DECISION TREE INDUCTION FOR DYNAMIC, HIGH-DIMENSIONAL DATA USING P-TREES[1]

Anne Denton and William Perrizo
Department of Computer Science,
North Dakota State University
Fargo, ND 58105-5164, USA
{Anne.Denton, William.Perrizo}@ndsu.nodak.edu

Abstract

Decision Tree Induction is a powerful classification tool that is much used in practice and works well for static data with dozens of attributes. We adapt the decision tree concept to a setting where data changes rapidly and hundreds or thousands of attributes may be relevant. Decision tree branches are evaluated as needed, based on the most recent data, focusing entirely on the data that needs to be classified. Our algorithm is based on the P-tree data structure that allows fast evaluation of counts of data points, and results in scaling that is better than linear in the data set size.

Keywords: Information Systems, Data Mining, Classification, Decision Tree Induction, P-Trees.

1 INTRODUCTION

A typical classification task consists of predicting a discrete property, such as whether a patient is sick, whether an e-mail message is spam, or whether a protein has a particular function. Traditionally a handful of attributes were used for the prediction. When evaluating whether a patient has a particular disease a doctor may do a dozen tests and use the results for his conclusion. Decision tree algorithms, such as C4.5 [1] and its successors, have proven extremely useful in this setting. Modern data mining problems are, however, often different in nature. When predicting the function of a gene, researchers can draw on a wealth of information that has been collected in biological databases and may contain thousands of relevant properties. Traditional decision tree algorithms are not suitable in this setting because, in any one prediction, they only use the information of a small number of attributes. Including more attributes would result in decision tree branches that are represented by few or no training points and cannot provide statistically significant information, a problem known as the curse of dimensionality [2]. Our algorithm builds decision tree branches around the sample that is to be classified (lazy classification). That means that the data set is not broken up unnecessarily by attribute values that are irrelevant to the sample in question. We consistently follow the ideal of an entirely instance-driven classification even when continuous attributes are involved. Whereas traditional decision tree algorithms, including lazy decision tree induction [3], discretize the data based on the data set available at training time, our algorithm uses intervals that are based on the test sample.

In contrast to other instance-based, such as Parzen window [4], kernel Podium [5], and k-nearest neighbor classification, our algorithm has a powerful attribute selection mechanism similar to decision tree algorithms. Kernel- and window-based techniques do not normally weight attributes according to their importance. For categorical attributes, this limitation is particularly problematic because it only allows two possible distances for each attribute, such as distance 0 if the values of categorical attributes match and 1 otherwise. Solutions that have been suggested include weighting attribute dimensions according to their information gain [6]; optimizing attribute weighting using genetic algorithms [7]; selecting important attributes in a wrapper approach [8]; and, in a more general kernel formulation, boosting as applied to heterogeneous kernels [9]. All of these approaches increase the algorithmic complexity and are therefore unsuitable to the typically large data set sizes encountered in data mining.

A further problem with traditional decision tree algorithms is that the classifier, i.e., the decision tree, has to be constructed every time the data changes. That is unacceptable in settings were new data arrives rapidly, such as predictions in computer networks. We use the P-tree data structure [10] to compute counts on the current data in a way that typically scales less then linear with data set size. The favorable scaling is achieved by a bit-column-wise storage organization in which sections of columns that are purely composed of 0 or 1 values at every level are eliminated from the calculation of counts. This leads to a best-case complexity that is logarithmic in the number of data points.

Section 2 discusses the algorithm, Section 3 presents our experimental setup and results, and Section 4 concludes the paper.

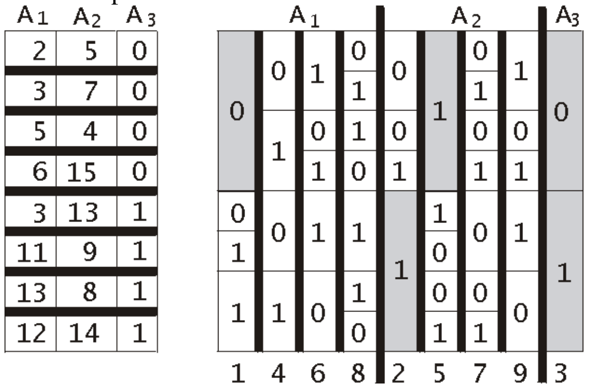
[1] Patents are pending on the P-tree technology. This work is partially supported by GSA Grant ACT#: K96130308, NSF Grant OSR-9553368 and DARPA Grant DAAH04-96-1-0329.

