

Unsupervised Learning

- Usually learning can be done in a supervised or unsupervised way
- There are a strong bias in the machine learning community towards supervised learning
- But a lot of concepts are learned unsupervisedly
- The discovery of new concepts always is unsupervised

Unsupervised Learning

- Goals:
 - Summarization: To obtain representations that describe an unlabeled dataset
 - Understanding: To discover the concepts inside the data
- These task are difficult because the discovery process is biased by context
 - Different answers can be valid depending of the discovery goal or the domain
 - There are few criterion to validate the results
- Representation of the clusters: Unstructured (partitions) or relational (hierarchies)

Numerical Taxonomy

- Data analysis by the search of predefined structures, fitting of known probability distributions/models
- It is assumed that the data is embedded in a N-dimensional space that has a distance function defined (similarity/dissimilarity)
- Examples are more related to the nearest examples than to the farthest

Numerical Taxonomy - Algorithms

Two main strategies:

- Hierarchical algorithms
 - Examples are organized as a binary tree
 - No explicit division in groups
- Partitional algorithms
 - Only a partition of the dataset is obtained

Hierarchical algorithms

- Based on graph theory
 - The examples form a full connected graph
 - Similarity define the length of the edges
 - The clustering is decided using connectivity criteria
- Based on matrix algebra
 - A distance matrix is calculated from the examples
 - The clustering is computed using the distance matrix
 - The distance matrix is updated after each step (different updating criteria)

Hierarchical algorithms

- Graphs
 - Single Linkage, Complete Linkage, MST
 - Divisive, Agglomerative
- Matrices
 - Johnson algorithm
 - Different update criteria (S-L, C-L, Centroid, minimum variance)

Computational cost

$$O(n_{inst}^3 \times num_dimensions)$$

Agglomerative Graph Algorithm

Algorithm: Agglomerative graph algorithm)

Compute Distance/similarity matrix

repeat

 Find the pair of examples with smallest similarity

 Add an edge to the graph corresponding to this pair

if *Agglomeration criteria holds* **then**

 Merge the clusters the pair belong

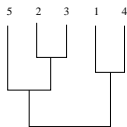
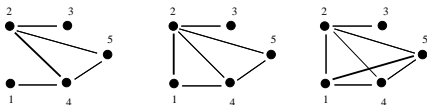
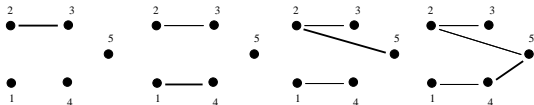
end

until *Only one Cluster*

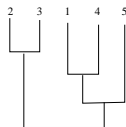
- Single linkage = The new edge is between two disconnected graphs
- Complete linkage = The new edge creates a clique with all the nodes of both subgraphs

Hierarchical algorithms - Graphs

	2	3	4	5
1	6	8	2	7
2		1	5	3
3			10	9
4				4



Single Link



Complete Link

Agglomerative Johnson algorithm

Algorithm: Agglomerative Johnson algorithm

Compute Distance/similarity matrix

repeat

Find the pair of groups/examples with the smallest similarity

Merge the pair of groups/examples

Delete the rows and columns corresponding to the pair of groups/examples

Add a new row and column with the new distances to the new group

until *Matrix has one element*

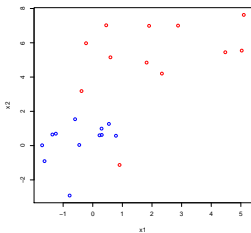
- Single linkage = New distance is the distance between the nearest examples
- Complete linkage = New distance is the distance between the farthest examples
- Average linkage = New distance is the distance between centroids

Hierarchical algorithms - Matrices

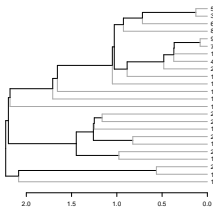
	2	3	4	5
1	6	8	2	7
2		1	5	3
3			10	9
4				4
	2,3	4	5	
1	7	2	7	
2,3		7.5	6	
4			4	
	1,4	5		
2,3	7.25	6		
1,4		5.5		
	1,4,5			
2,3	6.725			

