Data Mining and Clustering

**Data Mining for Frequent Itemsets**

Frequent patterns are patterns that appear frequently in a dataset. If a substructure occurs frequently, it is called a structured pattern. These patterns can be represented in the form of association rules.

Rule support and confidence are two measures of rule interest. Association rules are considered if they satisfy both a minimum support threshold and a minimum confidence threshold.

support(A => B) = P(A union B)

confidence(A => B) = P(B|A) = support(A U B) / support(A)

Rules that support both minimum support and confidence are known as strong rules.

A set of items is called an itemset. The occurrence frequency of an itemset is the number of transactions that contain the itemset. This is known also by frequency, count or support count.

In general, association rule mining is done by

1. Finding all frequent itemsets
2. Generate strong association rules from the frequent itemsets

Because step 1 dominates execution time, the overall performance of the algorithm is governed by that step.

A major challenge in frequent itemset mining is the generation of a large number of itemsets, as frequent itemsets have subsets which are frequent as well. For this, we introduced closed frequent itemsets and maximal frequent itemsets.

An itemset X is closed in a dataset D if there exists no proper super-itemset Y such that Y has the same support count as X in D. An itemset X is closed frequent in D if X is both closed and frequent in D.

An itemset X is maximal frequent if X is frequent and there exists no super itemset Y such that X is a subset of Y and Y is frequent.

**Apriori Algorithm**

The algorithm uses prior knowledge of frequent itemset properties to perform a level-wise search, where k-itemsets are used to explore k+1-itemsets.

The Apriori property is : All nonempty subsets of a frequent itemset must also be frequent. This belongs to the anti monotonicity property, that if a set cannot pass a test, all its supersets will fail the same test.

The Apriori process is split into 2 main algorithms

1. Join
   1. To find Lk, a set of candidate k itemsets is generated by joining Lk-1 with itself.
2. Prune
   1. Any (k-1) itemset that is not frequent cannot be a subset of a frequent itemset.
   2. If any (k-1) itemset is not in Lk-1, then the candidate is removed, and this is called subset testing.

To generate association rules from the frequent itemsets

1. For each frequent itemset, generate all non empty subsets of the itemset.
2. For every non empty subset s, output the rule s => l -s if the confidence of s is greater than the minimum confidence threshold.

**Classification and Clustering**

Data classification mainly is a 2 step process

1. Learning step
   1. Build classification model from training set of data
2. Classification step

**Decision Tree Induction**

Decision trees are popular because

1. Construction of this tree requires no domain knowledge
2. They can handle multidimensional data
3. Intuitive and human readable information
4. Good accuracy depending on data

Decision trees adopt a greedy approach in a top-down recursive divide-and-conquer fashion.

Partitioning of a tree stops only when any one of the following is true

1. All tuples in partition D belong to the same class
2. There are no remaining attributes on which the tuples may be further partitioned, in which majority voting is applied
3. There are no tuples for a given branch

The resulting complexity is O(n x |D| x log|D|) where n is the number of attributes and D is the number of training tuples.

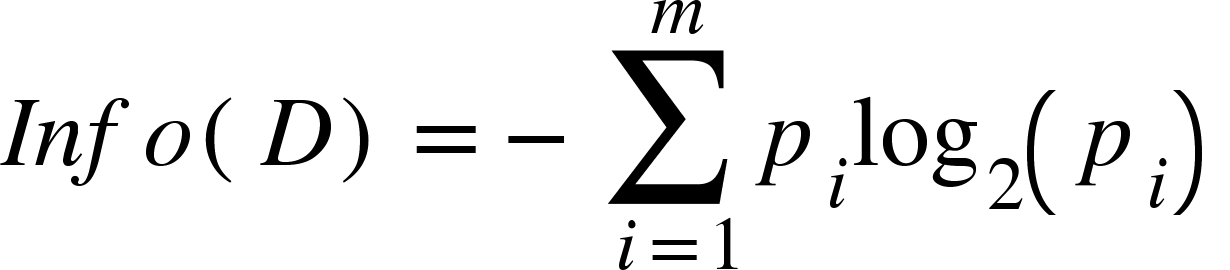
An attribute selection measure is a heuristic for selecting the splitting criterion that best separates a given data partition D.