2 ALGORITHM

Our algorithm is loosely based on decision tree induction [1] in selecting attributes successively based on their relevance to the classification task. Data points that match in all selected attributes are considered relevant to the prediction task and the class label of the sample of interest is determined from the plurality of votes among those points. Attribute selection is based on optimizing the gain of information as defined by Shannon [12]. In contrast to conventional decision tree induction [1] tree branches are constructed as needed according to the sample that is to be classified, similar to lazy decision tree induction [3]. The main differences compared with [3] lie in our treatment of continuous attributes using a window function and the efficient count calculation using P-trees.

2.1 P-trees

The P-tree data structure was originally developed for spatial data [10] but has been successfully applied in many contexts [11,13] and is in describe in detail in those publications. P-Trees store bit-columns of the data in sequence to allow compression as well as the fast evaluation of counts of records that satisfy a particular condition. A tree-based structure replaces subtrees that consist entirely of 0 values by a higher level "pure 0" node, and subtrees that consist entirely of 1 values by higher level "pure 1" nodes. The number of records that satisfy a particular condition is now evaluated by a bit-wise AND on the compressed bit-sequences. Figure 1 illustrates the storage of a table with 2 integer and one Boolean attribute. The number of records with A_1 = 12 (i.e. the bit sequence 1100) is evaluated as a bit-wise AND of the two P-Trees corresponding to the higher order bits of A_1 and the complements of the two P-Trees corresponding to the lower order bits. This AND operation can be done very efficiently for the first half of the data set, since the single high-level 0-bit already indicates that the condition is not satisfied for any of the records. This is the basis for a scaling better than O(N) for such operations.



A_1	A_2	A_3
2	5	0
3	7	0
5	4	0
6	15	0
3	13	1
11	9	1
13	8	1
12	14	1

Figure 1: Storage of tables as hierarchically compressed bit columns

The efficiency of P-tree operations relies on the compression of the bit sequences, and thereby on the ordering of rows. For data that shows inherent continuity, such as spatial or multimedia data, such an ordering can be easily constructed. If data shows no natural continuity it may be beneficial to sort it. We sort according to all highest order bits first. Figure 1 indicates at the bottom the sequence in which bits are used for sorting.

2.2 HOBbit Distance

The nature of a P-tree-based data representation with its bit-column structure has a strong impact on the kinds of algorithms that will be efficient. Determining the number of points in an interval that is defined by the HOBbit distance [11] requires only one AND operation and is therefore particularly efficient:

$$d_{HOBbit}(a_s,a_t) = \begin{cases} 0 & \text{for } a_s = a_t \\ \max\limits_{j=0}^{\infty}\left(j+1 \,\middle|\, \left\lfloor \frac{a_s}{2^j} \right\rfloor \neq \left\lfloor \frac{a_t}{2^j} \right\rfloor\right) & \text{for } a_s \neq a_t \end{cases}$$

where a_s and a_t are attribute values, and $\lfloor\ \rfloor$ denotes the floor function. The HOBbit distance can also be understood as the number of bits by which two values have to be right-shifted to make them equal.

2.3 Pruning

It is well known that decision trees have to be pruned to work successfully [1]. Information gain alone is, therefore, not a sufficient criterion to decide which attributes to use for classification. We use statistical significance as a stopping criterion, similar to decision tree algorithms that prune during tree construction. In our algorithm, significance is calculated on a different subset of the data than information gain to get a statistically sound estimate. The training set is split into two parts, with two-thirds of the data being used to determine information gain and one-third to test significance through Fisher's exact test [14]. An attribute is considered relevant only if it leads to a split that is significant, e.g., at the 1% level. The full set is then taken to determine the predicted class label through plurality vote.

