Unsupervised Learning

- Unsupervised learning method: no target value (class label) to be predicted, the goal is finding common patterns or grouping similar examples.
- Differences between models/algorithms:
  - Conceptual (model-based) vs. partitioning
  - Exclusive vs. overlapping
  - Deterministic vs. probabilistic
  - Hierarchical vs. flat
  - Incremental vs. batch learning
- Evaluating clustering quality: subjective approaches, objective functions (e.g. category utility, entropy).
- Major approaches:
  - Cluster/2: flat, conceptual (model-based), batch learning, possibly overlapping, deterministic.
  - Partitioning methods: flat, batch learning, exclusive, deterministic or probabilistic. Algorithms: k-means, probability-based clustering (EM)
  - Hierarchical clustering
    - Partitioning: agglomerative (bottom-up) or divisible (top-down).
    - Conceptual: Cobweb, category utility function.

CLUSTER/2

1. Select k objects (seeds) from the set of observed objects (randomly or using some selection function).
2. For each seed, using it as a positive example the all the other seeds as negative examples, find a maximally general description that covers all positive and none of the negative examples.
3. Classify all objects form the sample in categories according to these descriptions. Then replace each maximally general descriptions with a maximally specific one, that cover all objects in the category. (This possibly avoids category overlapping.)
4. If there are still overlapping categories, then using some metric (e.g. euclidean distance) find central objects in each category and repeat steps 1-3 using these objects as seeds.
5. Stop when some quality criterion for the category descriptions is satisfied. Such a criterion might be the complexity of the descriptions (e.g. the number of conjuncts)
6. If there is no improvement of the categories after several steps, then choose new seeds using another criterion (e.g. the objects near the edge of the category).

Partitioning methods – k-means

- Iterative distance-based clustering.
- Used by statisticians for decades.
- Similarly to Cluster/2 uses k seeds (predefined k), but is based on a distance measure:
  1. Select k instances (cluster centers) from the sample (usually at random).
  2. Assign instances to clusters according to their distance to the cluster centers.
  3. Find new cluster centers and go to step 2 until the process converges (i.e. the same instances are assigned to each cluster in two consecutive passes).
- The clustering depends greatly on the initial choice of cluster centers – the algorithm may fall in a local minimum.
- Example of bad chioce of cluster centers: four instances at the vertices of a rectangle, two initial cluster centers – midpoints of the long sides of the rectangle. This is a stable configuration, however not a good clustering.
- Solution to the local minimum problem: restart the algorithm with another set of cluster centers.
- Hierarchical k-means: apply $k = 2$ recursively to the resulting clusters.

Probabilty-based clustering

Why probabilities?

- Restricted amount of evidence implies probabilistic reasoning.
- From a probabilistic perspective, we want to find the most likely clusters given the data.
- An instance only has certain probability of belonging to a particular cluster.

Probabilty-based clustering – mixture models

- For a single attribute: three parameters - mean, standard deviation and sampling probability.
- Each cluster A is defined by a mean (μ_A) and a standard deviation (σ_A).
- Samples are taken from each cluster A with a specified probability of sampling $P(A)$.
- Finite mixture problem: given a dataset, find the mean, standard deviation and the probability of sampling for each cluster.
- If we know the classification of each instance, then:
  - mean = average, $\mu = \frac{1}{n}\sum_1^n x_i$;
  - standard deviation, $\sigma^2 = \frac{1}{n-1}\sum_1^n (x_i - m)^2$;
  - probability of sampling for class A, $P(A)$ = proportion of instances in it.
- If we know the three parameters, the probability that an instance x belongs to cluster A is:

  $P(A|x) = \frac{P(x|A)P(A)}{P(x)}$,

  where $P(x|A)$ is the density function for A, $f(x; \mu_A, \sigma_A) = \frac{1}{\sigma_A\sqrt{2\pi}}\, e^{\frac{-(x-\mu_A)^2}{2\sigma_A^2}}$.

  $P(x)$ is not necessary as we calculate the numerators for all clusters and normalize them by dividing by their sum.

  $\rightarrow$ In fact, this is exactly the Naive Bayes approach.
- For more attributes: naive Bayes assumption – independence between attributes. The joint probabilities of an instance are calculated as a product of the probabilities of all attributes.

EM (expectation maximization)

- Similarly to k-means, first select the cluster parameters (μ_A, σ_A and $P(A)$) or guess the classes of the instances, then iterate.
- Adjustment needed: we know cluster probabilities, not actual clusters for each instance. So, we use these probabilities as weights.
- For cluster A:

  $\mu_A = \frac{\sum_1^n w_i x_i}{\sum_1^n w_i}$, where w_i is the probability that x_i belongs to cluster A;

  $\sigma_A^2 = \frac{\sum_1^n w_i (x_i - \mu)^2}{\sum_1^n w_i}$.
- When to stop iteration - maximizing overall likelihood that the data come form the dataset with the given parameters ("goodness" of clustering):

  $Log - likelihood = \sum_i log(\ \sum_A P(A) P(x_i|A)\ )$

  Stop when the difference between two successive iteration becomes negligible (i.e. there is no improvement of clustering quality).

