Interactive and Manual Construction of Classification Trees

Stephen Pauwels STEPHEN.PAUWELS@STUDENT.UANTWERP.BE
University of Antwerp, Antwerpen, Belgium

Sandy Moens SANDY.MOENS@UANTWERP.BE
University of Antwerp, Antwerpen, Belgium

Bart Goethals BART.GOETHALS@UANTWERP.BE
University of Antwerp, Antwerpen, Belgium

Keywords: Interactive Classification, Manual Classification, Classification Trees

Abstract

We propose an approach in which a user can build tree based classification models under the assistance of a computer. We extend MIME, an existing framework for pattern discovery and exploration, with several mechanisms and visualisations for aiding a user to 1) construct new trees from scratch and 2) adapt existing trees, by showing good splitting points in the data, showing troublesome parts in the tree and automating computations of trees by standard techniques. We illustrate how our system can be used by non-classification experts to build interesting decision trees. Moreover, we provide a short experimental evaluation showing how trees generated using our tool compare to existing algorithms and their pruning techniques.

1. Introduction

Classification models are an important tool for decision making in databases. Their goal is to predict labels for new data instances based on information extracted from previously labeled instances. One typically used model is the classification tree, for which an example is shown in Figure 1. As for other data mining techniques, the construction of classification trees is 1) an iterative parameter refinement process and 2) requires both input from a computer and a user. On one hand, a computer can compute models over large amounts of data, while, on the other hand, a user has prior knowledge and semantic interpretation about the data. Blumenstock et al. showed that interactivity can be used to synergetically combine the two (Blumenstock et al., 2006). Moreover, through interactivity, the results become more understandable and meaningful (Blumenstock et al., 2006)(Liu & Salvendy, 2007).

The problem with most existing approaches, however, is that a user only has limited interaction capabilities (Ankerst et al., 1999)(Ware et al., 2001), in the sense that they only allow to grow a tree and not prune it down again. In our approach, focusing on tree based classifiers, the user is the driving force like in BaobabView (van den Elzen & van Wijk, 2011). To this end, she is given a pallet of tools that guide her during the construction of trees. At each point during the construction, the user is able to choose which splitting criterion she wants to use. She is thereby guided by several tools, e.g., the calculation of best splitting attribute, the overal quality of the tree, the distribution of the classes in a specific node in the tree, and more.

The contributions of our work are as follows:

- We augmented MIME (Goethals et al., 2011), an interactive tool for pattern discovery and exploration, with a new module for creating and adapting decision trees (Section 3). We used the same interaction techniques and information transferral methods that are widely used in MIME to aid the user during the construction process.

- We show in Section 4 how non-classification experts can use our tool for building classification trees. We show how visual cues can be acted upon by the user of the system.

Appearing in *Proceedings of BENELEARN 2014*.