2.4. Pursuing Multiple Paths

The number of attributes that can be considered in a decision-tree-like setting, while maintaining a particular level of significance, is limited due to the "curse of dimensionality" [2]. Although our algorithm suffers less from this problem then conventional decision tree techniques due to its focus on the sample attributes, we still observe a benefit from combining multiple classifiers to get better statistics. A similar approach is taken by bagging algorithms [15]. We use a very simple alternative in which several branches are pursued, each starting with a different attribute. The attributes with the highest information gain are picked as starting attributes, and branches are constructed in the standard way from thereon. The votes of all branches are combined into one final vote. This modification leads to a particularly high improvement for data sets with many attributes because it gives some attributes a vote that would not otherwise have one.

3 IMPLEMENTATION AND RESULTS

We implemented all algorithms in Java and evaluated them on 7 data sets. Data sets were selected to have at least 3000 data points and a binary class label. Two-thirds of the data were taken as a training set and one-third as a test set. Due to the consistently large size of data sets, cross-validation was considered unnecessary. All experiments were done using the same parameter values for all data sets.

3.1 Data Sets

Five of the data sets were obtained from the UCI machine learning library [16] where full documentation on the data sets is available. These data sets include the following:

- adult data set: Census data are used to predict whether income is greater than $50,000.
- spam data set: Word and letter frequencies are used to classify e-mail as spam.
- sick-euthyroid data set: Medical data are used to predict sickness from thyroid disease.
- kr-vs.-kp (king-rook-vs.-king-pawn) data set: Configurations on a chess board are used to predict if "white can win."
- mushroom data set: Physical characteristics are used to classify mushrooms as edible or poisonous.

Two additional data sets were used. A gene-function data set was generated from yeast data available at the web site of the Munich Information Center for Protein Sequences [17]. The highest level of the gene localization, protein class, complex, pathway and phenotype hierarchies were used to predict a function, "metabolism." Since proteins can have multiple localizations and other properties, each domain value was taken as a Boolean attribute that was 1 if the protein is known to have the localization and 0 otherwise leading to 146 Boolean attributes. A second data set was generated from spatial data. The RGB colors in the photograph of a cornfield are used to predict the yield of the field. Class label is the first bit of the 8-bit yield information; i.e., the class label is 1 if yield is higher than 128 for a given pixel. No preprocessing of the data was done. Some attributes, however, were identified as being logarithmic in nature, and the logarithm was encoded in P-trees: "capital-gain" and "capital-loss" of the adult data set, and all attributes of the "spam" data set.

3.2 Results

Table 2 compares the results for our decision-tree-based algorithm with the results for C4.5. It can be seen that for the bioinformatics data set the error rate clearly decreases for both lazy decision tree implementations. This supports the claim that while C4.5 is suitable for data sets with a small number of attributes of traditional data sets, its usefulness decreases in the face of new applications such as bioinformatics data with hundreds of attributes. Note that the gene-function data set is still smaller than most data sets in the bioinformatics field. It can furthermore be seen that the lazy decision tree algorithm that involves 20 paths is competitive all data sets, which is a significant achievement since C4.5 involves many advanced tree pruning steps that cannot be done for lazy classification that lacks the extensive tree construction phase.

Table 1. Error Rates of decision-tree-based classification.

	C4.5	Decision-tree-based	(+/-)	20 paths
adult	14.0	16.0	0.3	14.9
spam	7.2	11.5	0.9	7.1
sick-euthyroid	2.2	2.8	0.5	2.9
kr-vs-kp	0.5	1.7	0.4	0.8
mushroom	0	0	0	0
gene-function	15.7	15.5	0.8	15.5
crop	19.0	18.8	0.2	19.0

3.3 Decision-tree Results Compared with 20 Paths

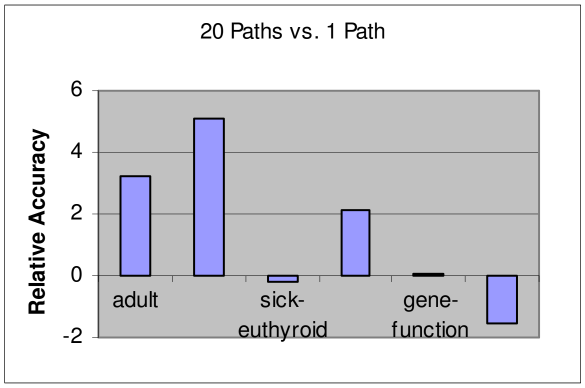


Figure 4. Difference between a vote based on 20 paths and a vote based on 1 path in the decision-tree-based algorithm in units of the standard error.