Criterion functions for clustering

- Distance-based functions.

  Sum of squared error:

$$m_A = \frac{1}{n} \sum_{x \in A} x$$

$$J = \sum_A \sum_{x \in A} ||x - m_A||^2$$

  Optimal clustering minilizes J: *minimal variance* clustering.

- Probability (entropy) based functions.

  Probability of instance $P(x_i) = \sum_A P(A)P(x_i|A)$

  Probability of sample $x_1, ..., x_n$:

$$\Pi_i^n(\sum_A P(A)P(x_i|A))$$

  Log-likelihood:

$$\sum_i^n \log(\sum_A P(A)P(x_i|A))$$

- Category utility (Cobweb):

$$CU = \frac{1}{n} \sum_C P(C) \sum_A \sum_v [P(A = v|C)^2 - P(A = v)^2]$$

- Error-based evaluation: evaluate clusters with respect to classes using preclassified examples.

Hierarchical partitioning methods

- Bottom-up (agglomerative): at each step merge the two closest clusters.
- Top-down (divisible): split the current set into two clusters and proceed recursively with the subsets.
- Distance function between instances (e.g. Euclidean distance).
- Distance function between clusters (e.g. distance between centers, minimal distance, average distance).
- Criteria for stopping merging or splitting:
  - desired number of clusters;
  - distance between the closest clusters is above (top-down) or below (bottom-up) a threshold.
- Algorithms:
  - *Nearest neighbor (single-linkage)* agglomerative clustering: cluster distance = minimal distance between elements. Merging stops when distance > threshold. In fact, this is an algorithm for generating a *minimal spanning tree.*
  - *Farthest neighbor (complete-linkage)* agglomerative clustering: cluster distance = maximal distance between elements. Merging stops when distance > threshold. The algorithm computes the *complete* subgraph for every cluster.
- Problems: greedy algorithm (local minimum), once created a subtree cannot be restructured.

Hierarchical conceptual clustering: Cobweb

- Incremental clustering algorithm, which builds a taxonomy of clusters without having a predefined number of clusters.
- The clusters are represented probabilistically by conditional probability $P(A = v|C)$ with which attribute A has value v, given that the instance belongs to class C.
- The algorithm starts with an empty root node.
- Instances are added one by one.
- For each instance the following options (operators) are considered:
  - classifying the instance into an existing class;
  - creating a new class and placing the instance into it;
  - combining two classes into a single class (merging) and placing the new instance in the resulting hierarchy;
  - dividing a class into two classes (splitting) and placing the new instance in the resulting hierarchy.
- The algorithm searches the space of possible hierarchies by applying the above operators and an evaluation function based on the category utility.

Measuring quality of clustering – Category utility (CU) function

- CU attempts to maximize both the probability that two instances in the same category have attribute values in common and the probability that instances from different categories have different attribute values.

$$CU = \sum_{C} \sum_{A} \sum_{v} P(A = v) P(A = v|C) P(C|A = v)$$

- $P(A = v|C)$ is the probability that an instance has value v for its attribute A, given that it belongs to category C. The higher this probability, the more likely two instances in a category share the same attribute values.
- $P(C|A = v)$ is the probability that an instance belongs to category C, given that it has value v for its attribute A. The greater this probability, the less likely instances from different categories will have attribute values in common.
- $P(A = v)$ is a weight, assuring that frequently occurring attribute values will have stronger influence on the evaluation.

Category utility

- After applying Bayes rule we get

$$CU = \sum_C \sum_A \sum_v P(C)P(A = v|C)^2$$

- $\sum_A \sum_v P(A = v|C)^2$ is the expected number of attribute values that one can correctly guess for an arbitrary member of class C. This expectation assumes a probability matching strategy, in which one guesses an attribute value with a probability equal to its probability of occurring.
- Without knowing the cluster structure the above term is $\sum_A \sum_v P(A = v)^2$.
- The final CU is defined as the increase in the expected number of attribute values that can be correctly guessed, given a set of n categories, over the expected number of correct guesses without such knowledge. That is:

$$CU = \frac{1}{n} \sum_C P(C) \sum_A \sum_v [P(A = v|C)^2 - P(A = v)^2]$$

- The above expression is divided by n to allow comparing different size clusterings.
- Handling numeric attributes (Classit): assuming normal distribution and using probability density function (based on mean and standard deviation).