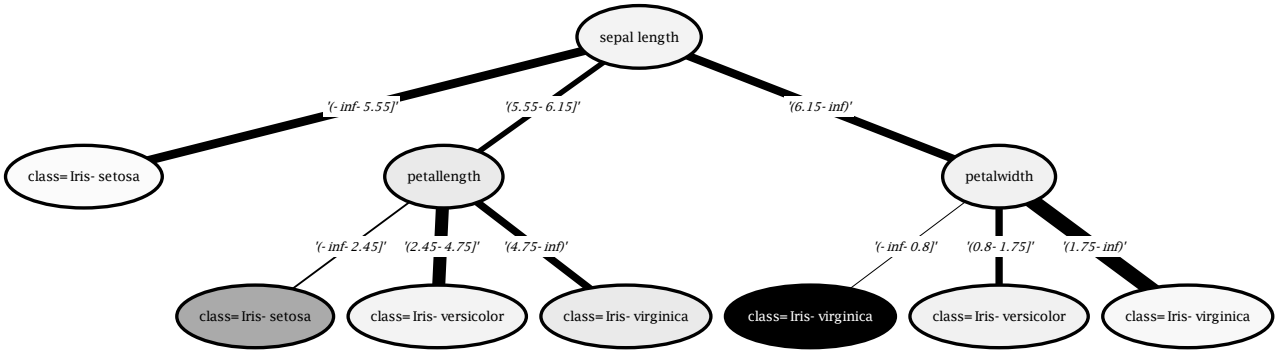

Figure 1. Classification tree for Iris-dataset represented in MIME

Support	Size	Itemset	Confidence	Accuracy
47	2	sepal length='(-inf-5.55]' → class=Iris-setosa	0.7966	0.9000
3	3	petallength='(-inf-2.45]' sepal length='(5.55-6.15]' → class=Iris-setosa	1.0000	0.6867
21	3	petallength='(2.45-4.75]' sepal length='(5.55-6.15]' → class=Iris-versicolor	1.0000	0.8067
10	3	petallength='(4.75-inf)' sepal length='(5.55-6.15]' → class=Iris-virginica	0.8333	0.7200
0	3	petalwidth='(-inf-0.8]' sepal length='(6.15-inf)' → class=Iris-virginica	NaN	0.6667
16	3	petalwidth='(0.8-1.75]' sepal length='(6.15-inf)' → class=Iris-versicolor	0.8889	0.7600
37	3	petalwidth='(1.75-inf)' sepal length='(6.15-inf)' → class=Iris-virginica	1.0000	0.9133

Figure 2. Classification tree represented as a set of rules in MIME

- We show in Section 5 that with our tool, a user can easily simplify existing computer generated classification models.

2. Iterative Interaction in MIME

We use MIME as knowledge representation and knowledge discovery framework (Goethals et al., 2011). Its main philosophy is that a user is the central part of the discovery process, as only she can decide what is truly interesting for her (Geng & Hamilton, 2006). However, as a user is not able to analyze large amounts of data on her own, she needs to be aided by multiple data mining algorithms and tools.

The knowledge presented in MIME is given as a collection of patterns. Originally, supported pattern types are association rules, e.g., {sepal length=‘(5.55-6.15]’, petallength=‘(-inf-2.45]’} → {class=Iris-setosa} and itemsets, e.g., {sepal length=‘(6.15-inf]’, petalwidth=‘(-inf-0.8]’}. These pattern types are easy to understand and use, even by non-expert users. A user can then easily modify the collection of patterns by adapting single patterns or even removing full patterns from the collection. In this work we augmented MIME with the interactive construction of classification trees and we added a tree model view, also node-link diagram (Liu & Salvendy, 2007), for this purpose. The constructed classification trees can be transformed easily to a set of association rules, where a left-hand side is a set of items and the right-hand side is a single class. An example classification tree and its corresponding collection of rules are given in Figures 1 and 2.

Patterns presented in MIME can be scored by different

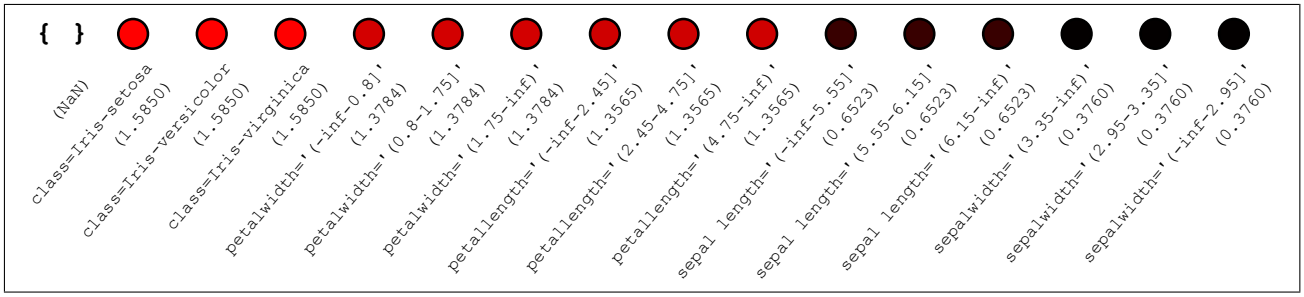

Figure 3. Source Dock containing all attribute-value combinations in the data, sorted by Gain Ratio

