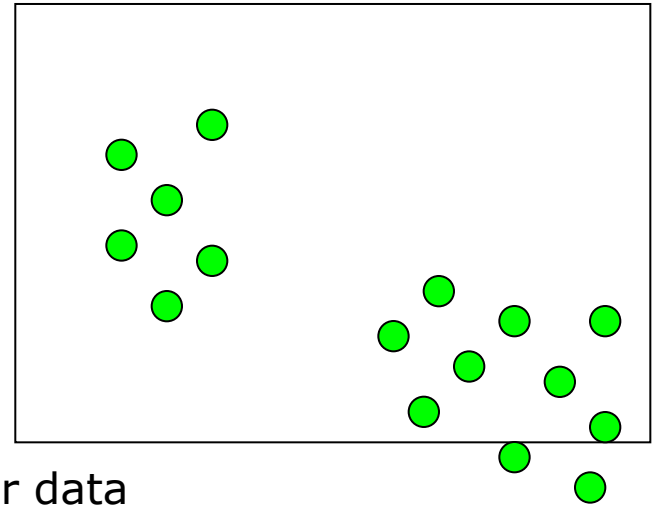
- Tools
  - LibSVM ☐ http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - Weka ☐ SMO
  - SVMLight ☐ http://svmlight.joachims.org/
  - SVMTorch ☐ http://www.torch.ch/
  - http://www.support-vector-machines.org/

Intelligent systems – Machine Learning

- Typology
  - Experience criteria:
    - Supervised learning
    - Unsupervised learning
    - Active learning
    - Reinforcement learning
  - Algorithm criteria
    - Decision trees
    - Artificial Neural Networks
    - Evolutionary Algorithms
    - Support Vector Machines
    - Hidden Markov Models
    - **K-means**

Unsupervised learning

- Aim
  - to find a model or a structure of data



- Solved problems
  - Identification of groups (clusters)
    - Analysis of genes
    - Image processing
    - Analysis of social networks
    - Market segmentation
    - Analysis of astronomical data
    - Clusters of computers
  - Dimension reduction
  - Identification of causes (explanations) for data
  - Modelling the data densities

- Specific
  - **Data are not annotated** (labelled)

Unsupervised learning – definition

Separates the un-labelled examples in disjoint subsets (clusters) such as:

- Examples of the same cluster are similar
- Examples of different clusters are different

Definition

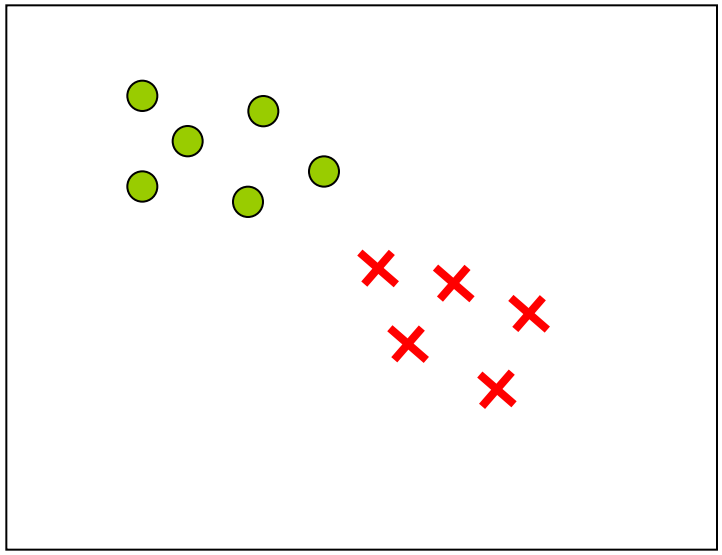
- Given
  - A set of data (examples, instances, cases)
    - Training data
      - As ***atribute_data***$_i$, where
        - $i = 1,N$ (N = # of training data
        - ***atribute_data***$_i$= (atr_{i1}, atr_{i2}, ..., atr_{im}), m – # of attributes (characteristics, properties) of data
    - Testing data
      - As (***atribute_data***$_i$), $i = 1,n$ (n = # of testing data)
- Determine
  - An unknown function that groups the training data in more classes
    - # of classes can be pre-defined (k) or unknown
    - Data of the same class are similar
  - The class of a new testing data by using the learnt grouping (on training data)