Attribute selection measures are also known as splitting rules, because they determine how the tuples at a given node are to be split.

**Information Gain**

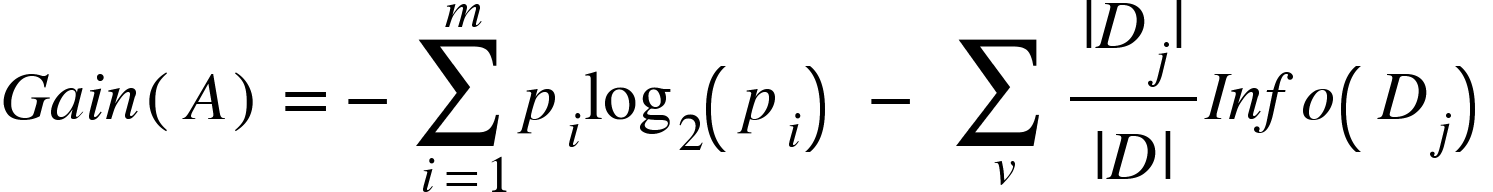
ID3 uses IG as an attribute selection measure.

The expected information needed to classify a tuple in D is given by



Where pi is the nonzero probability that a tuple in D belongs to class Ci estimated by Ci/D.

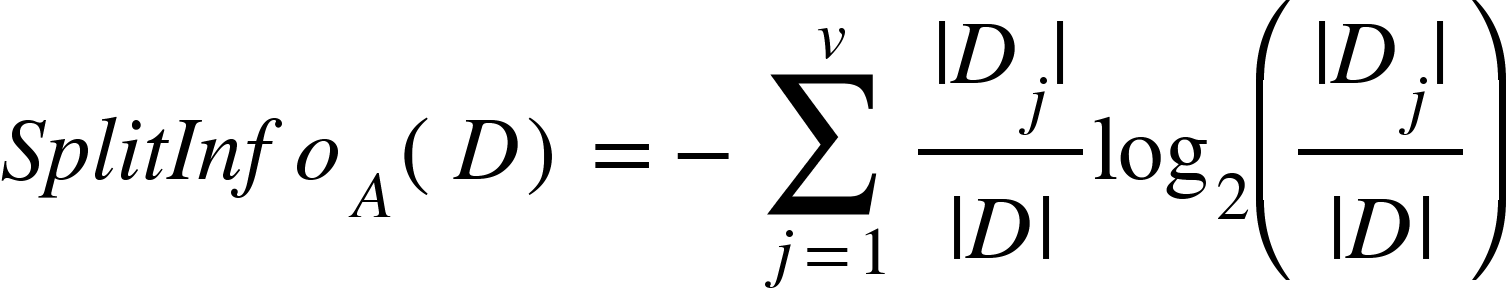
Information gain is given by



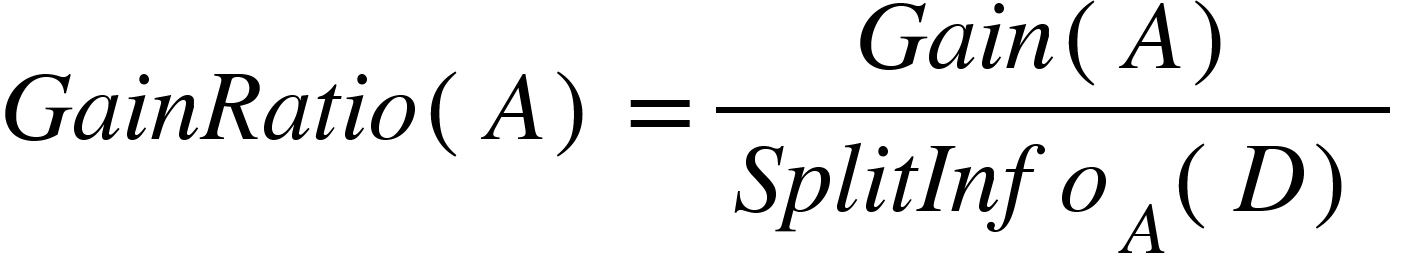
**Gain Ratio**

Information Gain is usually biased, selecting attributes with a large number of values.