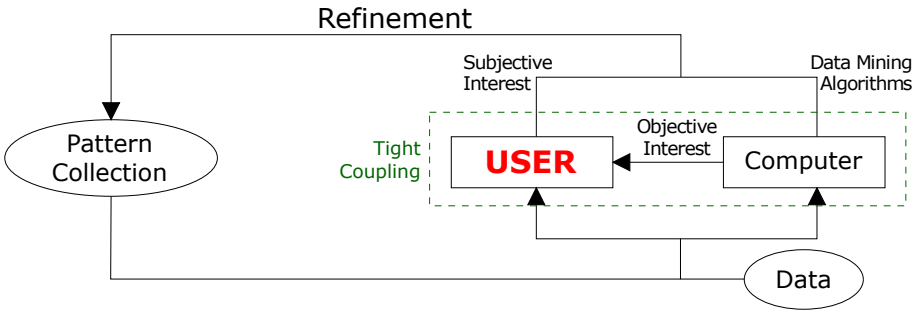

Figure 4. Iterative workflow of MIME

objective interestingness measures (Geng & Hamilton, 2006). These measures give an indication of how interesting a pattern potentially is for a user. For instance, a widely used objective measure for association rules is confidence and gives the probability of a rule being valid: if the rule {sepal length='(5.55-6.15]', petallength='(-inf-2.45]'} → {class=Iris-setosa} has 80% confidence, then in 4 out of 5 times, an iris flower with sepal length and petal length in the boundaries will in fact be of the Iris-setosa family. For itemsets, a well-known measure is the support. The latter indicates how many times the itemset effectively occurs in the raw data.

One of the prominent features in MIME is the *Best Pattern Extension* (Goethals et al., 2011). This is based on an active objective measure selected by a user, e.g., support or confidence. It scores the remaining items as extensions for the selected pattern and shows them in decreasing order of their score. As such, the top scoring items are always shown left-most in the Source Dock (see Figure 3).

The main workflow of MIME is shown in Figure 4. It shows that MIME is based on an iterative refinement of a collection of patterns by a user and an assisting computer. The iterative process originates from hidden knowledge in the data that a user wants to extract. For instance, consider the Iris dataset (Bache & Lichman, 2013), which provides the main characteristics of flowers in the Iris family. Then, what is interesting for a retailer selling flowers is not necessarily interesting for a flower biologist, i.e., the interesting knowledge is subjective to a specific user. Moreover, a person often does not know in advance what he or she is looking for, but is interested in patterns that are for instance actionable (Silberschatz & Tuzhilin, 1995)(Geng & Hamilton, 2006). This makes the discovery of interesting patterns into an iterative, exploratory process. To guide a user to interesting patterns, MIME provides on-the-fly computation of different objective measures (e.g., support and confidence), proposed patterns (e.g., *Best Pattern Extension*) and multiple data mining algorithms (e.g., Apriori (Agrawal & Srikant, 1994) and Eclat (Zaki et al., 1997)). A user can use these different tools to start building pattern collections. Essentially, this pattern collection represents a knowledge base for the system as well as for the user. The individual patterns and the collection as a whole can then be scored by objective measures and further refined. The refinement process stops whenever the user is satisfied with the discovered collection of patterns.

3. Interactive Construction of Classification Trees

Originally MIME is designed to explore data represented as itemsets and association rules. With this work we add support to build classification trees inter-

attribute name	attribute type	attribute domain	item
class	categorical	Iris-setosa, Iris-verisicolor, Iris-virginica	class=Iris-setosa class=Iris-versicolor class=Iris-virginica
petalwidth	numerical	(-inf – inf)	petalwidth='(-inf – 0.8]' petalwidth='(0.8 – 1.75]' petalwidth='(1.75 – inf)'

Figure 5. Example conversion of categorical and numerical attributes to items.

actively. We note that such trees are in fact rule-based classification models. One of the prominent ideas is that a user should not decide in advance which type of mining techniques or what type of patterns/models she is interested in. In fact, constructed trees are automatically represented as association rules in the background. This allows MIME to quickly show the association rules that describe the decision tree. An example tree for the Iris dataset is given in Figure 1. The corresponding rule set is shown in Figure 2: this is a mapping of tree branches to association rules, by traversing the tree top to bottom and left to right.

Classification trees are constructed from labeled datasets with numerical or categorical values. In MIME all attributes with their respective values are presented as attribute-value combinations. For categorical attributes the conversion is direct: an attribute with 3 values results in 3 items. Numerical attributes are discretized when loading the dataset. To this end, MIME uses implementations made available by WEKA (Hall et al., 2009), allowing a user to choose for instance the number of bins in which the values are grouped. Examples are given in Figure 5. In the remainder of this work, we use the terms item and attribute-value combination interchangeably.