Hierarchical algorithms - Example

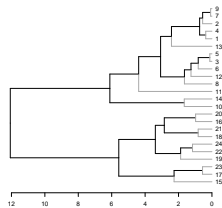
Data



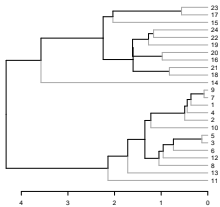
Single Link



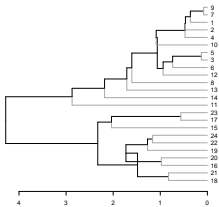
Complete Link



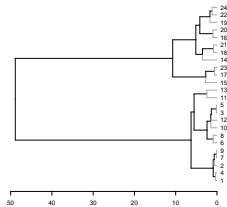
Median



Centroid



Ward



Hierarchical algorithms - Problems

- A partition of the data is not given, it has to be decided a posteriori
- Some undesirable and strange behaviours could appear (chaining, inversions)
- Dendrogram is not a practical representation for large amount of data
- Its computational cost is high

Partitional algorithms

The computational cost to find the optimal partition of N objects in K groups is NP-hard

- Square error minimization (k-means)
- Mixture decomposition (Expectation-Maximization)
- Density estimation
- Graph theory: MST, Relative neighbourhood graph, Gabriel graph
- Neighbourhood relationships
- Fuzzy clustering
- Spectral Clustering
- Unsupervised Neural networks

K-means

- We assume that the shape of the clusters is hyperspherical
- An iterative algorithm assigns each example to one of K groups (K is a parameter)
- Hill Climbing search
- Optimization criteria (square error, minimize the distance of each example to the centroid of the class)

$$Distorsion = \sum_{k=1}^K \sum_{i \in C_k} \|x_i - \mu_k\|^2$$

- The algorithm converges to a local minima

K-means

Algorithm: K-means (X : Examples, k :integer)

Generate k prototypes with the k first examples

Assign the $n-k$ examples to its nearest prototype

$SumD =$ Sum of square distances examples-prototypes

repeat

 Recalculate prototypes

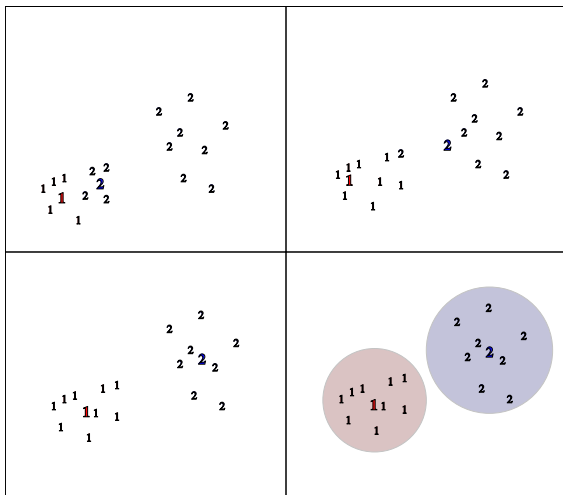
 Reassign examples to its nearest prototype

$SumI = SumD$

$SumD =$ Sum of square distances examples-prototypes

until $SumI - SumD < \epsilon$

K-means



K-means - practical problems

- The algorithm is sensitive to the initialization (to run the algorithm from random initializations could be a good idea)
- Find the value of k is not an easy problem (experimentation with different values is needed)
- You can obtain a solution even if the classes are not hyperspherical (some classes could be splitted)
- No guarantee about the quality of the solution
- The space and computational cost makes it not suitable for big datasets

