Towards an Effective Cooperation of the User and the Computer for Classification

Mihael Ankerst, Martin Ester, Hans-Peter Kriegel
Institute for Computer Science, University of Munich, Oettingenstr. 67, D-80538 München, Germany
{ ankerst | ester | kriegel }@dbs.informatik.uni-muenchen.de

ABSTRACT

Decision trees have been successfully used for the task of classification. However, state-of-the-art algorithms do not incorporate the user in the tree construction process. This paper presents a new user-centered approach to this process where the user and the computer can both contribute their strengths: the user provides domain knowledge and evaluates intermediate results of the algorithm, the computer automatically creates patterns satisfying user constraints and generates appropriate visualizations of these patterns. In this cooperative approach, domain knowledge of the user can direct the search of the algorithm. Additionally, by providing adequate data and knowledge visualizations, the pattern recognition capabilities of the human can be used to increase the effectivity of decision tree construction. Furthermore, the user gets a deeper understanding of the decision tree than just obtaining it as a result of an algorithm. To achieve the intended level of cooperation, we introduce a new visualization of data with categorical and numerical attributes. A novel technique for visualizing decision trees is presented which provides deep insights into the process of decision tree construction. As a key contribution, we integrate a state-of-the-art algorithm for decision tree construction such that many different styles of cooperation - ranging from completely manual over combined to completely automatic classification - are supported. An experimental performance evaluation demonstrates that our cooperative approach yields an efficient construction of decision trees that have a small size, but a high accuracy.

1. INTRODUCTION

Classification is one of the major tasks of data mining. The goal of classification [12] is to assign a new object to a class from a given set of classes based on the attribute values of this object. Furthermore, classification is based on some discovered model which forms an important piece of knowledge about the application domain. Over the years, many classification algorithms have been proposed such as decision tree classifiers which have become very popular. From a training set, i.e. a set of objects for which their attribute values and their correct class is known, a decision tree classifier learns a discrete-valued classification function which is represented by a decision tree.

Although classification is sometimes also called “supervised” learning, in general, the present classification algorithms provide only very limited forms of guidance or “supervision” by the user. Typically, the user selects the dataset and sets the values for some parameters of the algorithm - which are often difficult to determine a priori. Then the algorithm runs for some considerable time and returns a pretty large set of patterns. The user does not obtain any intermediate results and has no possibility to intervene during the execution of the algorithm. We argue that the user should be involved more interactively in the process of classification because of the following reasons:

(1) By providing adequate data and knowledge visualizations, the pattern recognition capabilities of the human can be used to increase the effectivity of decision tree construction.

(2) Due to their active envolvement, the users have a deeper understanding of the resulting decision tree. Thus, the trust into the data mining system can be greatly improved (c.f. [15] which argue analogously in favour of boosted naive Bayes classification).

(3) When obtaining intermediate results from the algorithm, the user can provide domain knowledge to focus the further search of the algorithm. Using domain knowledge has been recognized as a promising approach for constraining knowledge discovery and for avoiding overfitting [6].

Our goal is an effective cooperation between the user and the computer so that both contribute what they do best:

- The user specifies the task, focuses the search, evaluates the (intermediate) results of the algorithm and provides his domain knowledge which is difficult to formalize and to incorporate into an algorithm.
- The computer runs an algorithm to automatically create (intermediate) patterns satisfying the specified user constraints and creates appropriate visualizations of these patterns for the user.

We believe that this cooperation will greatly improve the effectiveness as well as the efficiency of classification and other data mining algorithms. For example, [8] explored this approach for the task of mining association rules. The user specifies a set of constraints which are pushed deeply into the data mining algorithm to focus its search as far as possible. In a recent paper [1], a new approach for the task of classification was presented. PBC (Perception-Based Classification) was introduced as an interactive decision tree classifier based on a multidimensional visualization technique. However, the only task of the computer was to visualize the training data, and the user selected the split attributes and split points and thus constructed the decision tree manually.

In this paper, we explore the full potential of an effective cooperation between the user and the computer. While the first version of PBC supported only numerical attributes, PBC now also handles categorical attributes and thus supports a much broader range of applications. We present a new technique for visualizing decision trees which provides more insights into the process of decision tree construction than previous techniques. Most importantly, we integrate a state-of-the-art algorithm for decision tree construction such that many different styles of cooperation (ranging from completely manual over combined to completely automatic classification) are supported. For example, the user may choose to construct the most important top levels of the decision tree manually and run an algorithm for the further tree expansion to speed-up the decision tree construction. To demonstrate the practical impact, we conduct an extensive performance evaluation comparing different styles of cooperation.

The rest of this paper is organized as follows. Section 2 reviews related work from the areas of decision tree classification and visualization. In section 3, we introduce our techniques of visualizing the data and the knowledge. Section 4 presents the integration of an algorithm for cooperative decision tree construction. Section 5 reports the results of an extensive experimental evaluation on several well-known datasets. The last section summarizes this paper and discusses some directions for future research.

2. RELATED WORK

A decision tree classifier constructs a tree in a top-down fashion performing a greedy search through the very large space of all possible decision trees. At each current node, the attribute that is most useful for the task of classification (with respect to the subset of all training objects having passed all tests on the path to the current node) is selected. Criteria such as the information gain or the gini index have been used to measure the usefulness of an attribute. Domain knowledge about the semantics of the attributes and the attribute values is not considered by the criteria. Note that greedy algorithms for decision tree construction do not allow to backtrack to a previous choice of an attribute when it finally turns out to be suboptimal.

Many different algorithms for learning decision trees have been developed over the last 20 years. For instance, *CART* [4] was one of the earliest systems which, in particular, encorporates an effective pruning phase. Their so-called minimum cost complexity pruning cuts off branches that have a large number of leaf nodes yielding just a small reduction of the apparent error.

SLIQ [11] is a scalable decision tree classifier that can handle both numerical and categorical attributes. In order to make SLIQ scalable for large training sets, special data structures called attribute lists are introduced which avoid sorting the numerical attributes for each selection of the next split. Furthermore, a greedy algorithm for efficient selection of splits of categorical attributes is presented. An experimental evaluation demonstrates that SLIQ produces decision trees with state-of-the-art accuracy and tree size with a much better efficiency for large training sets.