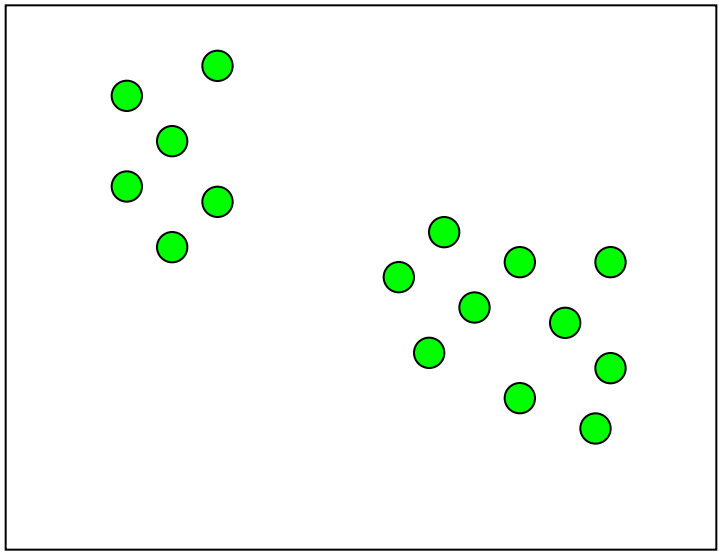
Other names

- Clustering

Unsupervised learning – definition

- Supervised *vs.* unsupervised





Unsupervised learning – definition

- Distance between 2 elements ***p*** and ***q*** ϵR^m
  - Euclid distance
    - $d(\mathbf{p},\mathbf{q})=sqrt(\Sigma_{j=1,2,...,m}(p_j-q_j)^2)$
  - Manhattan distance
    - $d(\mathbf{p},\mathbf{q})=\Sigma_{j=1,2,...,m}|p_j-q_j|$
  - Mahalanobis distance
    - $d(\mathbf{p},\mathbf{q})=sqrt(\mathbf{p}-\mathbf{q})S^{-1}(\mathbf{p}-\mathbf{q})),$
      - Where S is the covariance matrix ($S= E[(\mathbf{p}-E[\mathbf{p}])(\mathbf{q}-E[\mathbf{q}])]$)
  - Internal product
    - $d(\mathbf{p},\mathbf{q})=\Sigma_{j=1,2,...,m}p_jq_j$
  - Cosine distance
    - $d(\mathbf{p},\mathbf{q})=\Sigma_{j=1,2,...,m}p_jq_j / (sqrt(\Sigma_{j=1,2,...,m}p_j^2) * sqrt(\Sigma_{j=1,2,...,m}q_j^2))$
  - Hamming distance
    - # of differences between ***p*** and ***q***
  - Levenshtein distance
    - Minimal # of operations required for transforming ***p*** in ***q***

- Distance *vs.* similarity
  - Distance ☐ minimisation
  - Similarity ☐ maximisation

Unsupervised learning – definition

Application

- Gene clustering
- Market segmentation (for client clustering)
- news.google.com

Unsupervised learning – process

Process

- 2 steps:
  - Learning
    - Determine (learn), by using an algorithm, the existing clusters
  - Testing
    - Include a new data in one of the identified (during training) clusters

Learning quality (clustering validation)

- Internal criteria
  - Large similarity inside the cluster and reduce similarity between clusters
- External criteria
  - Using benchmarks composed of *apriori* grouped data

Unsupervised learning – evaluation

Performance measures:

- Internal criteria
  - Distance inside the cluster
  - Distance between clusters
  - Davies-Bouldin index
  - Dunn index
- External criteria
  - Comparison with known data – impossible in real-world applications
  - Precision
  - Recall
  - F-measure

Unsupervised learning – evaluation

Internal criteria

- Distance inside cluster c_j that contains n_j instances
  - Average distance (among instances)
    - $D_a(c_j) = \sum_{xi1,xi2 \epsilon cj} ||x_{i1} - x_{i2}|| / (n_j(n_j\text{-}1))$
  - Nearest neighbour distance
    - $D_{nn}(c_j) = \sum_{xi1 \epsilon cj} min_{xi2 \epsilon cj} ||x_{i1} - x_{i2}|| / n_j$
  - Distance to centroids
    - $D_c(c_j) = \sum_{xi, \epsilon cj} ||x_i - \mu_j|| / n_j$, where $\mu_j = 1/n_j \sum_{xi \epsilon cj} x_i$