Mixture Decomposition - EM algorithm

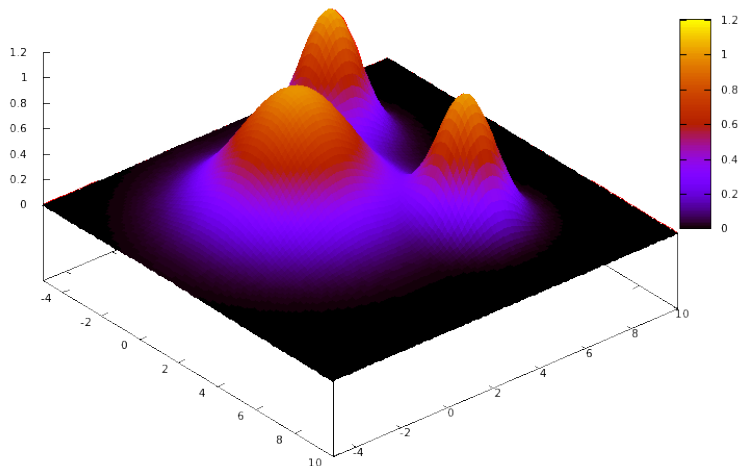
- We assume that the data are drawn from a mixture of probability distribution functions (usually Gaussian), we are looking for the parameters of the distributions that explain better the data
- The model of the data is:

$$P(x|\theta) = \sum_{i=1}^K P(w_i)P(x|\theta_i, w_i)$$

Being K the number of clusters and $\sum_{i=1}^K P(w_i) = 1$

- Each instance has a probability to belong to a class

Mixture Decomposition - EM algorithm



Mixture Decomposition - EM algorithm

- The goal is to estimate the parameters of the distribution that describes each class (e.g.: means and standard deviations)
- The algorithm maximizes the likelihood of the distribution respect the dataset
- It performs iteratively two steps
 - **Expectation:** We calculate a function that assigns a degree of membership to all the instances to any of the K probability distributions
 - **Maximization:** We re-estimate the parameters of the distributions to maximize the membership

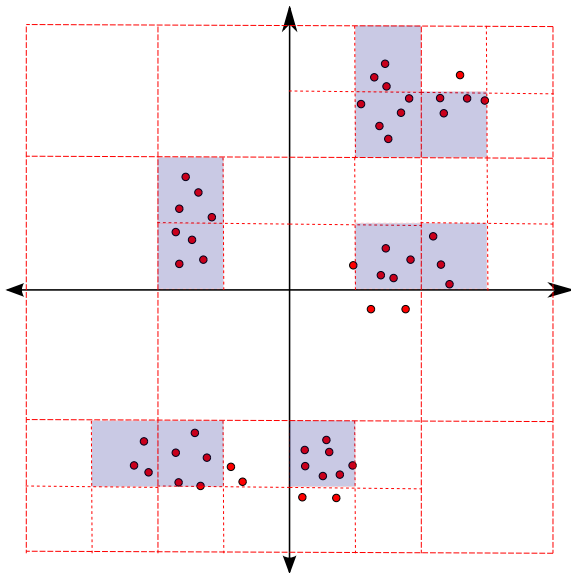
EM algorithm - Comments

- K-means is a particular case of this algorithm
- The main advantage is that we obtain a membership as a probability (soft assignments)
- Using different probability distribution we can find different kinds of structures.

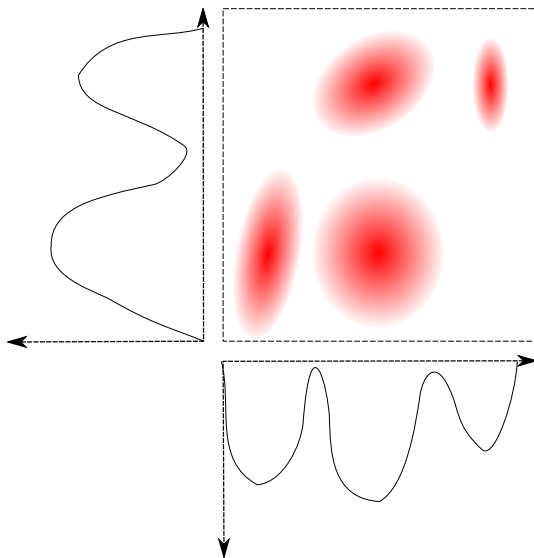
Density estimation

- The number of groups is not decided beforehand
- We are looking for regions with high density of examples
- We are not limited to predefined shapes (there is no model)
- Different approaches:
 - Space partitioning (exponential number of partitions)
 - Multidimensional histograms (we look for high density regions with less dimensions)
- Usually it is only applied to datasets with low dimensionality