Visual representation of data as a basis for the human-computer interface has evolved rapidly in recent years. The increasing amount of available digital information in all kinds of applications has led to the challenge of dealing with both high dimensionality and large amounts of data. [9] gives a comprehensive overview over existing *visualization techniques* for large amounts of multidimensional data which have no standard mapping into the Cartesian coordinate system.

Recently, several techniques of visual data mining have been introduced. [5] presents the technique of *Independence Diagrams* for visualizing dependencies between two attributes. The brightness of a cell in the two-dimensional grid is set proportional to the density of corresponding data objects. This is one of the few techniques which does not visualize the discovered knowledge but the underlying data. However, the proposed technique is limited to two attributes at the same time. [10] presents a decision table classifier and a mechanism to visualize the resulting *decision tables*. It is argued that the visualization is appropriate for business users not familiar with machine learning concepts.

A commercial system for interactive decision tree construction is *SPSS AnswerTree* [16] which - in contrast to our approach - does not visualize the training data but only the decision tree. Furthermore, the interaction happens before the tree construction, i.e. the user defines values for global parameters such as maximum tree depth or minimum support for a node of the decision tree.

KnowledgeSEEKER [3] is another commercial system for interactive decision tree construction. It offers an intuitive graphical user interface and the results are displayed in the form of a clear and interactive decision tree. Both the sensitivity of the correlation finding and the volume of the information displayed are user-defined. In KnowledgeSEEKER, however, the frequency of class labels is not visualized in connection with the attribute values which severely limits the incorporation of domain knowledge.

PBC (Perception Based Classification) [1] is a prototype system which visualizes the training data in order to support interactive decision tree construction. A novel method is introduced for visualizing multi-dimensional data with a class label such that their degree of impurity with respect to class membership can be easily perceived by a user. The method performs pixel-oriented visualization and maps the classes to colors in an appropriate way. The basic steps of one interaction loop may be divided into the data interaction steps and the knowledge interaction steps. The data interaction steps include the visualization of the data and the methods for manually selecting an attribute and its split points. The knowledge interaction steps cover the representation of the current decision tree which is expanded after every attribute split and enable the user to operate on the nodes in terms of assigning a class label, invoking the visualization of any other node or backtracking to some previously created node of the decision tree.

Recently, several papers have investigated the relation between accuracy and comprehensibility in data mining and the approach of integrating the user into the process of knowledge discovery. [15] stresses the importance of the comprehensibility of discovered knowledge in order to build trust into the data mining system. A variant of the boosted naive Bayes classifier is introduced which creates understandable explanations of the results while retaining a high predictive performance. [6] concludes that greater simplicity does not typically lead to greater accuracy in data mining. Therefore, the search space cannot be simply restricted to simple models and it is proposed to use domain knowledge for constraining the search of a data mining algorithm. It is argued that often even weak and general domain knowledge is sufficient for this purpose. [8] presents a general approach to constrained association rule mining. The user specifies a set of constraints on the data, the level of abstraction, the rules and their interestingness. These constraints are pushed deeply into the data mining algorithm to focus its search as far as possible.

3. VISUALIZING THE DATA AND THE KNOWLEDGE

The techniques of visualization introduced in [1] have two major shortcomings: First, only data with numerical attributes can be visualized. Second, the decision tree is visualized in a standard way which is not related to the data visualization. In this chapter, we present new techniques overcoming these drawbacks and offering a much broader functionality. Categorical attributes rise special problems and a method for their visualization is outlined in section 3.1. In section 3.2, a new technique of visualizing decision trees is introduced which is based on our technique of data visualization.

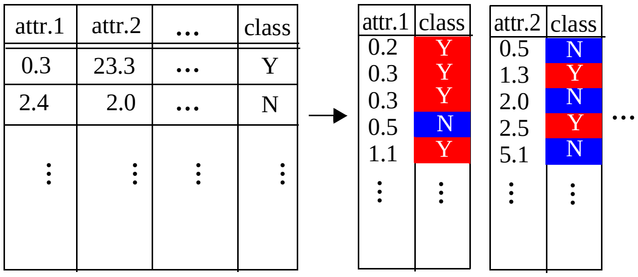
3.1 Techniques for Data Visualization

Our technique for data visualization is based on two main concepts:

- Each attribute of the training data is visualized in a separate area of the screen.
- The different class labels of the training objects are represented by different colors.

In the following, these conecpts are elaborated in more detail.

In our approach, the training data records are mapped to attribute lists containing one entry (attribute value, class label) for each of the training records (cf. figure 1). Note that the entries of each attribute list are sorted in ascending order of attribute values. Figure 1 also illustrates a possible color coding of the different class labels. Thus, sequences of consecutive attribute values sharing the same class label can be easily identified. For example, we observe that attribute 1 is a better candidate for a split than attribute 2 and “attribute 1 < 0.4” yields a good split w.r.t. the training data.



attr.1	attr.2	...	class
0.3	23.3	...	Y
2.4	2.0	...	N
⋮	⋮	⋮	⋮

→

attr.1	class
0.2	Y
0.3	Y
0.3	Y
0.5	N
1.1	Y
⋮	⋮

attr.2	class
0.5	N
1.3	Y
2.0	N
2.5	Y
5.1	N
⋮	⋮

...

Figure 1. Mapping the training data records to attribute lists

Bar visualization of the data

For the data visualization, the different attributes have to be mapped to different areas of the screen. The Circle Segments technique [2] has been proposed to arrange the attribute values of different attributes in different segments. To better support the cooperation of the user and the computer, we introduce a new visualization of the data which has two advantages. The first advantage is based on a perceptional issue. When the user selects a splitting attribute based on his perception, he often has to consider the size of a partition for comparison with the size of another partition. Due to the constant height of the bars, it is much easier to estimate the partition size using the bar visualization than using the circle segments visualization. The second reason is that our new decision tree visualization is based on a bar visualization and, consequently, it is convenient to provide similar data visualizations in order to decrease the learning time for the user.

