Classification: Grafted Decision Trees

Emil Brissman & Kajsa Eriksson

Linköping University Campus Norrköping

2011-12-01

1 Introduction

Decision trees play a significant role in data mining when it comes to classification problems. Alongside decision rules it is the method for deducing classification models to apply to unclassified data. Decision trees for classification have the purpose to classify instances correctly based on attribute values. The wish is always that the classification performed by the tree is 100 % correct with no misclassifications. However this is nearly never the case since to be completely correct the tree would have to be trained with all possible instances. It is not likely to have access to all that data and not the purpose either, for what use would one have for the classification tree then, if all instances are already used?

There are several techniques and methods available to make the decision tree's performance as high as possible. This paper will discuss one of them, called grafting, and also mention some alternatives and relevant co-techniques such as boosting and pruning.

Grafting is an algorithm for adding nodes to the tree as a post-process. Its purpose is to increase the probability of rightly classifying instances that fall outside the areas covered by the training data. Four algorithms implementing this idea in a slightly different way will be covered in this paper. They are all implemented as post-processors of the well known C4.5 classification algorithm introduced by Quinland (1993) and are therefore named C4.5X, C4.5+, C4.5++ and C4.5A.

Boosting and specifically AdaBoost are an alternative method that in the process comes up with a classification prediction for the regions without training data. Webb (1999) compares this method with grafting and concludes that grafting, just like boosting, reduce the prediction error of the tree. However grafting results in a far less complex tree than boosting since grafting only creates a single tree, in contrast to the several trees that the boosting algorithm in Tan et al. (2006) produce.

This paper will describe the idea behind grafting and the specific algorithms mentioned above. Their usage and ideas alongside benefits and drawbacks will be presented in chapter two. Chapter three offers examples of applications and references to other work regarding grafting. Finally in chapter four some conclusions about grafting decision trees will be presented.

2 Techniques to address the problem

A wide spread and accepted theorem is Occam's Razor (Webb 1996) which states that if two hypothesis explains a problem, then the simplest one of them is to be preferred. Translated into decision tree context this means that out of two trees that solve a classification problem the least complex tree is preferred. The basic idea of tree grafting comes from the wish to discard the "simplest is best" method for selecting a good tree and instead focus on the fact that objects that are similar in some sense have the highest probability of belonging to the same class. In other words it overlooks the possibility of yielding more complex trees if the result, in the end, is a better classification model.

There are two properties to watch out for when a classification model is to be built. They are complementary and a balance between them is of uttermost importance. The first one is called bias which is the term for when the leaves of a tree represents such a large region that it is too general to encompass just one class. The other one is called variance and represents the inverse case, when the region of the leaf is so small that a class cannot fit within and a big portion is left outside (Webb 1996). Too much of either property obviously results in an increase in error rate which is why it is so important to keep a good balance between them. Applying grafting techniques to a decision tree is a way to try to keep the bias low since grafting compute new branches that split the original leaf regions into smaller ones. However the trick is to not split the regions too much so that the variance increases significantly.

2.1 C4.5X

The C4.5X algorithm was the pioneer grafting algorithm developed to, as stated above, prove that a more complex decision tree should not always be discarded. After extensive testing it managed to somewhat prove this because of its success in reducing prediction errors.

The algorithm tries to find areas without any training data that the C4.5 algorithm has given a class that might not be the best one from a similarity point of view. When such an area is found it adds branches to the tree to split that area's class space into smaller partitions. It will not do any branching or splitting that worsens the performance in classifying the existing training data. The algorithm works only for continuous attributes.

The algorithm works as described in Webb (1996):

1. For each leaf *l* in the tree each attribute *a* is considered.
2. For each *a* all possible thresholds below and above the region represented in leaf *l* are explored.
3. A maximum and minimum value are computed such that only instances with *min*<*value*<*max* for attribute *a* are considered. In other words only instances with a value for *a* that can reach leaf *l* is considered.
4. For each value observed that fall within the range of leaf *l* but outside the range of the actual instances' values in *l,* a support is calculated for reclassifying the region above/below that threshold.
5. The support is calculated using binary Laplacian accuracy estimate *(P +1 / T+2)* where *P* is the number of instances that belong to the certain class and *T* is the number of instances at the ancestor level for which *min*<*value*<*threshold.* And then the same is made for *threshold*<*value*<*max*.
6. The single threshold value for the attribute *a* that has the highest estimate is matched against the evidence for the original classification of the leaf region. If it is higher than a new branch is added that creates a new leaf with a region that was previously a part of the region of *l*, but is now separated and can be classified as another more likely class even though no training data is present there.

2.2 C4.5+

C4.5+ is a first improvement of the original C4.5X algorithm. The big difference between them is that C4.5X only cuts and adds a new leaf for the maximum support threshold. C4.5+ implements the possibility to add multiple new leaves for each original leaf.