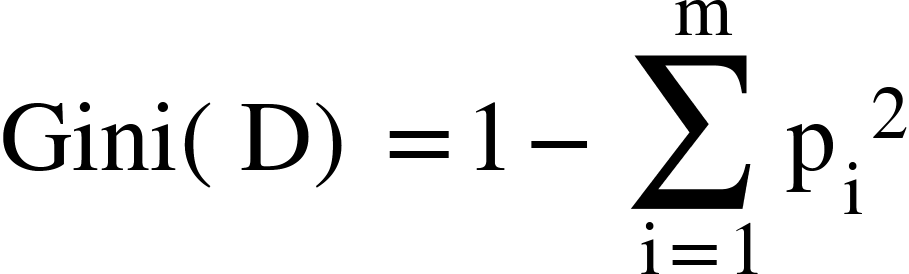
Here, split info is used analogously with entropy as



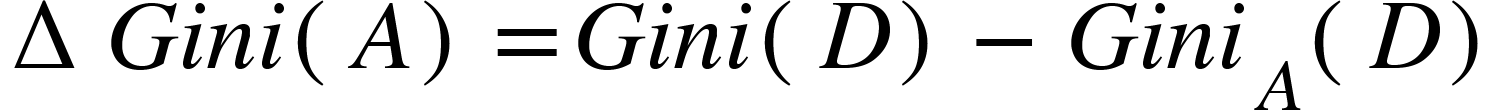
And the gain ratio is given as



**Gini Index**

****

The reduction in impurity is given by



The attribute that maximises this difference is taken.

**Other Selection Measures**

1. Minimum Description Length
   1. Least bias towards multivalued attributes
   2. Select tree that requires fewest bits to
      1. Encode the tree
      2. Encode the exceptions to the tree
2. Multivariate splits
   1. Take combinations of attributes using attribute construction methods

To represent anomalies in training due to noise and outliers, pruning is done.

Pre-pruning is done when construction of a tree is halted early, making the current node a leaf, using statistical significance measures.

Post-pruning is done when we have to remove subtrees from a fully built tree to increase accuracy.

The cost complexity of a pruning algorithm is an example of post pruning, where the cost complexity of a tree is described to be a function of the number of leaves and the error rate of the tree. If the cost complexity of a pruned tree is lesser than the actual tree, the tree is pruned.

A pruning set of class labels is used to estimate cost complexity. This is independent of the training set used to build the unpruned tree and of any test set used for accuracy estimation. It generates a set of progressively pruned trees, and the tree with the lowest cost complexity is taken.

Pessimistic pruning uses error rate estimates to make decisions on subtree pruning. It uses the training set to make these estimates, and adjust the rates to counter any biases.

The MDL principle can be used by seeing which tree requires the lowest number of bits to be needed for encoding.

Pre-pruning and post-pruning can also be done in an interleaved manner, to generate reliable trees.

Decision trees can suffer from

1. Repetition - attribute tested repeatedly along a given branch
2. Replication - Duplicate subtrees exist within a tree

The use of multivariate splits and different knowledge representation rules can fix this.

Scalability is a main issue for decision trees, and so an AVC set is maintained (Attribute-Value, Class), for each attribute, at each tree node, describing the training tuples at that node, allowing the tree to fit in the memory.

BOAT (Bootstrapped Optimistic Algorithm for Tree construction) uses bootstrapping to create smaller samples of the given training data and each subset constructs a tree. These trees are used to construct a larger tree, giving a close approximation to the actual tree that would be built. This requires only two scans of D, allowing good complexity as well.