The bar visualization is performed as follows. Within a bar, the sorted attribute values are mapped to pixels in a line-by-line fashion according to their order. Each attribute is visualized independently from the other attributes in a separate bar. Figure 2 illustrates the method of the bar visualization for the case of two attributes.

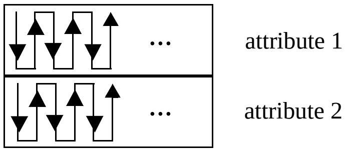


Figure 2. Illustration of the bar visualization

The amount of training data that can be visualized by the bar technique is determined by the product of the number of data records and the number of attributes. For representing all attributes at the same time (without scroll-bars), this product minus the number of pixels used for the border lines of the bars may not exceed the resolution of the window. All the STATLOG datasets used in our experimental evaluation (see section 5) can be easily visualized with the bar technique, for example, the DNA training data consisting of 2000 data records can be visualized with all their 180 attributes.

Handling categorical attributes

Each attribute is sorted separately and the induced order is used for the arrangement of the pixels. This approach is natural for numerical attributes, but in the case of categorical attributes with no implicit order the same approach is obviously suffering several drawbacks. In a naive approach, we would simply map different categories to different numbers and as a consequence the induced order would have a major impact on the user’s capability to perceive the goodness of split points in this attribute. Furthermore, even if a cluster in a set of non adjacent categories could be perceived it would not be possible to select these categories for one subtree of the current node. Instead, this attribute would have to be split in many categories yielding an unnecessary number of son nodes.

Hence a crucial question for the visualization of categorical attributes is the sorting of the categories. Algorithmically searching for the best split point within a categorical attribute is expensive. For an attribute with n different categories, several heuristics have been proposed to find a solution in the space of $2^{n-1} - 1$ possible binary partitions (if we limit the search space to binary splits). In [7], SLIQext, an extension of a previously proposed algorithm for splitting categorical attributes is evaluated. SLIQext searches for the best binary split by starting with an empty first set and a second set containing all categories. Then, it moves in a greedy fashion that element from the second set to the first set which yields the best split. This process iterates until the second set becomes empty and finally the best partition among all the considered partitions is chosen.

Since good results were reported for SLIQext, we use this order in which the categorical values are moved from the second set to the first set. Thus, we group the categories together that belong to the same set and support the user in detecting and evaluating the goodness of a split in that attribute. Since this order serves as a default order of the categories, the user can always select the split point that would be computed by SLIQext by exactly setting one split point. On demand, the category borders can be highlighted. Furthermore, we increase the power in two ways: First, the user is enabled to change the sorting of the categories by manually dragging them to a desired position. Second, as for numerical attributes, the user is not restricted to set just one split point for one attribute, but he can set an arbitrary number of split points.

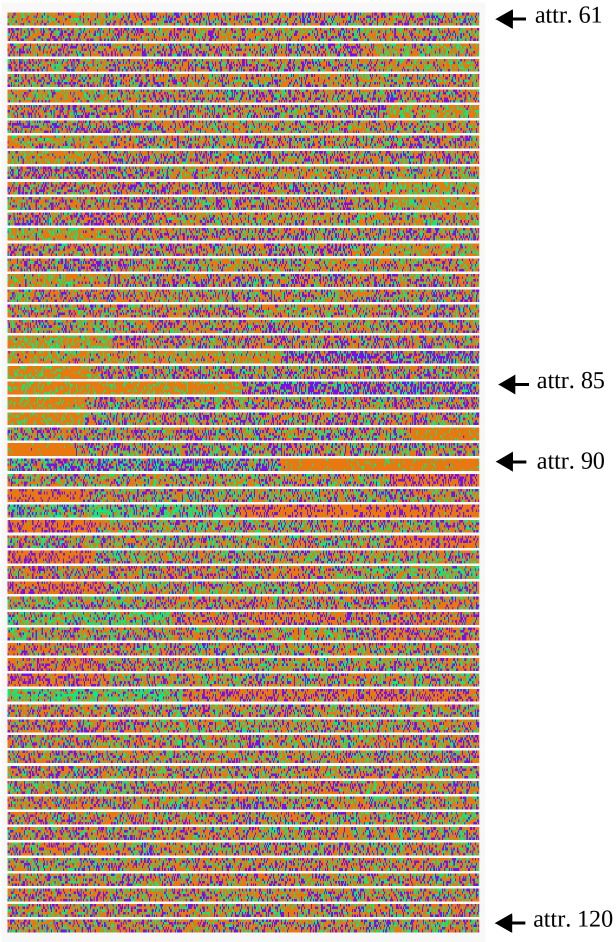


Figure 3. Visualization of the DNA training data

Figure 3 depicts the visualization of the DNA training data from the STATLOG benchmark [13] which have only categorical attributes (note that this figure has been scaled down). As suggested in the description of the training data, only 60 of the 180 attributes were used. The visualization indicates that attributes 85 and 90 are good candidates for splitting. In fact, attribute 90 is chosen as the optimal one if using the gini-index to measure the purity of the resulting partitions.

3.2 A New Technique for Knowledge Visualization

We propose a new visualization technique for decision trees which is based on the bar visualization of the data. This visualization technique does not only depict the decision tree in a clear way but it also provides a lot of explanations why the tree was constructed this way. Furthermore, the proposed technique offers a new possibility to analyze and compare different decision trees constructed by any algorithm performing univariate splits.

In our new visualization technique, each node is represented by the data visualization of the chosen splitting attribute of that node. For each level of the tree, a bar is drawn representing all nodes of this level. The top level bar corresponds to the root node of the decision tree. On lower levels of the tree the number of records and thus the number of pixels is reduced if there are leaves in upper levels - leaves are underlined with a black line. Black vertical lines indicate the split points set in the current bar. On lower levels, partitions of the data inherited from upper levels are marked by white vertical lines at the same horizontal position as the original split point. Attribute and node information at the mouse pointer position (attribute name, attribute value, min., max. value and number of records in this node,...) is displayed on demand. Upon a mouse click, the system switches back to the data visualization of the corresponding node.

Compared to a standard visualization of a decision tree, a lot of additional information is provided which is very helpful in explaining and analyzing the decision tree:

- size of the node (number of training records corresponding to the node)
- quality of the split (purity of the resulting partitions)
- class distribution (frequency and location of the training instances of all classes).