Unsupervised learning – evaluation

Internal criteria

- Distance between two clusters c_{j1} and c_{j2}
  - Simple link
    - $d_s(c_{j1}, c_{j2}) = min_{xi1 \epsilon cj1,\ xi2 \epsilon cj2} \{||x_{i1} - x_{i2}||\}$
  - Complete link
    - $d_{co}(c_{j1}, c_{j2}) = max_{xi1 \epsilon cj1,\ xi2 \epsilon cj2} \{||x_{i1} - x_{i2}||\}$
  - Average link
    - $d_a(c_{j1}, c_{j2}) = \Sigma_{xi1 \epsilon cj1,\ xi2 \epsilon cj2} \{||x_{i1} - x_{i2}||\} / (n_{j1} * n_{j2})$
  - Link between centroids
    - $d_{ce}(c_{j1}, c_{j2}) = ||\mu_{j1} - \mu_{j2}||$

Unsupervised learning – evaluation

Internal criteria

- Davies-Bouldin index ☐ min ☐ compact clusters
  - $DB = 1/nc * \Sigma_{i=1,2,...,nc} max_{j=1, 2, ..., nc, j \neq i}((\sigma_i + \sigma_j)/d(\mu_i, \mu_j))$
  - where:
    - nc – # of clusters
    - μ_i – centroid of cluster i
    - σ_i – average of distances between elements form cluster i and the centroid μ_i
    - $d(\mu_i, \mu_j)$ – distance between centroid μ_i and centroid μ_j
- Dunn index
  - Identifies the dense clusters and well separated
  - $D=d_{min}/d_{max}$
  - where:
    - d_{min} – minimal distance between 2 elements from different clusters - intra-cluster distance
    - d_{max} – maximal distance between 2 elements from the same cluster - inter-cluster distance

Unsupervised learning – typology

- How the clusters are forming
  - Hierarchic clustering
  - Non-hierarchical (partitioned) clustering
  - Clustering based on data density
  - Clustering based on a grid

Unsupervised learning – typology

- How the clusters are forming
  - Hierarchic clustering
    - Creates a dendogram (taxonomic tree)
      - Creates the clusters (recursively)
      - k (# of clusters) is unknown
    - Agglomerative clustering (bottom-up) ☐ small clusters to large clusters
    - Divisive clustering (top-down) ☐ large clusters to small clusters
    - Eg.
      - Clustering ierarhic agglomerative

Unsupervised learning – typology

How the clusters are formed

- Non-hierarchical
  - Partitional ☐ determine a data separation ☐ all the clusters in the same time
  - Optimises an objective function defined
    - Locally – by using some features only
    - Globally – by using all attributes

    that can be:

    - squared error – sum of squared distances between data and the cluster's centroid ☐ min
      - Ex. *K-means*
    - Graph-based
      - Ex. Clustering based in minimum spanning tree
    - Based on probabilistic models
      - Ex. Identify the data distribution ☐ expectation maximisation
    - Based on the nearest neighbour

  - Required to fix k apriori ☐ fix the initial clusters
    - Algorithm is run more times with different parameters and the most efficient version is selected

  - Ex. *K-means, ACO*

Unsupervised learning – typology

How the clusters are forming

- Based on data densities
  - Data density and data connectivity
    - Cluster formation is based on data density from a given region
    - Cluster formation is based on data connectivity from a given region
  - Function of data density
    - Tries to model the data distribution
  - Advantage:
    - Modeling of clusters of any shape

Unsupervised learning – typology

- How the cluster are forming
  - Based on a grid
    - Is not a distinct approach
      - Can be hierarchic, partitional or density-based
    - Involves data space segmentation in regular areas
    - Objects are placed on a multi-dimensional grid
    - Eg. ACO

Unsupervised learning – typology

- How the algorithms work
  - Agglomerative clustering
    1. Initially, each instance form a cluster
    2. Compute the distance between any 2 clusters
    3. Reunion the closest 2 clusters
    4. Repeat steps 3 and 4 until a single cluster is obtained or other stop criterion is satisfied
  - Divisive clustering
    1. Establish the number of clusters (*k*)
    2. Initialise the centre of each cluster
    3. Determine a data separation
    4. Re-compute the centre of each cluster
    5. Repeat steps 3 and 4 until the partition is unchanged (algorithm converges)

- How the algorithm takes into account the attributes (features)
  - Monotonic – attributes are taken into account one-by-one
  - Polytonic – attributes are simultaneous taken into

Unsupervised learning – typology

- How the data belong to clusters
  - Exact clustering (hard clustering)
    - Each instance x_i has associated a label (class) c_j
  - Fuzzy clustering
    - Associates to each input $\boldsymbol{x_i}$ a degree (probability) of membership f_{ij} to a certain class c_j □ an instance $\boldsymbol{x_i}$ that can belong to several clusters

Unsupervised learning – algorithms

- Agglomerative hierarchical clustering
- K-means
- AMA
- Probabilistic models
- Nearest neighbour
- Fuzzy
- Artificial Neural Network
- Evolutionary algorithms
- ACO