Perception-based classification is an interactive approach based on multidimensional visualization techniques. It uses a pixel oriented approach, which maps d-dimensional objects to a circle partitioned into d segments, each representing one attribute. Each value is mapped to a pixel, reflecting the object’s class label.

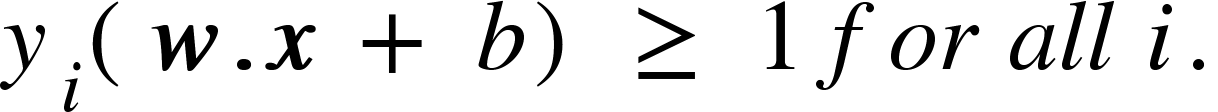
**Support Vector Machines**

Support vector machines use a nonlinear mapping to transform the original data to a higher dimension. Within this new dimension, it searches for an optimal hyperplane separating the data.

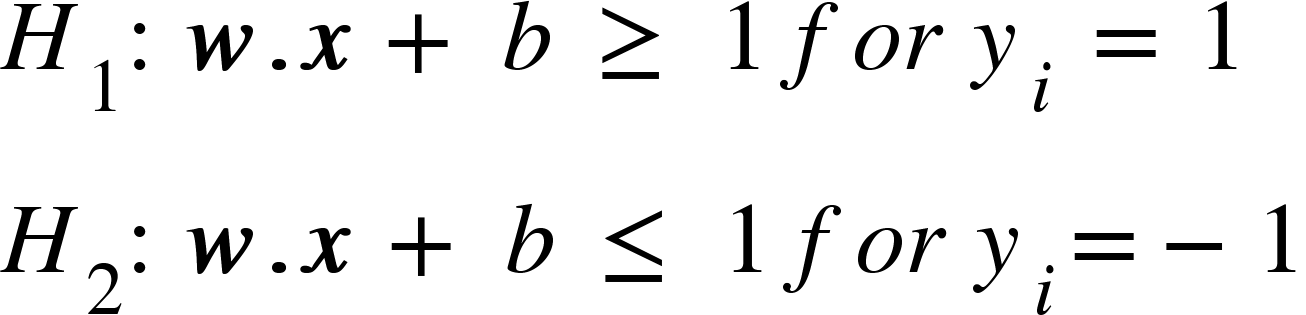
An SVM tries to find the maximum marginal hyperplane, which maximises the distance between the support vectors chosen.

The separating hyperplane can be given as **W.X** + b = 0.

Any point lying above this plane satisfies w1x1+w2x2+..+b > 0 and <0 for points below the plane.

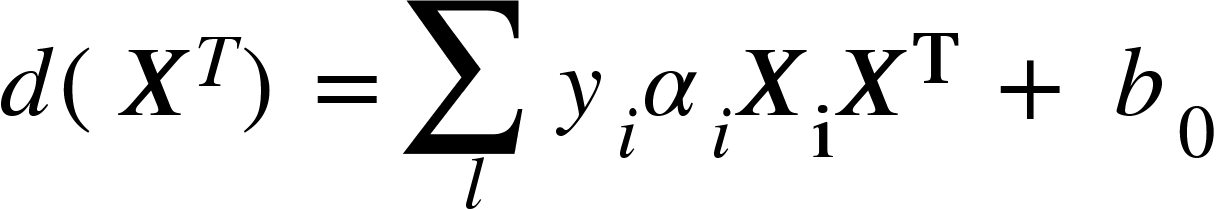
This can be combined as .

The data points lying on the hyperplanes H1 and H2 are called the support vectors where



The margin is thus given by 2/||**W||.**

The classification of a new datapoint can be given by the Lagrangian form



This is only when the data is linearly separable.

When data is linearly inseparable, we transform the data into a higher dimensional space, and then a linear separating hyperplane is found. This nonlinear mapping function can be done by applying a kernel function K(Xi,Xj) to the original data as f(Xi).f(Xj).