Some of this information might also be provided by annotating the standard visualization of a decision tree (for example, annotating the nodes with the number of records or the gini-index), but this approach clearly fails for more complex information such as the class distribution. We argue that the proposed visualization provides a lot of additional information in a rather compact way.

Figure 4 illustrates the visualization of a decision tree for the Segment training data from the Statlog benchmark [13] having 19 numerical attributes.

4. INTEGRATING ALGORITHMS INTO COOPERATIVE DECISION TREE CONSTRUCTION

We have argued for an effective cooperation between the user and the computer so that both contribute what they do best. Our fundamental paradigm is “the user as the supervisor”, i.e. the system supports the user and the user always has the final decision. In [1], the user has to select the splitting attribute as well as the split points for each node. The approach of interactive (manual) decision tree construction is briefly reviewed in section 4.1. In this approach, the users can incorporate their domain knowledge, but for large decision trees a completely manual construction may become rather tedious. Instead, the strengths of the computer should be fully exploited to unburden the user as far as reasonable. Therefore, in section 4.2 we present several functions of algorithmic support for the user and in section 4.3 we discuss the process of cooperative classification.

4.1 Interactive Decision Tree Construction

Interactive decision tree construction works as follows. Initially, the complete training set is visualized in the *Data Interaction Window* together with an empty decision tree in the *Knowledge Interaction Window.* The user selects a splitting attribute and an arbitrary number of split points. Then the current decision tree in the Knowledge Interaction Window is expanded. If the user does not want to remove a level of the decision tree, he selects a node of the decision tree. Either he assigns a class label to this node (which yields a leaf node) or he requests the visualization of the training data corresponding to this node. The latter case leads to a new visualization of every attribute except the ones used for splitting criteria on the same path from the root. Thus the user returns to the start of the interaction loop. The interaction is finished when a class has been assigned to each leaf of the decision tree.

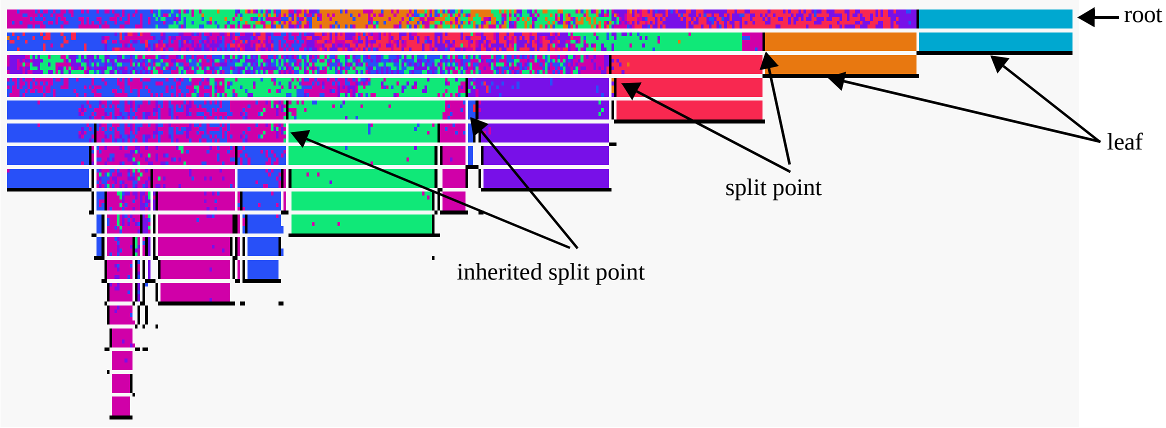


Figure 4. Visualization of a decision tree for the Segment training data

Interactive decision tree construction allows to transfer knowledge in both directions. On one hand, domain knowledge of an expert can be profitably included in the tree construction phase. On the other hand, after going through the interactive construction of a decision tree, the user has a much deeper understanding of the data than just knowing the decision tree generated by an arbitrary algorithm. Interactive visual classification is very effective whenever understandability of the discovered model is as important as classification accuracy. Note that in many situations, for example if the attribute values are measured automatically, accurate classification of new objects is not the only goal but users seek a general understanding of the characteristics and the differences of the classes.

The approach of interactive classification has several additional advantages compared to algorithmic approaches. First, the user may set an arbitrary number of split points which can reduce the tree size in comparison to binary decision trees which are generated by most state-of-the-art algorithms. Furthermore, in contrast to the greedy search performed by algorithmic approaches, the user can backtrack to any node of the tree when a subtree turns out not to be optimal.

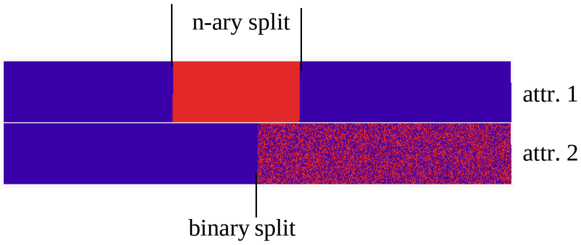


Figure 5. Comparison of n-ary and binary split

Figure 5 illustrates a case where an n-ary split is clearly superior to a binary split. 50,000 synthetic data records from two different classes with two numerical attributes are visualized. The visualization of attribute 1 reveals three pure partitions: 1/3 of the data records from class A, 1/4 of the data records belonging to class B and 5/12 of the data records from class A. The visualization of attribute 2 indicates one pure partition (1/2 of the data records all belonging to class A) and a mixed partition (1/2 of the data records, 50% from class A and 50% from class B). When performing binary splits and using the gini-index to evaluate their quality, the second attribute would be selected and split in the middle. On the other hand, when manually performing n-ary splits, we would choose attribute 1 and split it into the three pure partitions. The ternary split of attribute 1 exploits the whole information contained in this attribute and results in three leaf nodes with 100% purity. The binary split, however, creates only one leaf node and the other subtree has to be further expanded which may result in a considerably larger decision tree.