An example to illustrate the algorithm is as follows. An instance space has been divided by the C4.5 algorithm into different classifications with respect to two attributes A and B, as in figure 1. The blue area corresponds to a leaf and there is an uncertainty to which class the question mark seen in the figure belong. The area will be considered to be cut in several pieces by the C4.5+ grafting algorithm to produce less prediction errors. The evidence in support for the majority class and a cut at B=3 is calculated with respect to the ancestor node (B<=5) and is equal to (9+1)/(9+2) = 0.909. This is the highest value and so B=3 is the best cut between 1<=B<=5.

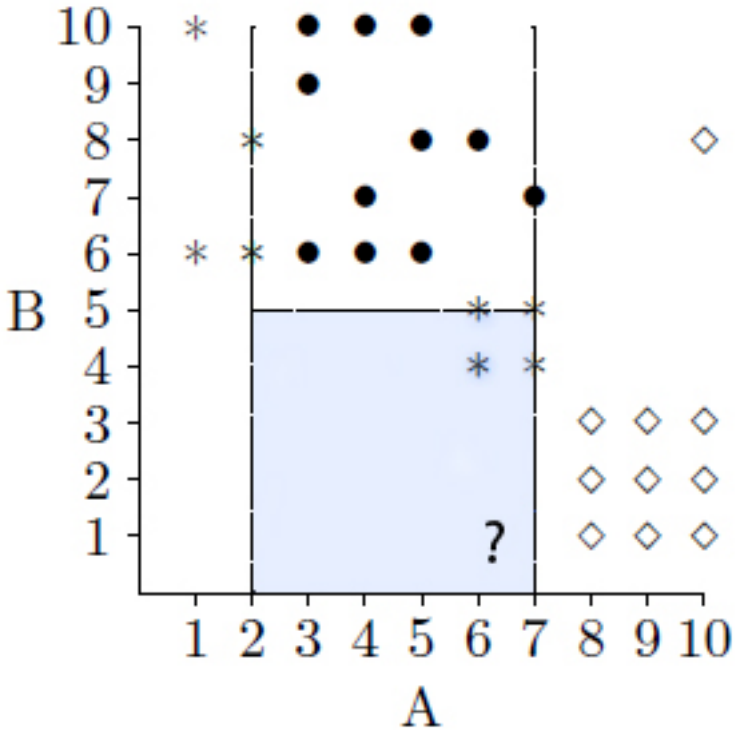

Figure 1. Instance space before grafting: Objects are classified as * in the blue region. The black dot in the blue region has been misclassified.

This value is stored in a list together with the best cuts for each attribute. The list is later used to create new branches and leaves between the leaf and its parents. Figure 2 illustrates the result after grafting the tree from in figure 1. Three new leaves a, b and c in the figure now belongs to the classes ◊, * and ◊ since the support for those classes were the highest in those new regions. Now it is possible to classify the question mark with a small prediction error (Webb 1997).

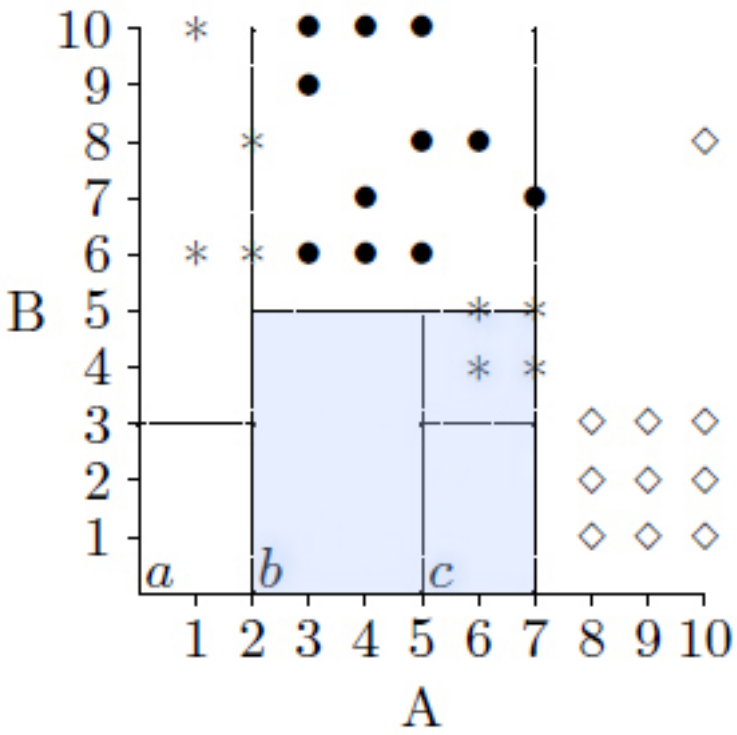

Figure 2. Instance space after grafting: Three new spaces have been added with new classifications.