Density estimation

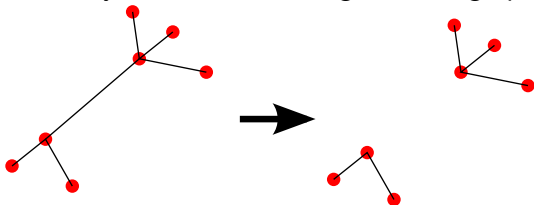


Density estimation



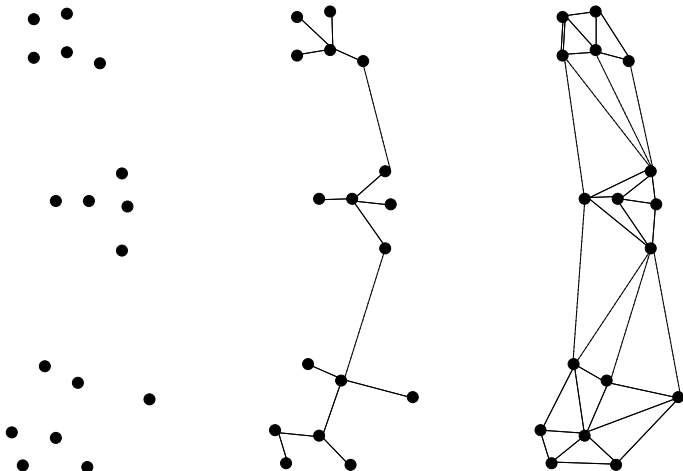
Based in graph theory

- We create different kinds of graphs with the dataset (MST, Voronoi, Delanau, ...)
- We give consistency criteria for the edges of the graph (delete)



- The result is a set of unconnected components
- Two advantages: we do not need to know the number of classes, we do not look for a specific model (any shape is possible)

Based in graph theory



Incremental algorithms: Neighbourhood relationship

- The commonality among all the algorithms until this point is that they are not incremental
- Incrementality allows to update a model with new data without starting from scratch
- These algorithms use the neighbourhood relationship defined from a similarity function
- This neighbourhood determines what instances belong to the same group
- Examples: Nearest Neighbour, Mutual Neighbour

Nearest Neighbour

Algorithm: Nearest-Neighbour (X: Examples, D:double)

Generate a prototype with the first example

while *there are examples* **do**

 e= current example

 d= distance of e to the the nearest prototype

if $d \leq D$ **then**

 Introduce the example in the class

 Recompute the prototype

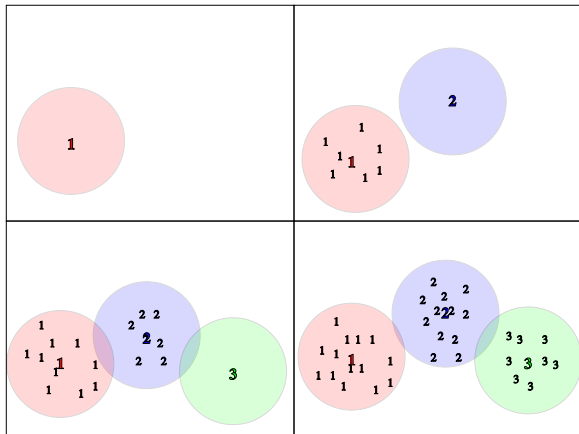
else

 Create a new prototype with this example

end

end

Nearest Neighbour



Disadvantages of Numerical Taxonomy

From the artificial intelligence perspective

- Are based only on syntactic similarity, no context information is used
- Mainly focused on numerical data
- There are no explicit representation of the groups
- Analysis and search are done only by the data analyst
- No possibility of biasing the search using domain information
- Interpretability is not considered

Cognitive Psychology

- Its research includes the models of human conceptualization
 - How the concepts that we use are created
 - How concepts are characterized/described/modelled
 - How are organized
- It gives theories about characteristics and properties of the concepts
- It is the basis of some of the algorithms for unsupervised symbolic machine learning

Cognitive Psychology (II)

Three theories about conceptualization

- Classic view: Disjoint Concepts
- Probabilistic/exemplar view: Similarity
- Theory based view: Context