4.2 Algorithmic Support for the User

An algorithm automatically creates a decision tree for the given training data. This complete functionality as well as parts of this functionality may support the user in interactively constructing a decision tree. Note that any algorithm performing univariate splits, i.e. considering one split attribute at a time, may be integrated into PBC. In the following, we present several types of algorithmic support and discuss their potential benefits. Some of these functions are applied to the so-called *active node*, i.e. the unique node of the decision tree which the user has selected for visualization.

propose split

For a set of attributes selected by the user, the attribute with the best split together with the optimum split point of this attribute is calculated and visualized. If a singleton attribute is specified as input, only the optimum split point for this attribute is determined.

The function *propose split* turns out to be useful in two cases: first, whenever there are several candidate attributes with very similar class distributions and, second, when none of the attributes yields a good split which can be perceived from the visualisation.

look-ahead

For some hypothetical split of the active node of the decision tree, the subtree of a given maximum depth is calculated and visualized with the new visualization technique for decision trees (see section 3.2). This function offers a view on the hypothetical expansion up to a user specified number of levels or until a user specified minimum number of records per node. If the look-ahead function is invoked for a limited number of levels, it is very fast (some seconds of runtime) because no pruning is performed in this case.

The *look-ahead* function may provide valuable insights for selecting the next split attribute. Without the look-ahead function, when there are several candidate attributes the user selects one of them, chooses one or several split points and continues the tree construction. If at a later stage the expanded branch does not yield a satisfying subtree, the user will backtrack to the root node of this branch and will remove the previously expanded subtree. He will select another candidate attribute, split it and proceed. Utilizing the new function, however, the user requests a look-ahead for each candidate attribute before actually performing a split. Thus, the necessity of backtracking may be avoided.

expand subtree

For the active node of the decision tree, the algorithm automatically expands the tree. Several parameters may be provided to restrict the algorithmic expansion such as the maximum number of levels and the minimum number of data records or the minimum purity per leaf node.

The pruning of automatically created subtrees rises some questions. Should we only prune the subtree created automatically or prune the whole tree - including the subtrees created manually? According to our paradigm of the user as a supervisor, pruning is only applied to automatically created trees. We distinguish two different uses of the *expand subtree* function: the maximum number of levels is either specified or it is unspecified. In the first case, the decision tree is usually post-processed by the user. No pruning is performed if a maximum number of levels is specified. Otherwise, if no maximum number of levels is specified, the user wants the system to complete this subtree for him and pruning of the automatically created subtree is performed if desired.

The function *expand subtree* is useful in particular if the number of records of the active node is relatively small. Furthermore, this function can save a lot of user time because the manual creation of a subtree may take much more time than the automatic creation.

4.3 The Process of Cooperative Decision Tree Construction

Depending on the user and on the application, different styles of cooperation between the user and the system may be preferred. In the extreme cases, either the user or the computer would do all the work of decision tree construction, but typically a balance between the two partners is sought. In the following, we discuss different styles of using the PBC system. Figure 6 depicts the process of cooperative decision tree construction with its different steps and their dependencies. Operations of the user and operations of the system are clearly distinguished. Note the central circle marked “User” which can be reached either directly after the visualization of the current decision tree or after the selection of a new active node or after a system provided look-ahead. At this stage, the user has the choice between five different options: (1) remove the active node, (2) assign a class label to the active node, (3) select a split attribute and split points, (4) let the system propose a split attribute and split point or (5) let the system expand the active node. In the following, the process is explained in more detail.

The system is initialized with a decision tree consisting only of the root node as the active node. Later, the active node may be changed by the user at each time. The user always has several alternative options to proceed which can be categorized into two groups:

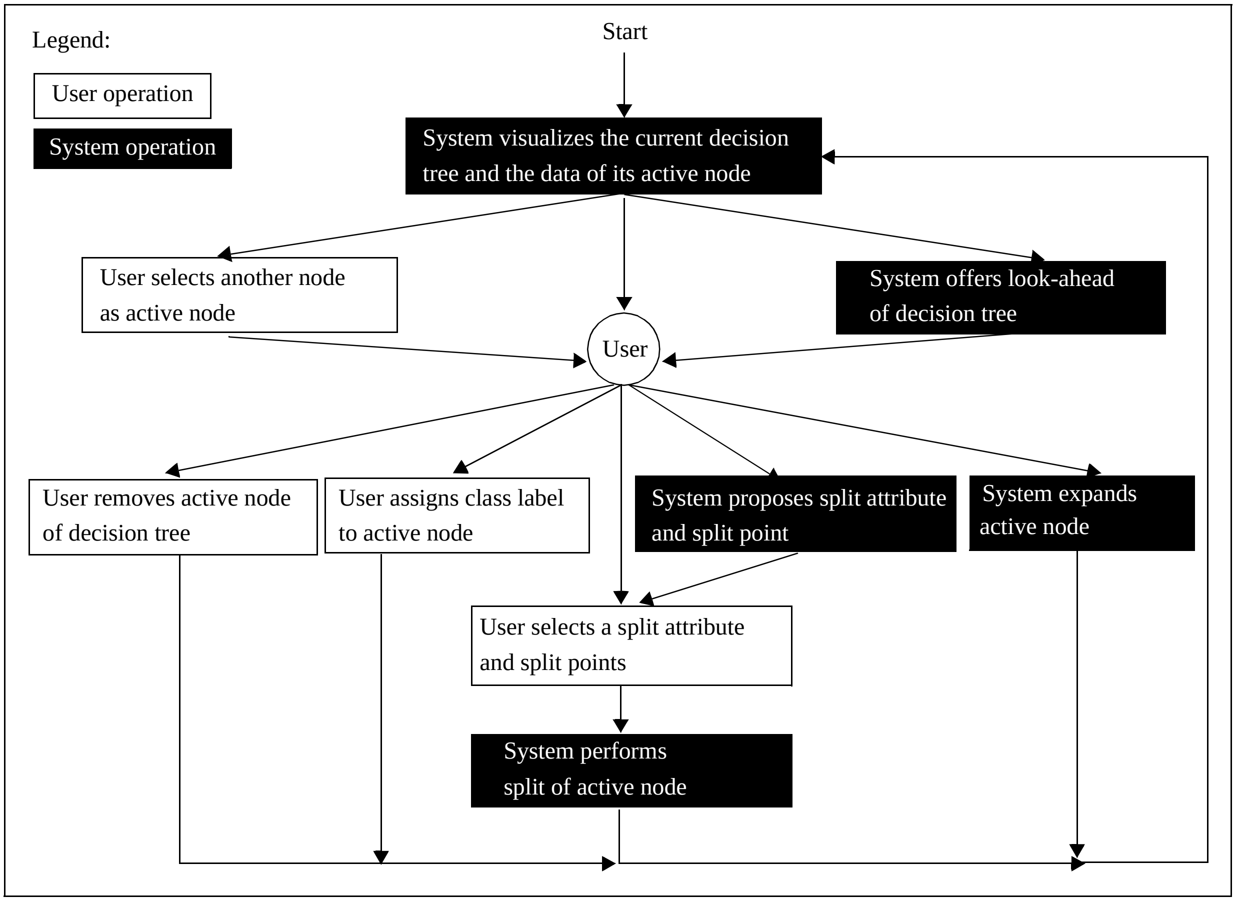

Legend:
User operation
System operation
Start
System visualizes the current decision tree and the data of its active node
User selects another node as active node
System offers look-ahead of decision tree
User
User removes active node of decision tree
User assigns class label to active node
System proposes split attribute and split point
System expands active node
User selects a split attribute and split points
System performs split of active node

Figure 6. The process of cooperative decision tree construction

- *local functions*
  Local functions are functions local w.r.t. the active node and do not impact other nodes of the decision tree. These functions include propose split, look-ahead and split the active node. The user splits the active node, either according to his visual impression or according to the algorithmic proposal. While in the first version of PBC an attribute could be used only once on each path as a split attribute, now it can be used several times as in the standard algorithms.
- *global functions*
  Global functions change the global structure of the decision tree. These functions either remove the active node, make the active node a leaf node by assigning a class label to it or let the system expand the active node up to a given maximum number of levels.

The process of decision tree construction can be terminated by the user whenever he is satisfied with the quality (e.g., the classification accuracy and the tree size) of the result.

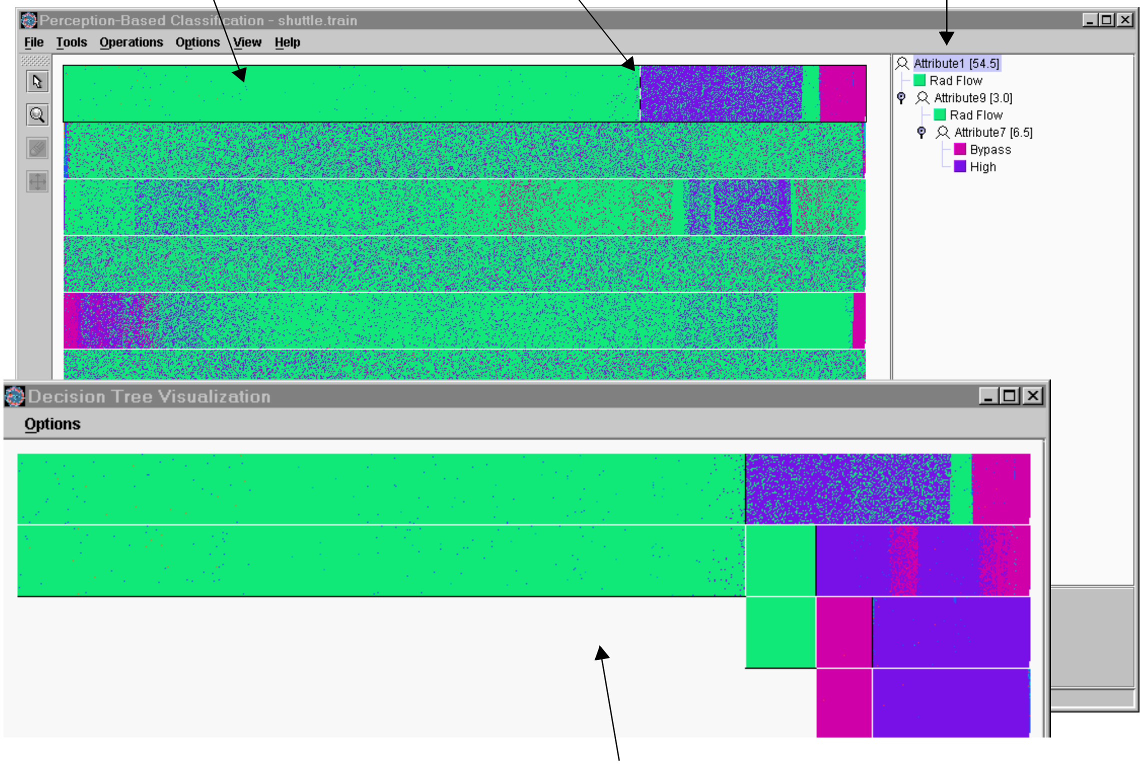
Figure 7 depicts screen shots of the PBC system when constructing a decision tree for the Shuttle data [13] (see also section 5.1). The main window visualizes the data of the active node and depicts the whole decision tree in standard representation. The local functions can be applied on the visualization of the data on the left side, while the global functions can be invoked in the visualization of the decision tree on the right side. The additional window in the foreground depicts the same decision tree using the new technique of visualization.

To conclude this section, our approach of cooperative decision tree construction extends the power of state-of-the-art purely algorithmic approaches in two important ways:

- look-ahead and backtracking
  Due to efficiency reasons, decision tree algorithms choose one split attribute after the other and they do not perform backtracking. In cooperative tree construction, however, the look-ahead function provides an estimate for the global goodness of a split and, furthermore, the user may backtrack to any node of the decision tree as soon as he realizes that some suboptimal subtree has been generated.
- n-ary splits
  State-of-the-art algorithms always perform binary splits. In cooperative tree construction, however, the user may set an arbitrary number of split points which can significantly reduce the tree size and improve the accuracy in comparison to binary decision trees. Of course, binary splits are supported as an important special case.