The general idea of building a classification tree is splitting the data into smaller parts, such that, with high accuracy, a class can be assigned to a sub part of the data instances. New data instances having similar characteristics can then be assigned the same class label. Splitting data instances is achieved by taking a single attribute and one or more of its value combinations. For instance, in Figure 1 all data instances are present in the root. Next, a split is made using 'sepal length', such that three mutually exclusive parts are constructed. New instances having a sepal length between $-\infty$ and 5.55, are assigned class 'Iris-setosa' when a new classification is made. The remaining parts are further split using some of the remaining attributes. The final goal is to assign accurate labels to new data instances. Note that a node represents a conditional database, consisting of all instances containing all items on the branch to the node.

Several splitting criteria have been proposed in the literature, mostly based on the underlying class distributions in the conditional databases. The following measures are incorporated into our framework:

- **Classification Error:** Measures the error rate of a node, using the dominating class in the node when assigning class labels. The errors is weighted with respect to the size of the conditional database.
- **Class Purity:** Measures the total confidence of dominating classes in the conditional databases. This measure compares the share of the largest class to the share of all remaining classes combined.
- **Gini Index:** Gives the probability that a randomly chosen instance is classified incorrectly when assigned a random class label.
- **Gain Ratio:** Gives the differences in class entropy in the original data and the weighted class entropy in the conditional databases. The entropy is a measure for chaos in the data: the more classes and the more equal their share, the higher the entropy.

3.1. Visual Representation

Our tree representation consists of 3 different node types: an internal node, a leaf node and an empty node. An internal node (Figure 6(a)) represents an attribute/split in the data, and the branches are the different values for the attribute. A leaf node (Figure 6(b)) is a node corresponding to the class with a specific class value/label. This represents the class that is assigned to instances in the conditional database. An empty node (Figure 6(c)) is a node where no attribute or class value is assigned.

The source dock gives an overview of all available items (attribute-value combinations) (see Figure 3). Each one of these items can be used while building a tree.

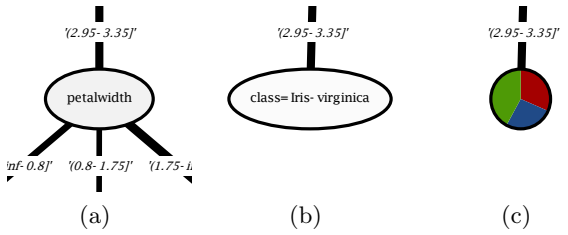

Figure 6. Different types of nodes

We note that measures for sorting items in the source dock, i.e., active measures, apply to the entire attribute. Therefore, all items dealing with the same attribute have the same value for the active measure.

Our tree representation in MIME inherently represents actionable knowledge to quickly point users to where and how the current tree can be enhanced. We implemented one visual cue assisting the growth of trees and two visual cues assisting the pruning of trees:

- **Pie Chart Representation:** Empty nodes need user attention and should be interacted with. We visualize suchs node as pie-charts reflecting the underlying class distribution (see Figure 6(c)). As such, a user is quickly able to see which class dominates the conditional database.
- **Size of the Conditional Database:** To indicate the relative number of instances that are part of a split, we set the thickness of edges proportional to its size (see Figure 1): thick edges indicates many instances, thin edges indicates fewer instances. The sizes can help in deciding which branches can be pruned when trying to reduce the tree complexity.
- **Degree of Overfitting:** After generating a tree, some branches may overfit the data, classifying only 1 or 2 instances. To determine if a node X is overgitting the data we use

  $$Overfit(X) = num\_leaves/num\_instances,$$

  with num_leaves the number of leaves that can be reached from the current node and $num\_instances$ the number of classified instances in the node. Using this formula, we obtain values between 0 and 1, such that 1 indicates extremely overfitting and low values indicates not overfitting. We coloured all nodes in a grayscale, indicating their degree of overfitting.