Classic view

A concept is defined from a list of necessary and sufficient properties

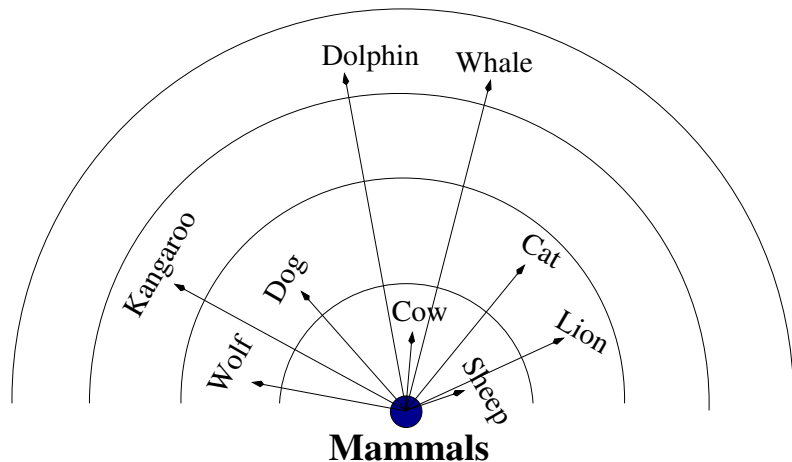
Problems:

- Is difficult to find this list of necessary and sufficient properties for all concepts
- Grade of membership to a concept of the examples of a concept (Typicality)
- Its difficult to assign some examples to a concept (Ambiguity)
- Preferential level for conceptualization (Basic level)

Probabilistic/exemplar based view

- Probabilistic
 - A concept is a list of attributes, but they are neither sufficient nor necessary
 - Concepts are organized respect their *family resemblance*
 - There is a prototype that represents the central tendency of the concept
 - Membership to a concept is defined from the number of attributes that are shared with the prototype
- Exemplar based
 - A concept is defined by a list of representative examples
 - Membership to a concept is defined from the number of similar examples
 - The number of similar examples is a measure of the typicality of an example
 - How the representative examples are chosen?

Probabilistic/exemplar based view



Probabilistic/exemplar based view - Comparison

- Probabilistic view is more space efficient (just one prototype)
- Probabilistic view has a description of the concepts
- Exemplar based view is biased because of the existence of outliers
- Exemplar based view allows partial comparisons
- Exemplar based view is used by expert individuals, probabilistic view is the categorization used by non experienced individuals
- Exemplar based theory is more accepted as the actual human conceptualization

Probabilistic/exemplar based view - Assumptions

- Similarity among examples are an increasing function of the attributes shared and decreasing of the attributes not shared
- All attributes are independent
- All attributes are on the same level of abstractness
- A concept is equivalent to a list of properties

Theory based view

- Concepts are organised around theories that explain the categorization
- Similarity it is not an absolute, it depends on the context (the perception of an attribute can change)
- Attributes are interrelated (some are redundant)
- Attributes can belong to different levels of abstractness
- Concepts are more complex than a list of attributes

Probabilistic/exemplar based view

Games

Poker

Chess

Checkers

Basketball

Marbles

Shoot-them-up

Similarity?

*What defines a game?
Relevant characteristics?
Explanation?*

Unsupervised learning

- We try to extend the view from numerical taxonomy
 - Using qualitative data
 - Allowing the algorithm to do the search in the space of parameters
 - Using background knowledge to reduce the search space
 - Obtaining characterizations of the results and looking for interpretability
- Including the ideas from cognitive psychology about conceptualization

Unsupervised learning

- Conceptual clustering
 - Non incremental algorithms
 - The result is a partition or a hierarchy
 - Based on numerical taxonomy algorithms
- Concept formation
 - Incremental algorithms
 - The result is a hierarchy
 - Inspired in the ideas from cognitive psychology

Conceptual Clustering

- Input:
 - A dataset
 - A set of attributes that characterize the dataset
 - Background knowledge (constraints about the concepts, relationships among attributes, relevance, ...)
- Result:
 - A partition or a hierarchy that describes the dataset and characterizes the groups