Bar visualization of the Shuttle training data of the active node with animated split line

Standard representation of the decision tree with higlighted active node (here: root node)



New visualization of the same decision tree (all four levels)

Figure 7. Screen shots of the PBC system

5. EXPERIMENTAL EVALUATION

We performed an experimental evaluation of cooperative decision tree construction on several well-known benchmark datasets. The implementation of the PBC system and the whole test environment is described in section 5.1. In section 5.2, we compare manual decision tree construction using PBC with standard algorithmic approaches and in section 5.3 we compare some different styles of cooperative classification supported by PBC.

5.1 Test Environment

The complete PBC system was implemented in Java using version 2 of the Java Virtual Machine. We could not reasonably integrate some existing software but reimplemented a classification algorithm for proper integration into the PBC system using the existing data structures. Our algorithm realizes the SLIQ tree growth phase (which effectively handles numerical and categorical attributes) and the CART pruning phase (which can be applied using cross-validation as well as using train and test).

For the experimental evaluation, we chose several well-known training data sets used as benchmark data in the STATLOG project [13]. We selected the following data sets representing a good mix of numerical and categorical attributes:

- Shuttle
  43,500 training records with 9 numerical attributes, 7 different classes, train and test.
- Satimage
  4,435 training records with 36 numerical attributes, 6 different classes, train and test.
- Segment
  2,310 training records with 19 numerical attributes, 7 different classes, 10-fold cross-validation.
- Australian
  690 training records with 6 numerical and 8 categorical attributes, 2 different classes, 10-fold cross-validation.
- DNA
  2,000 training records with 60 categorical attributes (as recommended in the dataset description, 60 of the 180 available attributes were used), 3 different classes, train and test.

All experiments were performed on a Pentium Pro 180MHz with 192 MB main memory. For the growth phase of the algorithm, we set the parameters minimum purity to 99 % and minimum number of records per leaf to 5. We experimented with several different settings of these parameters and found that they had a rather strong impact on the resulting accuracy and tree size. The above parameter settings were chosen in order to obtain a performance of our algorithm which is comparable to the performance of algorithms reported in the literature for these datasets.

5.2 Comparison with Standard Algorithms

In this section, we compare manual PBC with standard algorithms in order to demonstrate that interactive manual decision tree construction is competitive to automatic approaches. The state-of-the-art algorithms IND-CART and IND-C4 [14] as well as SLIQ were chosen as comparison partners. The performance results for these algorithms were taken from [11] but similar results were reported, e.g., in the documentation of the STATLOG project [13].

Table 1 presents the accuracy of the standard algorithms and manual PBC. We observe that the accuracy of manual PBC is similar to the accuracy obtained by the algorithms for the first three datasets having only numerical attributes. For the latter two datasets with categorical attributes, the accuracy of manual PBC is somewhat lower. This result shows that categorical attributes are more difficult to visualize and it also motivates the integration of algorithms to improve the accuracy.

Dataset	IND-CART	IND-C4	SLIQ	PBC
Satimage	85.3	85.2	86.3	83.5
Segment	94.9	95.9	94.6	94.8
Shuttle	99.9	99.9	99.9	99.9
Australian	85.3	84.4	84.9	82.7
DNA	92.2	92.5	92.1	89.2

Table 1. Accuracy of standard algorithms and manual PBC

Table 2 depicts the tree size of the standard algorithms and manual PBC. Note that the tree size is defined as the number of all leaf nodes while in [1] the number of all decision tree nodes was considered. For nearly all combinations of datasets and algorithms, the tree size obtained by manual PBC is signficantly smaller than the size of the trees automatically constructed by algorithms. The reduction of the tree size ranges up to a factor of 10 indicating that the manually constructed decision trees are generally more comprehensible.

Dataset	IND-CART	IND-C4	SLIQ	PBC
Satimage	90	563.0	133	33
Segment	52	102.0	16.2	21.5
Shuttle	27	57	27	8.9
Australian	5.2	85	10.6	9.3
DNA	35.0	171	45.0	18

Table 2. Tree size of standard algorithms and manual PBC

To conclude this comparison of manual PBC with standard algorithms, interactive manual decision tree construction obtains similar accuracy for numerical attributes and somewhat lower accuracy for categorical attributes. On the other hand, the manually constructed trees are in general significantly smaller than automatically created decision trees.

5.3 Comparison of Different Styles of Cooperation

The PBC system supports many different styles of cooperation between the user and the computer as discussed in section 4.3. To evaluate and compare the performance of different styles, we defined some prototypical styles of cooperation as follows:

- *automatic*: the decision tree is built completely automatic without any human interaction.
- *automatic-manual*: the top two levels (the root and its direct descendants) of the decision tree are constructed by the algorithm (without pruning), then the tree is completed manually by the user.
- *manual-automatic*: the top two levels (the root and its direct descendants) of the decision tree are constructed manually and the resulting leaves are finally expanded by the algorithm (with pruning).
- *manual*: the decision tree is constructed completely manually without any algorithmic support.

Table 3 presents the accuracy obtained for the different styles of cooperation. We find that manual-automatic in general yields the best accuracy and automatic yields the second best results. Both styles are significantly better w.r.t. accuracy than the other two styles. We conclude that algorithmic support is necessary to obtain an optimum accuracy, in particular for data with categorical attributes. On the other hand, the combined manual-automatic style outperforms the completely automatic decision tree construction for most of the datasets especially when incorporating domain knowledge of the user.

Dataset	Automatic	Automatic-Manual	Manual-Automatic	Manual
Satimage	86.4	84.1	86.8	83.5
Segment	95.5	95.0	96.3	94.8
Shuttle	99.5	99.6	99.7	99.9
Australian	84.9	80.9	86.8	82.7
DNA	93.8	89.2	93.3	89.2

Table 3. Accuracy of PBC for different styles of cooperation

Table 4 reports the tree size (number of leaf nodes) for the four different styles of cooperative decision tree construction. Automatic-manual has in general a low tree size, but it is outperformed by the automatic style on the Shuttle and Australian datasets and by the manual style on the Satimage dataset. Manual-automatic turns out to be the worst approach w.r.t. tree size and, in particular, typically creates larger trees than the manual approach.