Unsupervised learning – algorithms

Agglomerative hierarchical clustering

a. Consider a distance between 2 instances $d(x_{i1}, x_{i2})$

b. Form N clusters, each of them containing an instance

c. Repeat
   - Determine the closest 2 clusters
   - Reunion the 2 clusters ☐ a cluster

Until *a single cluster is obtained*

Unsupervised learning – algorithms

Agglomerative hierarchical clustering

Distance between 2 clusters c_i and c_j:

- Simple link ☐ minimal distance between the objects of 2 clusters
  - $d(c_i, c_j) = min_{xi1 \epsilon ci,\ xi2 \epsilon cj}\ sim(\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2})$
- Complete link ☐ maximal distance between the objects of 2 clusters
  - $d(c_i, c_j) = max_{xi1 \epsilon ci,\ xi2 \epsilon cj}\ sim(\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2})$
- Average link ☐ mean of distances between the objects of 2 clusters
  - $d(c_i, c_j) = 1 / (n_i * n_j) \sum_{xi1 \epsilon ci} \sum_{xi2 \epsilon cj} d(\boldsymbol{x}_{i1}, \boldsymbol{x}_{i2})$
- Average link over group ☐ distance between the means (centroids) of 2 clusters
  - $d(c_i, c_j) = \rho(\boldsymbol{\mu}_i, \boldsymbol{\mu}_j)$, ρ – distance, $\boldsymbol{\mu}_j = 1/n_j \sum_{xi \epsilon cj} \boldsymbol{x}_i$

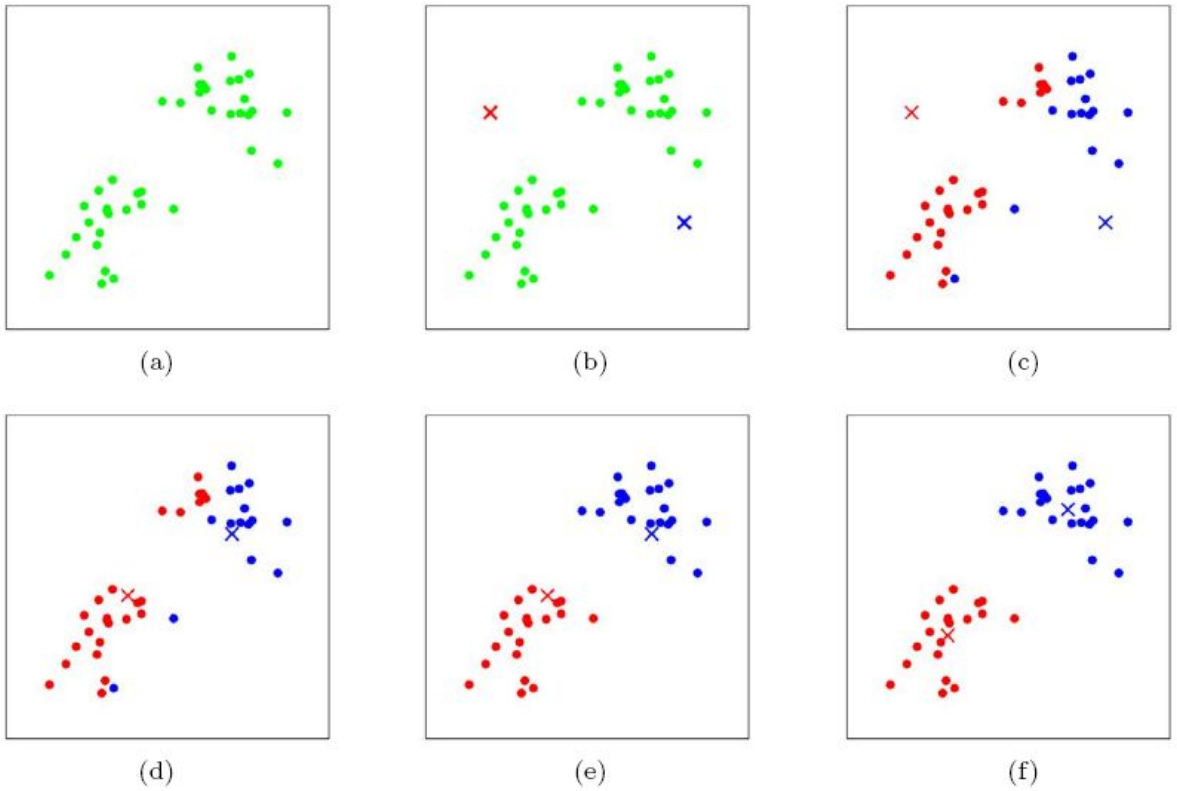
Unsupervised learning – algorithms

K-means (Lloyd algorithm / Voronoi iteration)