These algorithms combine two tasks

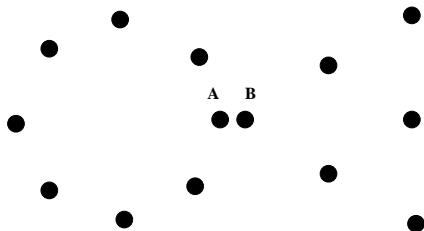
Clustering: We determine the membership of examples to groups that are considered a concept

Characterization: We obtain a description of the groups

Conceptual Clustering

- Similarity is important
- But we also have to take in account the context
- We can only consider a group if there is a matching concept

$$\textit{Similarity} = f(\textit{Obj}_i, \textit{Obj}_j, \textit{context}, \textit{available concepts})$$



Conceptual Clustering - CLUSTER/2

[Michalski, Stepp, 1983]

- Characterization using selectors (sets of conjunctive formulas)
- Algorithm similar to K-means
- Optimization of a quality function
- Search is guided by a set of weighted criteria
- Weights are chosen by the user
- Weights are selected depending on the kind of concepts that we are looking for
- Same data could result in different sets of concepts using different weights

Conceptual Clustering - CLUSTER/2

Lexicographic Evaluation Function (LEF)

- The fit between clusters and data: Descriptions cover the examples
- Simplicity of cluster descriptions: Number of descriptors needed
- Inter-cluster difference: Concepts well separated
- Discrimination index: Descriptions different enough
- Dimensionality reduction: Minimum number of descriptors

CLUSTER/2

Algorithm: CLUSTER/2 (X : Examples, k :integer)

Generate k descriptions with the first k examples

Obtain disjoint descriptions as conjunctive formulas

Assign each $n-k$ examples to their nearest prototype

Save the partition

repeat

 Select the more central examples from each group

 Compute LEF

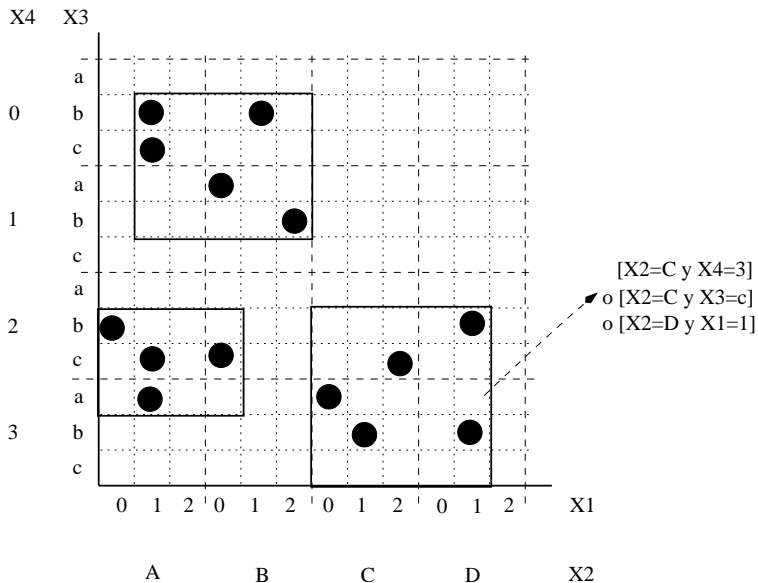
if *this partition is better* **then**

 substitute last partition with this

end

until *LEF is not improved*

CLUSTER/2 - Example



Concept Formation

- Learning has an incremental nature (experience is acquired from continuous observation, not at once)
- Concepts are learnt with their relationships (polithetic hierarchies of concepts)
- Search in the space of hierarchies
- An objective function measures the utility of the learnt structure
- The updating of the structure is performed by a set of conceptual operators
- The result depends on the order of the examples

Concept Formation - COBWEB

[Fisher, 1989]

- Based on the ideas from cognitive psychology
- Incrementally builds a hierarchy of concepts (top-down)
- The concepts are described probabilistically
- Defines four search operators
- Uses a measure to find the *basic level* (Category utility)

COBWEB - Category utility

- Measure defined from a set of categories
- Maximized by the categories in the basic level
- These classes maximize the predictivity of their attributes
- This measures bias the search to categories with high intra-similarity and low inter-similarity

