Implementierungstechniken für Hauptspeicherdatenbanksysteme Classification: Decision Trees

Dominik Vinan

February 6, 2018

Abstract

Decision Trees are a well-known part of most modern Machine Learning toolboxes. This work gives a general overview over the most popular kinds of decision trees and explains the ID3 algorithm in detail while examining the possible implementation in C++ and SQL.

1 Introduction

Decision trees are an example for supervised learning. In this case *learning* is not about saving an recalling information but to infer the problems' or datas' underlying structure and generalize it in order to recognize patterns or predict results in the future. For this work we focus on classification using Decision Trees.

The aim of decision tree algorithms is to maximise the classication accuracy while minimizing the amount of decisions that have to be made in order to classify a dataset. This is not easy as the data might contain noise and inconsistencies which makes perfect classification difficult. To cope with this problem, decision trees often use a greedy approach.

This work presents the most common forms of decision trees that are used for classification and implement one example in SQL such that it can be used in a database frontend.

The following sections will give an introduction into the three different types of decision tree approaches while going deeper on TDIDT with the algorithms ID3 and C4.5.

2 Decision Tree Basics

The different forms of decision trees very often use measures of purity. The following paragraphs provide background information about the concept of purity and the difference between Classification- and Regression Trees.

Purity Every node in a decision tree splits the currently processed data into two or more (depending on algorithm) subbranches. The main purpose of those algorithms is to find a useful splitting point for data which leads to correct classification with a high likelyhood. To find suitable values for splitting, many algorithms use measures of purity like the Gini Index or Entropy (while a lower value means more purity). A pure value has probability distribution near zero or one while an equally distributed value would be consideres as impure. Those concepts information purity measuring concepts can be used to determine good splitting variables.

Difference: Classification- & Regression Trees The main difference between those kinds of trees is the target variable. When decision trees are used for a categorial target valriable, they are called *Classification Tres.* In contrast, they are called *Regression Trees* if the target variable is continuous. Both kinds of trees can be used for both, continuous and categorial attributes.

3 Overview: Tree Learning Approaches

There are different tree learning approches that will be explained in the following sections.

3.1 CHi-square Automatic Interaction Detectors (CHAID)

The main difference to other algorithms as CART, ID3 or C4.5 is that CHAID avoids pruning as it stops the tree growth before it becomes too complex. On the other hand it is similar to ID3 and its successors, as it supports more than binary splits which also limits the tree growth from the beginning.

Chi-Squared Test The hypothensis-based Chi-Squared Test is used to find out if a dataset follows a certain probability distribution. However, it does not uncover any relationships between the data. The main question that this test answers is if the observed distribution differs from the expected distribution of data and how much it does differ.

3.2 Top Down Induction of Decision Trees (TDIDT)

Probably the most widespread way of creating decision trees is the top down induction of decision trees. Examples for TDIDT are the CART Algorithm, ID3 or C4.5. One characteristic of TDIDTs are the need for pruning as trees can become overly complex. As TDIDT is a non-Iterative approach, everytime the training data changes, the decision tree has to be calculated from the beginning. Classification and Regression Trees (CART) is also an Umbrella Term that is sometimes used as a synonym for TDIDT or the way round.

3.3 Spatial Decision Tree Learning (SDTL)

Another form of decision tree class are the Spatial Decision Trees that are used to find interesting relations in data like geographic images. A major concern of this kind of decision trees is the reduction of *Salt-And-Pepper Noise* and the misclassification rate. One kind of a SDTL is described in the paper [2] which also shows examples of beforementioned noise.

4 Iterative Dichotomiser 3 (ID3)

In his publication [3], quinlan presents the ID3 Algorithm and provides some example calculations for his splitting criteria. The following sections explain his splitting criteria while the names are sometimes changed for explanation.

4.1 Entropy and weighted Entropy

Entropy is a measure for an attributes' purity, i.e how well does a given attribute separate training samples according to their target classification. Quinlan uses a form of Entropy for the target attriubte which he calls $I(p, n)$ for the whole

tree branch. For the calculatio of his Information Gian, he uses a weighted form $Entropy(A)$ which calculates the sum of Entropies for every potential tree branch that would be created after splitting. For $I(p, n)$, p_p is the probability of p, which would be $\frac{5}{14}$ in a dataset with 14 rows and 5 times a positive outcome "Yes" as target value. On the other hand p_n would be the probability of the "No" cases, $\frac{9}{14}$ in the previous example.

$$I(p, n) = -p_p log_2 p_p - p_n log_2 p_n$$

$$Entropy(A) = \sum_{i=1}^{v} \frac{p_i + n_i}{p + n} I(p, n)$$

4.2 Information Gain

The Information Gain is is the Entropy of the current branch minus the weighted sum of Entropy-values of every subbranch if the tree was splitted at the current attribute. However, the information gain has a bias towards variables with many possible values (e.g. a timestamp column) which should be handled in a real-world dataset in order to avoid misclassification.

$$Gain(A) = I(p, n) - Entropy(A)$$

4.3 SplitInformation

To fight the Information Gains' bias, an addition to the pure Information Gain is the Gain Ratio which additionally uses a measure that is sometimes called *Split-Information* while Quinlan calls it IV in his publication [3]. Practically it is the Entropy of the current attribute with respect to its attribute values instead of the target attribute.

$$SplitInformation(A) = -\sum_{i=1}^{v} \frac{p_i + n_i}{p + n} log_2(\frac{p_i + n_i}{p + n})$$

4.4 Gain Ratio

After obtaining the value for the SplitInformation we can calculate the Gain Ratio which should correct the beforementioned bias of the Information Gain. However it is not always defined and an algorithm should catch the case that SplitInformation is 0.

$$GainRatio(A) = \frac{Gain(A)}{SplitInformation(A)}$$

4.5 Pruning

Even though the ID3 algorithm supports splits at multiple attribute values, the produced trees can still become overly complex which leads to overfitting and therefore an increase of missclassification. To solve this problem there is pruning. There are two general pruning approaches, namely *Pre-Pruning* and *Post-Pruning*.