- Suppose that k clusters will form
- Initialise *k* centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$
  - A centroid $\boldsymbol{\mu}_j$ ($i=1,2, \ldots, k$) is a vector of m values (m - # of features)
- Repeat until convergence
  - Associated to each instance the nearest centroid □ for each instance $\boldsymbol{x}_i$, $i = 1, 2, \ldots, N$
    - $c_i = arg\ min_{j = 1, 2, \ldots, k} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2$
  - Re-compute the centroids by moving them in the mean of instances associated to it □ for each cluster c_j, $j = 1, 2, \ldots, k$
    - $\boldsymbol{\mu}_j = \Sigma_{i=1,2, \ldots N} 1_{ci=j}\, \boldsymbol{x}_i \,/\, \Sigma_{i=1,2, \ldots N}\, 1_{ci=j}$

Unsupervised learning – algorithms

- K-means



Unsupervised learning – algorithms

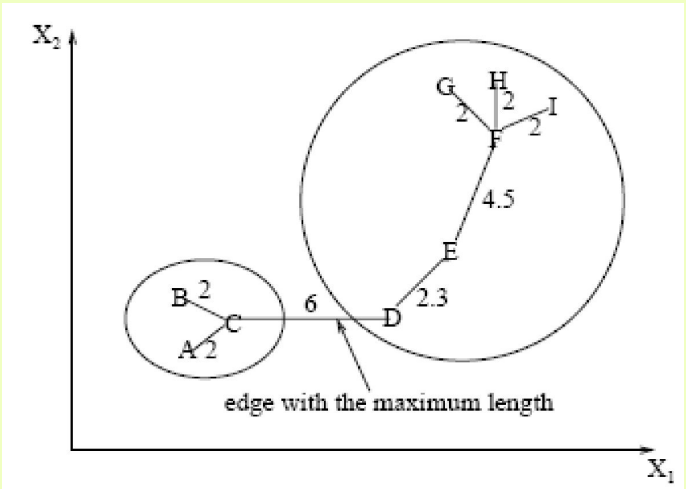
K-means

- Initialisation of k centroids $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$
  - With random values (in the definition domain of the problem)
  - With k instances of N (randomly selected)
- Does the algorithm always converge?
  - Yes, because of distortion function J
    - $J(c, \mu) = \sum_{i=1,2,\ldots,N} ||\boldsymbol{x}_i - \boldsymbol{\mu}_{cj}||^2$

    which is decreasing
  - Converges in a local optima
  - Finding the global optima ☐ NP-difficult problem

Unsupervised learning – algorithms
Clustering based on minimum spanning tree (AMA)

- Construct the minimum spanning tree of data
- Eliminate from the tree the longest edges and form clusters



Unsupervised learning - algorithms
Probabilistic models

- http://www.gatsby.ucl.ac.uk/~zoubin/course04/ul.pdf
- http://learning.eng.cam.ac.uk/zoubin/nipstut.pdf

Unsupervised learning - algorithms
Nearest neighbor

- Some of the instances are labeled

- It is repeated until all instances are labeled
  - An unlabeled instance will be included in the closest instance cluster
    - if the distance between the unlabeled instance and the labeled one is less than a threshold

Unsupervised learning - algorithms
Fuzzy clustering

- An initial fuzzy partitioning is established
  - The membership degrees matrix U, is constructed, where u_{ij} – the degree of membership of instance $\boldsymbol{x}_i$ ($i=1,2, \ldots, N$) to the cluster c_j ($j = 1, 2, \ldots, k$) ($u_{ij} \in [0,1]$)
    - The higher u_{ij} is, the higher the confidence that instance $\boldsymbol{x}_i$ is part of cluster c_j
- An objective function is established
  - $E^2(U) = \sum_{i=1,2, \ldots, N} \sum_{j=1,2,\ldots,k} u_{ij} ||\boldsymbol{x}_i - \boldsymbol{\mu}_j||^2$,
    - where $\boldsymbol{\mu}_j = \sum_{i=1,2, \ldots, N} u_{ij} \boldsymbol{x}_i$ – the center of the j^{th} fuzzy cluster
    - which is optimized (min) by re-assigning instances (in new clusters)
- Fuzzy Clustering ☐ Hard Clustering
  - imposing a threshold on the membership function u_{ij}

Thank you for your attention!