

Lecture 9 — 17 Sept 2007

Lecture: Naren Ramakrishnan

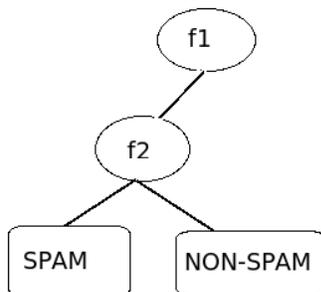
Scribe: Mahima Gopalakrishnan

1 Overview

We move from itemset mining and associations to the next topic: classification. In classification, we are given a set of labeled examples (labeled with the *class* information) and we desire to learn a classifier that predicts the class for new examples (instances). One of the common classification methods is a decision tree which can be viewed as a compaction of many rules.

2 Decision Trees

A decision tree is a sequence of conditions factored into a tree-structured series of branches. The following (partial) tree is an example of a classifier that can predict whether an email is SPAM or NON-SPAM based on some features.



In the above figure f_1 and f_2 are questions that could be as simple as a "YES" or "NO" question or a complex one that could give us a range of values. But how do we know the exact order in which the questions must be posed? This introduces the complexity involved in inducing a decision tree.

2.1 Choosing the Attributes

2.1.1 Splitting Criteria

To know the order in which attributes must be chosen to split the data, we need some measure that would allow us to compare the attributes on some scale and choose one above the other. One of the measures for selecting the "best" question or attribute is based on the level of *Impurity* in the resulting classes of data. *Impurity* could be defined as the amount of uncertainty present in the data and that the attribute which reduces the impurity most should be chosen.

- Examples of pure datasets: {"All emails are classified SPAM"}, {"All emails are classified NON-SPAM"}
- An example of an impure dataset: {"Mix of SPAM and NON-SPAM emails"}

Given probability p , some of the impurity measures studied in the literature are:

- Gini Index: $2p(1-p)$
- Entropy: $-[p \log p + (1-p) \log (1-p)]$
- Misclassification Rate: $1 - \max(p, 1-p)$

In the context of our example, we can define

P(spam): Probability of an email being a SPAM = p and

P(non-spam): Probability of an email being a NON-SPAM = $1-p$.

In general, when the dataset could be divided into two classes, then p is the proportion of instances in the database that has one value for the target attribute and $1-p$ is the proportion of instances in the database that has the second value for the same target attribute.

2.1.2 Generalization of the Impurity Measures

In the previous section, we assumed the instances in the database could be classified two-ways. When the number of classes becomes three or more i.e. C_1, C_2 and C_3 , where

$$P(C_1) = p$$

$$P(C_2) = q$$

$$P(C_3) = 1-p-q$$

then the impurity measures could be generalized as follows.

- Gini Index

$$\sum_{i,j,i \neq j} p_i p_j = 1 - \sum_i p_i^2$$
- Entropy

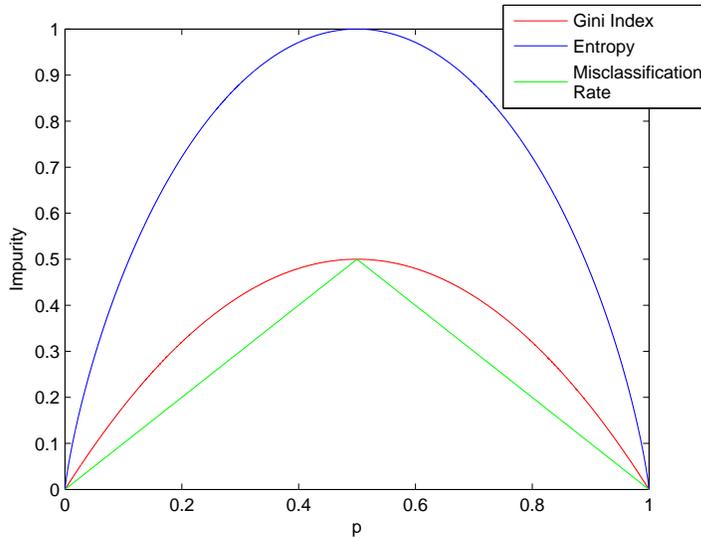
$$-[p \log p + q \log q + (1-p-q) \log (1-p-q)]$$
 or

$$-\sum_i p_i \log p_i$$
- Misclassification Rate

$$1 - \max(p, 1-p, 1-p-q)$$

2.1.3 A Comparison of the Impurity Measures

A comparison of the values of the three impurity measures, Gini Index, Entropy and Misclassification Rate, is given below.