Dataset	Automatic	Automatic-Manual	Manual-Automatic	Manual
Satimage	68	37	102	33
Segment	46.5	20.3	47.9	21.5
Shuttle	4	5	8	8.9
Australian	3.9	7.7	9.3	9.3
DNA	46	14	59	18

Table 4. Tree size of PBC for different styles of cooperation

Table 5 depicts the training time (in sec.) for the different styles of cooperation. We measured the total time for decision tree construction including the runtime of the algorithm and the time needed by the user for manual interaction. Automatic is the clear winner for all datasets except Shuttle. Interestingly, automatic-manual is the fastest style for the Shuttle data since the automatic part on the first two levels does not perform pruning and the manual part is considerably faster than the purely algorithmic construction with pruning. Manual-automatic is the second fastest style outperforming manual by a factor of up to 7. Thus, algorithmic support is very successful in speeding up interactive decision tree construction.

Dataset	Automatic	Automatic-Manual	Manual-Automatic	Manual
Satimage	232	857	269	1160
Segment	33	415	113	552
Shuttle	183	82	306	241
Australian	18	130	67	422
DNA	189	361	232	784

Table 5. Training time (in sec.) of PBC for different styles of cooperation

To conclude the comparison of different styles of cooperation, there is no clear overall winner because the results vary greatly for different datasets. Often, neither the completely automatic style nor the completely manual style yield the best results but they are outperformed by one of the combined styles (automatic-manual or manual-automatic). The manual-automatic style, for example, offers a good trade-off between accuracy, tree size and learning time for most of the datasets. We argue that a system for cooperative classification must support all these and further styles of cooperation of the user and the computer.

6. CONCLUSIONS

Current classification algorithms provide only very limited forms of guidance or "supervision" by the user. In this paper, we made the point for a cooperative approach to classification where both the user and the computer contribute what they do best. The goal is to exploit the human pattern recognition capabilities, to obtain a deeper understanding of the resulting decision tree and to constrain the search for knowledge using domain knowledge provided by the user.

We introduced novel techniques for visualizing training data and decision trees providing a lot of insights into the process of decision tree construction. Furthermore, we presented the integration of a state-of-the-art algorithm for decision tree construction supporting different styles of cooperation of the user and the computer. An extensive performance evaluation on several well-known benchmark datasets was conducted demonstrating that often neither the completely automatic style nor the completely manual style yield the best results but they are outperformed by one of the combined styles. We concluded that a system for cooperative classification should support many different styles of cooperation of the user and the computer.

Future work includes the following issues. In real applications, typically the cost of misclassification varies significantly for the different classes. It is an interesting question how to visualize different weights for the different class labels. Another issue in such applications is scalability w.r.t. training set size. Therefore, we want to investigate methods of visualizing very large training sets (having more records than the number of available pixels) on the screen with limited resolution and methods of efficiently integrating PBC into a database system (using secondary storage).

Acknowledgements

We want to thank all the people who contributed to the PBC system with their fruitful feedback, especially Christian Brendel, who implemented the algorithmic integration in our PBC system.

7. REFERENCES

[1] Ankerst M., Elsen C., Ester M., Kriegel H.-P.: “Visual Classification: An Interactive Approach to Decision Tree Construction”, *Proc. 5th Int. Conf. on Knowledge Discovery and Data Mining* (KDD'99), San Diego, CA, 1999, pp. 392-396.

[2] Ankerst M., Keim D. A., Kriegel H.-P.: “Circle Segments: A Technique for Visually Exploring Large Multidimensional Data Sets”, *Proc. Visualization ‘96*, Hot Topic Session, San Francisco, CA, 1996.

[3] http://www.angoss.com/

[4] Breiman L., Friedman J.H., Olshen R.A., Stone P.J.: “*Classification and Reggression Trees*”, Wadsworth Publishing, Belmont, CA, 1984.

[5] Berchtold S., Jagadish H.V., Ross K.A.: “Independence Diagrams: A Technique for Visual Data Mining”, *Proc. 4th Intl. Conf. on Knowledge Discovery and Data Mining* (KDD’98), New York City, 1998, pp. 139-143.

[6] Domingos P.: “The Role of Occam’s Razor in Knowledge Discovery”, *Data Mining and Knowledge Discovery*, an International Journal, Kluwer Academic Publishers, Vol.3, 1999, pp. 409-425.

[7] Coppersmith D., Hong S.J., Hosking J.R.M.: “Partitioning Nominal Attributes in Decision Trees”, *Data Mining and Knowledge Discovery*, an International Journal, Kluwer Academic Publishers, Vol.3, 1999, pp. 197-217.

[8] Han J., Lakshmanan L., Ng R.: “Constraint-Based, Multidimensional Data Mining”, *IEEE Computer*, Vol. 32, No. 8, 1999, pp. 46-50.

[9] Keim D. A.:”Visual Database Exploration Techniques”, *Proc. Tutorial Int. Conf. on Knowledge Discovery & Data Mining* (KDD’97), Newport Beach, CA, 1997.(http://www.informatik.uni-halle.de/~keim/PS/KDD97.pdf)

[10] Kohavi R., Sommerfield D.: “Targeting Business Users with Decision Table Classifiers”, *Proc. 4th Intl. Conf. on Knowledge Discovery and Data Mining* (KDD’98), New York City, 1998, pp. 249-253.

[11] Mehta M., Agrawal R., Rissanen J.: “SLIQ: A Fast Scalable Classifier for Data Mining”, *Proc. of the Int. Conf. on Extending Database Technology* (EDBT ‘96), Avignon, France, 1996.

[12] Mitchel T.M.: “*Machine Learning*”, McGraw Hill, 1997.

[13] Michie D., Spiegelhalter D.J., Taylor C.C.: “*Machine Learning, Neural and Statistical Classification*”, Ellis Horwood, 1994.
See also http://www.ncc.up.pt/liacc/ML/statlog/datasets.html.

[14] NASA Ames Research Center: “*Introduction to IND Version 2.1*”, 1992.

[15] Ridgeway G., Madigan D., Richardson T., O’Kane J.: “Interpretable Boosted Naive Bayes Classification”, *Proc. 4th Intl. Conf. on Knowledge Discovery and Data Mining* (KDD’98), New York City, 1998, pp. 101-106.

[16] http://www.spss.com/answertree/