3.2. Interacting with Trees

The main interaction mechanism employed in MIME is dragging and dropping of single items. In this work we use the same machanism to grow and prune generated trees:

- **Inserting Internal Nodes:** An empty node is transformed in an internal node by dragging a single item from the source dock and dropping it on the node. Upon dropping the item, a user can choose to add only this specific attribute-value combination or to add also all remaining combinations for the attribute. The default method for adding an attribute to the tree is to add all combinations as children. That is to be sure the dataset is fully covered by the classification tree.
- **Inserting Leaf Nodes:** When dropping a class-value combination, an empty node is automatically transformed into a leaf node. A second possibility for quickly adding a class value is using the context-menu. Here, the different values of the class attribute are sorted in ascending order with respect to their distribution in the current node. It is also possible to 'Add the Best Value', which automatically adds the class value dominating the node.
- **Removing Nodes:** Existing nodes in the current tree can be removed by dragging the node away. The original node and its complete subtree are completely pruned and replaced by an empty node.

Next to drag-and-drop operations, a user can select a node from the tree, triggering a recomputation of the current active measure for the remaining items in the dataset. This allows to show the best attributes for different splitting measures. A user can then examine all, or a few, different measures and decide which attribute-value combinations to select as new criterion for splitting into conditional databases.

At last, MIME also supports tree generation using functionality from the WEKA library (Hall et al., 2009). These tree generators can be used to build either an entire tree or, alternatively, to complete an existing tree starting from a selected node. Trees that are generated are again completely modifiable by the previously specified drag-and-drop operations. As such, a user can quickly try out different generators for specific portions of the tree and decide which of the trees is best suited for adaptation. Even more, different parts of the final tree can be generated using different tree generation algorithms! As such, a user can combine the strengths of different classification methods during the exploration process. Another advantage of this idea, is that a user is able to make intelligent decisions early on in an automated decision tree construction based on her semantic background of the data. For

instance, suppose we have data about the age, sex, height, weight, weight increase and hair length of a large collection of people, and we want to predict if a person is pregnant based on these different features. Then an obvious decision that is prominent is whether or not a person is female. However, an automated classification tree builder, may not be able to discover this feature as a main splitting criteria, while all people know that only women can become pregnant.

4. Illustrative Scenario

Throughout the following two sections we use the Car Evaluation dataset from the UCI machine learning repository (Bache & Lichman, 2013) to give a workflow example and to conduct preliminary experiments. The dataset contains evaluations of cars based on technical and cost related specifications. The dataset consists of 6 attributes with 25 different attribute-value pairs in 1728 transactions.

In this section, we show a typical example of how to build classification trees using MIME. A user can build trees either from scratch or starting from a pre-generated tree, after which all parts of the tree can be freely adapted. We show how to build classification trees starting from a generated classification tree.

We first let MIME generate a starting decision tree. Using the interface we can choose between the Id3, J48, REPTree and RandomTree algorithms. For this example we run RandomTree, resulting in a tree containing 422 rules with maximum size of 5. Notice that this also implies that the tree consists of 422 leaves and has maximum depth of 5. The classification error for the entire tree is equal to 0, meaning that each instance in the test set is classified correctly. Even though the quality, in terms of classification error, is good, we want to reduce the size/complexity of the tree. More specifically we want to reduce the number of leaf nodes as more compact trees are easier to interpret and comprehend. Moreover having less rules in the classification tree results in less overfitting of the data.

Using visual cues from Section 3.1 we find a node at depth 3 (Figure 7) with class distributions: *class=unacc*: 26, *class=acc*: 10, *class=vgood*: 0 and *class=good*: 0. Although this node only classifies 36 instances it leads to 23 leaf nodes, implying many of the leaf nodes in fact classify just one instance. This results in overfitting of parts of the data. Removing this node by dragging it away, reveals the underlying class distributions in a pie chart. This shows almost 3 out of 4 intances are classified by *class=unacc*. Remember also that the complete subtree of the node is removed. Next we drop *class=unacc* on the empty node. The tree has been reduced from 23 rules to one single rule, reducing complexity and increasing interpretability. However, as a consequence, the classification error has increased to 0.0058. This is however still a low error rate. We repeat this process for all internal nodes of the tree to obtain a tree with lower complexity and with a reasonable classification error.