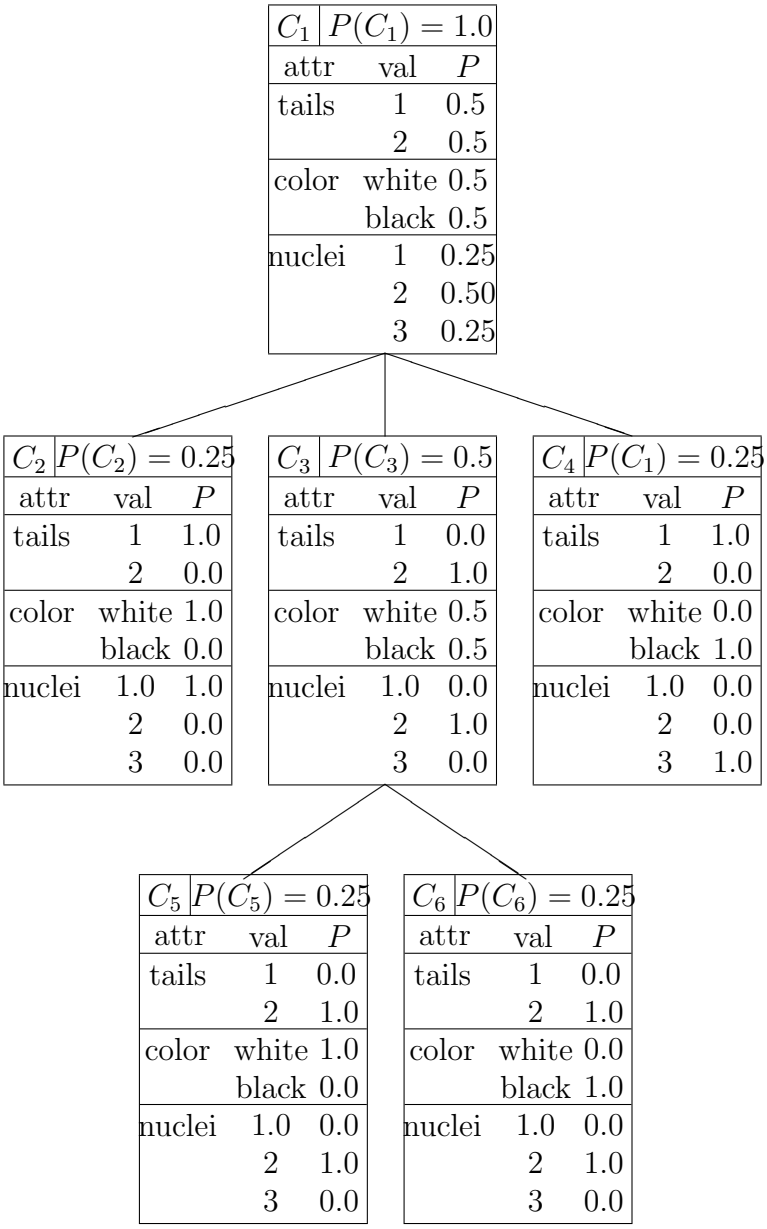
Control parameters

- Acuity: a single instance in a cluster results in a zero variance, which in turn produces infinite value for CU. The acuity parameter is the minimum value for the variance (can be interpreted as a measurement error).
- Cutoff: the minimum increase of CU to add a new node to the hierarchy, otherwise the new node is cut off.

Example

color	nuclei	tails
white	1	1
white	2	2
black	2	2
black	3	1

Cobweb's clustering hierarchy



Cobweb algorithm

cobweb($Node$,$Instance$)
begin

- If $Node$ is a leaf then begin
  Create two children of Node - L_1 and L_2;
  Set the probabilities of L_1 to those of $Node$;
  Set the probabilities of L_2 to those of $Insnatce$;
  Add $Instance$ to $Node$, updating $Node$'s probabilities.
  end

- else begin
  Add $Instance$ to $Node$, updating $Node$'s probabilities; For each child C of $Node$, compute the category utility of clustering achieved by placing $Instance$ in C;
  Calculate:
  S_1 = the score for the best categorization ($Instance$ is placed in C_1);
  S_2 = the score for the second best categorization ($Instance$ is placed in C_2);
  S_3 = the score for placing $Instance$ in a new category;
  S_4 = the score for merging C_1 and C_2 into one category;
  S_5 = the score for splitting C_1 (replacing it with its child categories.
  end

- If S_1 is the best score then call cobweb(C_1, $Instance$).

- If S_3 is the best score then set the new category's probabilities to those of $Instance$.

- If S_4 is the best score then call cobweb(C_m, $Instance$), where C_m is the result of merging C_1 and C_2.

- If S_5 is the best score then split C_1 and call cobweb($Node$, $Instance$).

end