Observe that the data is considered to be pure when $p = 0$ or $p = 1$, and impure otherwise. The data is maximally impure when $p = 0.5$ i.e. the uncertainty of classification is highest at $p = 0.5$. When $p = 0.5$, the entropy becomes 1-bit.

$$H(C) = -[2 \times \frac{1}{2} \log \frac{1}{2}] = 1 \text{ bit.}$$

3 Reduction in Impurity

We will proceed with adopting entropy as the impurity measure. We describe how to assess the reduction in impurity due to knowing the value of some attribute.

Let $H(C)$ be the entropy of the database where C denotes the classification label.

$$H(C) = -\sum_i [p_i \log p_i], \text{ where } p_i \text{ is the proportion of instances in the dataset with value } i \text{ for the class label.}$$

For example, let there be 10 records pertaining to 10 emails in the database. If there are 6 emails whose classification is “SPAM” and 4 emails with classification “NON-SPAM”, then the entropy of the database with respect to the classification is:

$$H(C) = -[\frac{6}{10} \log \frac{6}{10} + \frac{4}{10} \log \frac{4}{10}] = 0.29$$

Now, to compare the effect of splitting the database based on different attributes, we need to calculate the *Information Gain* of the data that results from the splitting of various attributes. Here, information gain refers to reduction in entropy.

Let B be the attribute w.r.t. which the information gain is being calculated. Let C_i be the subset of the instances in the database whose value for attribute B is i . Then,

$$(\text{Information Gain})_B = H(C) - \sum_{i \in B} \frac{|C_i|}{|C|} H(C_i)$$

where, $H(C_i)$ is the entropy of C given B. In essence, we calculate entropies of each ‘partition’ of the database induced by the attribute B and weightedly sum these entropies to be used in the above equation.

To select the attribute for the root node of the decision tree, we calculate the information gain for all the attributes B in the database and choose the one that yields the most reduction in entropy. Once the attribute with the highest attribute is chosen, the same procedure is carried out recursively on the subsets of the database that are induced at each level of the tree.

4 BOAT - Decision Trees using Bootstrapping

BOAT is an optimistic approach to build decision trees using a statistical technique called ‘Bootstrapping’. This algorithm’s main idea is to construct a sample tree T' from small subsets of data derived from the original database D and refining that tree during subsequent scanning of all the data.

Typical decision tree construction algorithms have two parts: *growth phase* and the *pruning phase*. BOAT concentrates on the first phase, i.e., growth phase. Since a whole database D cannot fit into main memory, a large subset of this database D' is stored in-memory to compute T' . Bootstrapping is then applied on this subset.

4.1 Coarse Splitting Criteria

The aim of bootstrapping is to obtain *coarse splitting criteria*. A coarse splitting criteria reduces the set of possible splitting criteria at every node n . So it is a coarse view of the final splitting criteria.

From the subset of dataset D' , we sample b training datasets with replacements and construct b bootstrap trees according to the procedure outlined above. For each node n , it is checked if the splitting attributes are identical. If not, the node n and its subtree is removed in all the bootstrap trees. Also, if the splitting attribute at n is numerical, then we have b split points from which we can obtain a *confidence interval*, such that the final split point would lie inside this interval. If the splitting attribute at n is categorical, then the subsets induced by the split should be identical in all the b trees. Otherwise, remove n and its subtree in all the b trees. Basically, the b bootstrap trees are overlaid on top of each other and the identical parts alone are selected. This is the sample tree T' .

4.2 Cleanup Phase

Now the T' that results can said to be ‘close’ to the final tree T, only when at every node n in T' and T the splitting attribute, X, is identical. In addition, if X is a numerical attribute, then the final split point, x , at n in T, should be inside the confidence interval of x in T. If X was a categorical attribute, then the sample splitting criteria at n should be identical in T and T' .

In this phase, the algorithm takes the sample tree T' and the coarse splitting criteria, and tries to compute the final splitting criteria by making one scan over the training database. The algorithm chooses that criteria that minimizes the value of the impurity function at every node. If the splitting attribute at node n in T' is categorical, then there is a splitting criteria already computed. The algorithm then checks if the sample

splitting subsets are equal to the final splitting subsets.

If the sample splitting attribute at n is numerical, then the final split point should lie inside its confidence interval with a high probability. Let x' be the value of the sample splitting attribute with minimum value, i' , for the impurity function. To prove that x' does not lie outside its confidence interval, the global minimum of the impurity function is calculated and compared with i' . They use this global minimum to also prove that the sample splitting attribute, X , is the final splitting attribute too.

Therefore, this method corrects for any difference between the sample and the final decision trees by checking the validity of the splitting criteria during the clean-up phase itself. So in the end the method outputs the final tree, which would be the same tree as if it were constructed by scanning the database once for every node.

References

- [1] Johannes Gehrke, Venkatesh Ganti, Raghu Ramakrishnan and Wei-Yin Loh, *BOAT Optimistic Decision Tree Construction*, Proceedings of the 1999 ACM SIGMOD international conference on Management of data, 169–180, 1999.