As mentioned previously, it is also possible to start building an empty tree without tree generation. The process is similar to the scenario we showed and also uses visual cues in combination with on-the-fly computed information to guide the user in building good classification trees.

5. Experimental Evaluation

For our experiment we generate a tree using the tree building algorithms Id3, J48, REPTree and RandomTree. We prune the resulting tree manually and compare them to the original tree with respect to size and classification error. We use the same dataset (car evaluations) as in the previous section. The resulting trees are shown in Table 1 and 2.

We build 4 trees using the algorithms incorporated in MIME using WEKA. For the Id3 tree we perform multiple iterations over the entire tree. First, we remove all nodes that are coloured black, since these have a large overfitting factor. For every removed node we add a leaf-node by choosing the most occurring class value in the node. In the second iteration, we remove nodes for which all its children are leaves with the same class value. This can obviously be simplified by pulling the class up one level. In the third iteration, we look at the class distribution of all nodes that only have leaves as children. If one value occurs more than twice the amount of all the others, we replace the node by that value. In the last iteration, we remove all nodes where one class value occurs at least as much as all the others together, i.e., in more than 50% of the instances.

For J48 and REPTree we constructed trees by enabling inherent pruning mechanisms. Therefore, no completely black nodes are produced by these methods. We immediately use the generated trees as a comparison for our manually pruned trees.

For the tree generated by the RandomTree algorithm we use a similar iterative pruning procedure as for Id3. However, we start in the root of the tree and descend in a depth-first way. When encountering a black node we remove it and replace it with a class value. Next, we also remove redundant leaves.

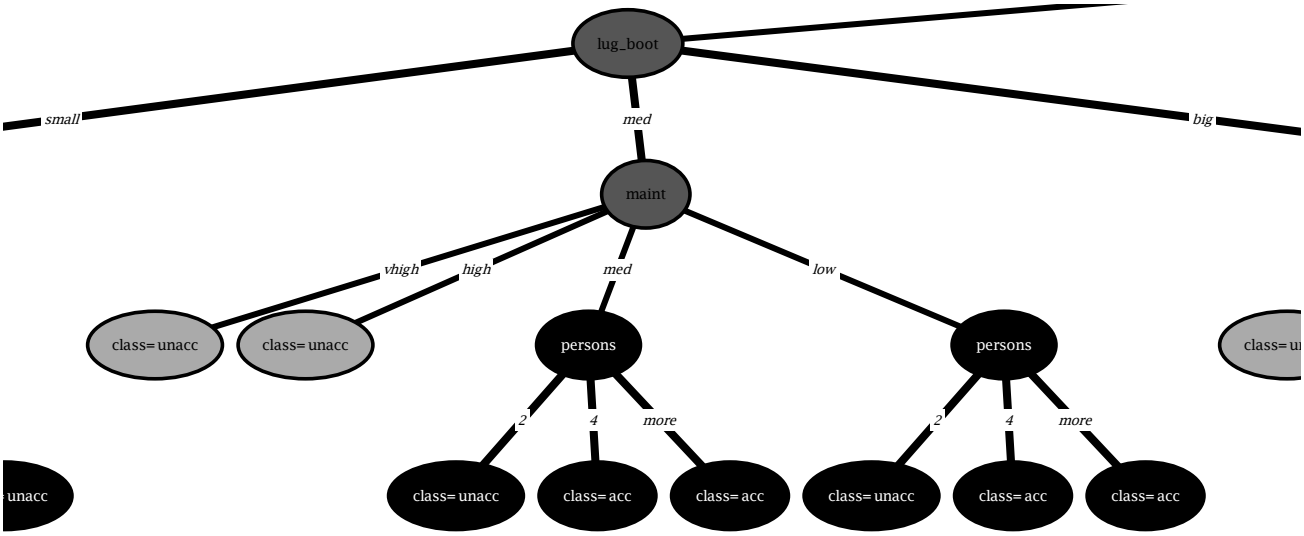

Figure 7. Original inner node

Id3		J48	
Classif. Error	#Leaves	Classif. Error	#Leaves
0	296	0.0370	131
REPTree		RandomTree	
Classif. Error	#Leaves	Classif. Error	#Leaves
0.0596	106	0	422