Pre-Pruning *Pre-Pruning* limits the tree complexity beforehand as it introduces a stopping criterion, e.g tree depth. The problem with this approach is that it might stopp at the wrong point which leads to a bad classification rate of the resulting decision tree.

Post-Pruning The most commonly used form of pruning is *Post-Pruning* which takes overly complex subbranches from the decision tree and transforms it into leaves. While this approach increases the misclassification rate for the training data, it decreases it for an unknown tuple.

5 C4.5

As the successor of ID3, the C4.5 algorithm basically follows the same approach but includes features that makes the C4.5 more useful in pratcial applications than ID3. An introduction to the C4.5 Algorithm and the sourcecode can be found in [4]. A good summary of the differences between C4.5, ID3 and CART can be found in [1].

Improvements over ID3

The C4.5 Algorithm is known as ID3s successor and combines features from CART and ID3 such as being able to use numeric values, handle missing values and a more efficient pruning method as explained in the following sections.

Numeric Values As we learned earlier, the ID3 algorithm can only distinguish discrete attribute-classes while decisions are often based on data that has both discrete classed and numeric values. To adapt to this fact, the C4.5 algorithm was extended to also support both kinds of data.

Missing Values While ID3 needed always complete datasets, the C4.5 Algorithm also supports missing datasets which are ignored.

Pruning Pruning speed and quality is improved.

6 CART (Algorithm)

Another member of the TDIDT Family is the CART Algorithm. The CART Algorithm can be used for growing Classification and Regression Trees and produces always binary decision trees i.e. every node has only two child nodes.

Gini Index

The impurity measurement for CART is the Gini Index. A higher Gini-Index means higher impurity of data while a lower Gini-Index means more purity and makes an attribute more suitable for a split. The Gini Index has a bias towards variables with larger partitions of possible values.

7 Implementation ID3

This section describes the ideas behind the example implementations for both C++ and SQL.

7.1 ID3 Algorithm

The following pseudocode was used as base for an Databasesystem-near implementation of the ID3 Algorithm. While it is close to algorithms that are available in books like it was fitted for the usage with SQL and C++. As we usually have the primary key in systems, we can let the user state it directly in order to ignore it for decision tree learning and use it for for partitioning.

Data: Database with primary key, target attribute and training columns
Result: ID3-Decision Tree
Initialization: Select primary key, target attribute and training columns;
if *Not (pure or stopping criterion satisfied)* **then**
  Calculate **Entropy** & **Information Gain** for every attribute;
  Select attribute **A** with **MAX(Information Gain)**;
  Make a tree node using attribute **A**;
  Split dataset into subsets for every attribute-value of **X**;
  Recursive call of ID3-Algorithm;
end

Algorithm 1: ID3 Pseudocode

7.2 SQL Implementation

The General idea was to have functions that can apply the ID3 Algorithm on a database with categorial values. After providing the names of the database, target attribute and training attribute columns, the function *infodelta* calculates the information gain for every attribute. The recursive call is not implemented at the moment.

Future work A major problem with the algorithm is the recursive call, as the functions have specified input and output columns. The suggested solution are two additional parameters that store the currently interesting columns and rows (as it is done in the C++ implementation) .

7.3 C++ Implementation

The general idea behind the C++ implementation is a algorithm implementation on a database that is implemented as a two-dimensional vector. The program supports databases of different size in columns and rows. While it only supports categorial variables that have to coded as string.

Future work The tree creation using the recursive function call could be better.

8 Strengths and Weaknesses of Decision trees

Being a relatively simple but strong part of a Maschine Learning toolbox decision trees have different strengths and weaknesses which should be considered when choosing the algorithm for a given Problem.[1]

Advantages of Decision Trees Decision Trees create rules that are human readable and easily understandable. The computation power that is needed for applying (not growing) decision trees is limited. As we have already learned from algorithms like C4.5, some Decision Tree Algorithms are capable of handling both, continuous and categorial variables.

[1] This section is roughly based on [5]

Disadvantages of Decision Trees While decision Trees are a valuable tool for classifying into categories, they are not the best choice when it comes to predict a certain value. Decision Trees need sufficient training data, preferably with a limited amount of classes as Decision Trees are prone to error when the training data contains many classes and a relatively small amount of examples. Growing and Pruning decision trees are computationally expensive. While Decision trees are very strong in classifying rectangular values they don't handle non-rectangular regions very well.

References

[1] Badr Hssina, Abdelkarim Merbouha, Hanane Ezzikouri, and Mohammed Erritali. A comparative study of decision tree id3 and c4. 5. *International Journal of Advanced Computer Science and Applications*, 4(2), 2014.

[2] Zhe Jiang, Shashi Shekhar, Xun Zhou, Joseph Knight, and Jennifer Corcoran. Focal-test-based spatial decision tree learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(6):1547–1559, 2015.

[3] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[4] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[5] Peter Waiganjo Wagacha. Induction of decision trees. *Foundations of Learning and Adaptive Systems*, 2003.