COBWEB - Category utility

- Intra class similarity : $P(A_i = V_{ij}|C_k)$
- Maximize \rightarrow most of the examples in the class share this value for this attribute
- Inter class similarity: $P(C_k|A_i = V_{ij})$
- Maximize \rightarrow fewer examples from other classes share this value for this attribute
- Maximize the tradeoff between the two measures for a given set of categories:

$$\sum_{k=1}^K P(A_i = V_{ij}) \sum_{i=1}^I \sum_{j=1}^J P(A_i = V_{ij}|C_k)P(C_k|A_i = V_{ij})$$

COBWEB - Category utility

- Using Bayes theorem:

$$\sum_{k=1}^K P(C_k) \sum_{i=1}^I \sum_{j=1}^J P(A_i = V_{ij} | C_k)^2$$

- $\sum_{i=1}^I \sum_{j=1}^J P(A_i = V_{ij} | C_k)^2$ represents the number of attributes that can be correctly predicted for a class
- We look for a partition that increases this number of attributes compared to a baseline: $\sum_{i=1}^I \sum_{j=1}^J P(A_i = V_{ij})^2$

COBWEB - Category utility

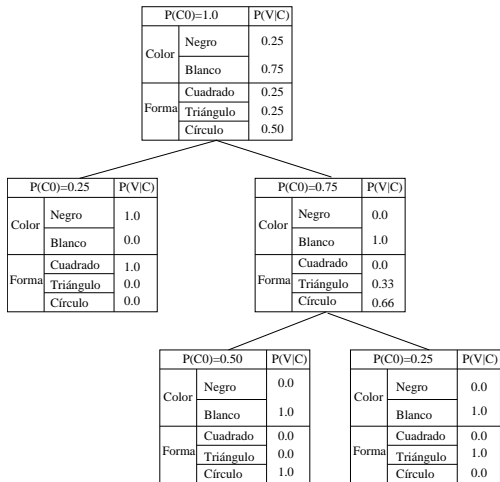
- Category utility for qualitative attributes for a set of k categories $\{C_1, \dots, C_k\}$

$$\frac{\sum_{k=1}^K P(C_k) \sum_{i=1}^I \sum_{j=1}^J P(A_i = V_{ij} | C_k)^2 - \sum_{i=1}^I \sum_{j=1}^J P(A_i = V_{ij})^2}{K}$$

- Category utility for quantitative attributes (Gaussian distributions)

$$\frac{\sum_{k=1}^K P(C_k) \sum_{i=1}^I \frac{1}{\sigma_{ik}} - \sum_{i=1}^I \frac{1}{\sigma_{ip}}}{K}$$

Probabilistic hierarchy



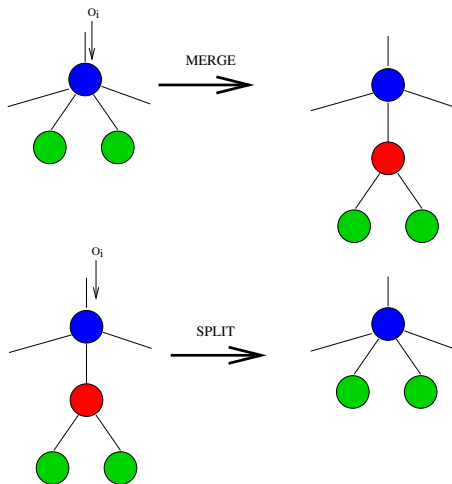
Algorithm

- Incremental insertion of each example in the hierarchy
- Look for the path from the root that puts the example in a leaf
- Decide at each level how to modify the hierarchy (which operator apply) to maximize CU and descend recursively the tree

Operators

- **Incorporate:** Put the example inside an existing class
- **New class:** Create a new class at this level
- **Merge:** Two concepts are merge and the example is incorporated inside the new class
- **Divide:** A concept is substituted by its children

Split - Merge



COBWEB

Procedure: Depth-first limited search COBWEB (x: Example, H: Hierarchy)

Update the father with the new example

if *we are in a leaf* **then**

| Create a new level with this example

else

| Compute **CU** of incorporating the example to each class

| Save the two best **CU**

| Compute **CU** of merging the best two classes

| Compute **CU** of splitting the best class

| Compute **CU** of creating a new class with the example

| Recursive call with the best choice

end