Table 1. Measures of tree generators

Id3 (after 1 iteration)		Id3 (after 2 iterations)	
Classif. Error	#Leaves	Classif. Error	#Leaves
0.0359	159	0.0359	129
Id3 (after 3 iterations)		Id3 (after 4 iterations)	
Classif. Error	#Leaves	Classif. Error	#Leaves
0.0544	117	0.1782	18
Random (1 iteration)		Random (2 iterations)	
Classif. Error	#Leaves	Classif. Error	#Leaves
0.0851	150	0.0851	141

Table 2. Measures of adapted trees

Comparing Table 1 to Table 2 we see that Id3 contains less leaves than J48 after 2 iterations and also a smaller error. When comparing to REPTree we see that we have 23 leaves more, but also an error which is 0.0237 smaller. The result after 3 iterations over Id3 is more similar to the result of the REPTree algorithm. After applying the fourth iteration on Id3 we get a larger difference between the fully automated algorithms and our own manual pruning method. Our tree now only has 18 leaves compared to 296 we got from the original Id3 algorithm. On the downside the classification error has increased to 0.1782, which is much larger than the original tree or after the other iterations. In some cases such an error is not a very big issue. First of all, the classification error only takes the training data into account. Secondly, an expert user in the data domain can sometimes detect overfitting and faulty instances in the dataset. Moreover, some users know which decisions are most crucial when classifying new instances. This can explain why a classification tree can get a high error for a given training set. While in practice it may prove to be a decent classification model.

The experiment for the RandomTree gives less good results: we obtain a tree with a higher number of leaves and a larger classification error. The approach of always removing the nodes starting from the root thus gives worse results opposed to starting at the leaves and going up the tree. This is because nodes which indicate that they are overfitting can be non-overfitting after the removal of some of its children.

We have shown that we have some interesting tools in MIME which give a user the possibility to prune a tree by herself, without an algorithm that intervenes.

6. Related Work

Constructing classification trees is often done by a computer alone. Some approaches try to involve the user more into this construction process. For the related work we only focus on the latter.

Ankerst et al.(Ankerst et al., 1999) developed a framework where a user interactively builds classification trees by creating split intervals. They allow backtrack-

ing by removing branches that a user dislikes. They however do not have the possibility of using existing techniques for building a classification tree, or to complete non-finished branches. Ware et al. (Ware et al., 2001) adopted a similar approach where trees are build by drawing splits in the true data. They also only allow for user constructed trees. These approaches are useful to let the user work very closely to the data and to let her understand the classifier better.

BaobabView is a framework created by Van de Elzen and Van Wijk (van den Elzen & van Wijk, 2011) that is very similar to our approach. They use similar cues (Section 3.1) and split measure computations, to assist users in growing and pruning trees. They also allow to complete only specific parts of an incomplete tree. A difference to our method is that they employ a confusion matrix to indicate the classification error, while we employ a grayscale immediately in the tree.

Han and Cerone (Han & Cercone, 2001) introduced parallel segments, a novel data visualization technique for data instances. They employ a second view that represents the current decision tree. For the construction they employ a 5 step interaction model with the data visualization. The different steps can be translated to drag-and-drop operation in our framework.

Some other data mining frameworks also allow the construction of classification trees, but in a much more restricted sense. In WEKA (Hall et al., 2009) a user has the ability to select multiple decision tree generators and to select which data attributes should be used for this purpose. Here, a user does not have the ability to adopt the resulting classifier to incorporate background knowledge. The only possible interaction is the adaptation of input parameters.

KNIME (Berthold et al., 2007) lets a user build trees by creating a complete workflow choosing which data attributes to load, what classification tree learner to use, how many folds to use, etc. The interaction is limited in the sense that the results can not be adapted directly, rather the workflow has to be altered for this purpose. The tool is therefore much less exploratory and interactive compared to ours.

7. Conclusion and Future Work

Fully automated tree generators often generate very large, complex tree structures that are not easy to interpret by humans. Therefore, a few interactive classification tools have been developed. However, the problem with most of these tools is that they do not give the user the full degree of freedom that would often be useful. Most methods only let the user decide which pieces of the data she wants to split and which need to be together. Other methods only foresee the ability to set specific, often difficult to set, parameters and then let an algorithm run to the end. They do not allow to make adaptations to the resulting tree easily.