Some kernel functions that can be used are

<math xmlns="http://www.w3.org/1998/Math/MathML"><mi>P</mi><mi>o</mi><mi>l</mi><mi>y</mi><mi>n</mi><mi>o</mi><mi>m</mi><mi>i</mi><mi>a</mi><mi>l</mi><mo>&#xA0;</mo><mi>k</mi><mi>e</mi><mi>r</mi><mi>n</mi><mi>e</mi><mi>l</mi><mo>&#xA0;</mo><mi>o</mi><mi>f</mi><mo>&#xA0;</mo><mi>d</mi><mi>e</mi><mi>g</mi><mi>r</mi><mi>e</mi><mi>e</mi><mo>&#xA0;</mo><mi>h</mi><mo>:</mo><mo>&#xA0;</mo><mi>K</mi><mfenced><mrow><msub><mi>X</mi><mi>i</mi></msub><mo>,</mo><msub><mi>X</mi><mi>j</mi></msub></mrow></mfenced><mo>=</mo><msup><mfenced><mrow><msub><mi>X</mi><mi>i</mi></msub><mo>&#xB7;</mo><msub><mi>X</mi><mi>j</mi></msub><mo>+</mo><mn>1</mn></mrow></mfenced><mi>h</mi></msup><mspace linebreak="newline"/><mi>G</mi><mi>a</mi><mi>u</mi><mi>s</mi><mi>s</mi><mi>i</mi><mi>a</mi><mi>n</mi><mo>&#xA0;</mo><mi>r</mi><mi>a</mi><mi>d</mi><mi>i</mi><mi>a</mi><mi>l</mi><mo>&#xA0;</mo><mi>b</mi><mi>a</mi><mi>s</mi><mi>i</mi><mi>s</mi><mo>&#xA0;</mo><mi>f</mi><mi>u</mi><mi>n</mi><mi>c</mi><mi>t</mi><mi>i</mi><mi>o</mi><mi>n</mi><mo>&#xA0;</mo><mi>k</mi><mi>e</mi><mi>r</mi><mi>n</mi><mi>e</mi><mi>l</mi><mo>:</mo><mo>&#x2009;</mo><mi>K</mi><mfenced><mrow><msub><mi>X</mi><mi>i</mi></msub><mo>,</mo><msub><mi>X</mi><mi>j</mi></msub></mrow></mfenced><mo>=</mo><msup><mi>e</mi><mrow><mo>-</mo><mfrac><mrow><mo>|</mo><mo>|</mo><msub><mi>X</mi><mi>i</mi></msub><mo>-</mo><msub><mi>X</mi><mi>j</mi></msub><mo>|</mo><msup><mo>|</mo><mn>2</mn></msup></mrow><mrow><mn>2</mn><msup><mi>&#x3C3;</mi><mn>2</mn></msup></mrow></mfrac></mrow></msup><mspace linebreak="newline"/><mi>S</mi><mi>i</mi><mi>g</mi><mi>m</mi><mi>o</mi><mi>i</mi><mi>d</mi><mo>&#xA0;</mo><mi>k</mi><mi>e</mi><mi>r</mi><mi>n</mi><mi>e</mi><mi>l</mi><mo>:</mo><mo>&#xA0;</mo><mi>K</mi><mfenced><mrow><msub><mi>X</mi><mi>i</mi></msub><mo>,</mo><msub><mi>X</mi><mi>j</mi></msub></mrow></mfenced><mo>&#xA0;</mo><mo>=</mo><mo>&#xA0;</mo><mi>tan</mi><mi>h</mi><mfenced><mrow><mi>&#x3BA;</mi><msub><mi>X</mi><mi>i</mi></msub><mo>&#xB7;</mo><msub><mi>X</mi><mi>j</mi></msub><mo>-</mo><mi>&#x3B4;</mi></mrow></mfenced></math>