C4.5+ allows reclassification of leaf regions that contain training examples that are misclassified by the original leaf, not just empty regions. Another difference from C4.5X is that C4.5+ allows the use of training examples from an ancestor node when the corresponding leaf has no training examples.

2.3 C4.5++

Webb (1999) further develops the C4.5+ algorithm into C4.5++ with the ability to apply it to discreet valued attributes. This new possibility decreases the computation time and also the average prediction error. In Webb (1999) a bias/variance analysis concluded that older versions of the algorithm primarily reduced error variance. The new algorithm, C4.5++, also considers the reduction of bias, or at least makes an effort in finding a balance between the two measures.

2.4 C4.5A

Webb (1999) also introduces the C4.5A algorithm called "Grafting from the all-tests-but-one partition" which is a more efficient way to evaluate the support of evidence. Webb (1999) explains the All-Test-But-One-Partition (ATBOP) region of a leaf as the region formed by removing all the decision surfaces that enclose the leaf. With the use of ATBOP the computational requirements are reduced because only the set of training data from the ATBOP region is considered for each leaf, compared to the entire training data set for every ancestor node in previous algorithms.

2.5 AdaBoost

The AdaBoost method in Tan et al. (2006) is a method that penalizes models that have poor accuracy for previous boosting rounds. It is a boosting algorithm which creates a committee of classifiers and adaptively changes the distribution of training examples to easily classify harder test examples. This method has shown to be successful at reducing decision tree error, as described by Webb (1999). The difference from grafting is that the grafting algorithm by Webb (1999) creates a far less complex tree compared to the tree resulting from merging the committee of classifiers by AdaBoost.

3 Applications

Grafting can be used as a post-process whenever a decision tree has been created. It will increase the complexity of the tree, but most significantly reduce the prediction error of the tree. Applications of the previously mentioned techniques could therefore preferably be present anywhere decision trees are and particularly to solve classification problems more efficient.

Previous chapter gave a brief example of applying grafting to a simple classification problem and Webb (1997) presents it in more detail. Webb (1999) offers an example on the latest technique, C4.5A. The C4.5A algorithm is also implemented in the open source data mining software Weka (University of Waikato) and is there called J48graft[1].

4 Conclusions

Grafting is a post-process that can be applied to decision trees. Its primary purpose is to decrease prediction error by reclassifying regions of the instance space where no training data exists or where there is only misclassified data. It finds the best suited cuts of existing leaf regions and branches out to create new leaves with other classifications than the original. In this process the tree naturally becomes more complex. But only branching that does not introduce any classification errors in data already rightly classified is considered. This ensures that the new tree reduce errors instead of introduce them.

Since most of the techniques are built upon each other, the difference is in terms of new releases and improvements of previous work. The differences between C4.5X and C4.5+ were that tests were made to prove that a new cut was good and did not decrease the prediction accuracy. Also the ability for multiple cuts, new leaves reclassifying regions and allowing global information to be used when local information does not exist, differs between the two algorithms.

Further development concludes differences such as an even more general algorithm working on discreet values and improvements on the computation time using the ATBOP region. Webb (1999) explains similarities between AdaBoost and grafting and concludes that grafting benefits from the theory of boosting and its decision committee. Even though the grafting algorithm makes the tree more complex, the complexity of the AdaBoost decision tree is higher.

[1] For documentation see: http://fiji.sc/javadoc/weka/classifiers/trees/J48graft.html

A process called pruning could be thought of as the opposite to grafting since pruning aims at reducing the complexity of the decision tree and still have good prediction accuracy while grafting, as we have seen does this by adding complexity to the tree. Surprisingly enough Webb (1997) concludes that pruning and grafting despite being opposites, or because of that, works well in parallel. The grafting algorithm take into account instances outside the analyzed leaf (global information) while pruning only looks at instances within the analyzed leaf (local information). In this way they seem to complement each other and using both on a decision tree yields a lower prediction error that using them separately in most cases (Webb 1997).

5 Bibliography

Kumar, V., Steinbach, M. & Tan, P.-N. (2006). *Introduction to Data Mining*. Pearson College Div.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Los Altos: Morgan Kaufmann.

University of Waikato. *Weka 3: Data Mining Software in Java*.
http://www.cs.waikato.ac.nz/ml/weka/index.html [2011-12-01]

Webb, G.I. (1996). Further Experimental Evidence against the Utility of Occam's Razor. *Journal of Artificial Intelligence Research,* vol. 4, pp. 397-417.

Webb, G.I (1997). Decision Tree Grafting. *Learning,* IJCAI'97 Proceedings of the Fifteenth international joint conference on Artificial intelligence, vol. 2, pp. 846-85.

Webb, G.I. (1999). Decision Tree Grafting From the All-Test-But-One Partition. *Machine Learning*, IJCAI '99 Proceedings of the Sixteenth international joint conference on Artificial intelligence, vol. 2, pp. 702-707.