In this work we adapted MIME, which already supports the exploration of itemsets and association rules, to provide a fully integrated proces of building a classification tree. We added the ability to build such trees fully manually by dragging attribute-value combinations, corresponding to splits in the data, onto empty nodes in a tree. To this end, a user can make use of splitting measures to sort remaining attributes in the data from best to worst splitting criteria. Alternatively, she can use different existing tree generators and completely adapt the resulting tree afterwards. It is also possible to complete a tree starting from a specific empty node using these generators.

Our experiments show that the tree generators produce good trees with few errors, but at the cost of complexity. By adapting generated trees we can reduce the size of a tree intelligently, based on our background knowledge. Although trees become less accurate, they also become more readable and understandable by users. More methods for simplifying generated classification trees are to come.

In the future, we want to add on-the-fly discretization of numerical attributes. We want to give the user the possibility to change splits of an attribute within the visualization for different branches, as opposed to having static splits employed by all branches. A new means for simplifying generated trees is to add the possibility to merge two or more splits for one attribute into one, which then can be merged to binary splits. Since MIME internally converts the tree to association rules we also want to add useful and meaningful ways to convert any collection rules to a (collection of) decision trees.

One main goal in MIME wrt this work is to keep all interactions between rules and trees as simple as possible, and that a user does not have to choose a specific pattern type. Ratherm she can instantly switch from building a collection of rules to building a classifier.

Acknowledgements

The authors would like to thank, in alphabetical order, the co-developers of the original software package: Tom De Schepper, Yannick Jadoul, Stefaan Kenis and Timothy Verstraeten.

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules in large databases. *Proceedings of the 20th International Conference on Very Large Data Bases* (pp. 487–499). Morgan Kaufmann Publishers Inc.

Ankerst, M., Elsen, C., Ester, M., & Kriegel, H.-P. (1999). Visual classification: An interactive approach to decision tree construction. *Proc. ACM SIGKDD* (pp. 392–396).

Bache, K., & Lichman, M. (2013). UCI machine learning repository.

Berthold, M. R., Cebron, N., Dill, F., Gabriel, T. R., Kötter, T., Meinl, T., Ohl, P., Sieb, C., Thiel, K., & Wiswedel, B. (2007). KNIME: The Konstanz Information Miner. *Studies in Classification, Data Analysis, and Knowledge Organization (GfKL 2007)*. Springer.

Blumenstock, A., Kempe, S., Lanquillon, C., Hipp, J., & Wirth, R. (2006). Interactivity closes the gap, lessons learned in an automotive industry application. *Proceedings of the 2006 workshop on Data Mining in Business Applications*. Philadelphia, Pennsylvania, USA: ACM.

Geng, L., & Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Comput. Surv.*

Goethals, B., Moens, S., & Vreeken, J. (2011). Mime: A framework for interactive visual pattern mining. *Proc. ACM SIGKDD* (pp. 757–760). ACM.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The weka data mining software: An update. *SIGKDD Explor. Newsl.*, *11*, 10–18.

Han, J., & Cercone, N. (2001). Interactive construction of decision trees. In *Advances in knowledge discovery and data mining*, Lecture Notes in Computer Science, 575–580. Springer Berlin Heidelberg.

Liu, Y., & Salvendy, G. (2007). Design and evaluation of visualization support to facilitate decision trees classification. *International Journal of Human-Computer Studies*, 95–110.

Silberschatz, A., & Tuzhilin, A. (1995). On subjective measures of interestingness in knowledge discovery. *KDD* (pp. 275–281).

van den Elzen, S., & van Wijk, J. J. (2011). Baobabview: Interactive construction and analysis of decision trees. *Visual Analytics Science and Technology (VAST), 2011 IEEE Conference on* (pp. 151–160).

Ware, M., Frank, E., Holmes, G., Hall, M., & Witten, I. H. (2001). Interactive machine learning: Letting users build classifiers. *International Journal of Human-Computer Studies*, *55*, 281 – 292.

Zaki, M. J., Parthasarathy, S., Ogihara, M., & Li, W. (1997). New algorithms for fast discovery of association rules. *In 3rd Intl. Conf. on Knowledge Discovery and Data Mining* (pp. 283–286). AAAI Press.