For the decision-tree-based algorithm a significance level of 1% was chosen. Information gain was calculated on two-thirds of the training data with significance being evaluated on the rest. The final vote was based on the entire training set. Accuracy was generally higher when a maximum of 20 paths were pursued. It can be seen in Figure 4 that the difference is particularly large for the spam data set that has many continuous attributes, which is a setting in which the algorithm is expected to work particularly well. The crop data set only had three attributes and therefore did not benefit.

3.3 Performance

Standard decision tree algorithms, whether they are eager or lazy, are based on database scans that scale linearly with the number of data points. The linear scaling is a serious problem in data mining problems that deal with thousands or millions of data points. A main benefit of using of P-trees lies in the fact that the AND operation that replaces database scans benefits from compression at every level. As a consequence, we see significantly sub-linear scaling. Figure 6 shows the execution time for decision-tree-based classification as a function of the number of data points for the adult data set. The solid line indicates a linear fit of the data. The actual performance results clearly show a less significant increase in execution time with data set size.

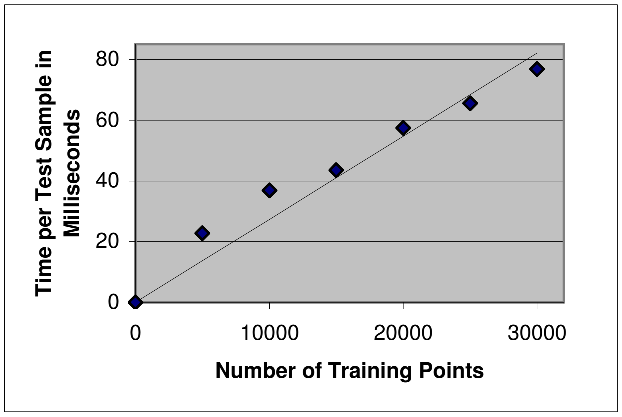


Figure 6. Scaling of execution time as a function of training set size. Diamonds correspond to measured execution time, and the line to a linear interpolation.

4. Conclusions

We have introduced a lazy decision-tree-based classifier that is particularly suitable to high-dimensional data and data that changes frequently. Our algorithms uses distance information within continuous attributes consistently by considering neighborhoods with respect to the unknown sample. Performance is achieved by using the P-tree data structure that allows efficient evaluation of counts of training points with particular properties. Neighborhoods are defined using the HOBbit distance that is particularly suitable to the bit-wise nature of the P-tree representation. We show that accuracy of our classifier can be improved by constructing multiple branches. We could show that for a bioinformatics data set, with many attributes, the accuracy of our algorithm is higher than that of C4.5. For traditional data sets with few attributes accuracies are comparable to C4.5 without the need of constructing and pruning a full decision tree. Finally, we showed that our algorithm has better than O(N) scaling with the number of training points.

References

[1] J. R. Quinlan, "C4.5: Programs for Machine Learning," Morgan Kaufmann Publishers, San Mateo, CA, 1993.
[2] D. Hand, H. Mannila, and P. Smyth, "Principles of Data Mining," The MIT Press, Cambridge, MA, 2001.
[3] J. Friedman, R. Kohavi, and Y. Yun, "Lazy Decision Trees," 13th Nat. Conf. on Art. Intell., 1996.
[4] R. Duda, P. Hart, and D. Stork, "Pattern Classification," 2nd edition, John Wiley and Sons, New York, 2001.
[5] W. Perrizo, Q. Ding, A. Denton, K. Scott, Q. Ding, M. Khan, "PINE-Podium Incremental Neighbor Evaluator for Spatial Data Using P-trees," Symposium on Appl. Computing (SAC'03), Melbourne, FL, 2003.
[6] S. Cost, and S. Salzberg, "A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features," Machine Learning, Vol. 10, 57-78, 1993.
[7] A. Perera, A. Denton, P. Kotala, W. Jackheck, W. Valdivia Granda, and W. Perrizo, "P-tree Classification of Yeast Gene Deletion Data," SIGKDD Expl., Dec. 2002.
[8] R. Kohavi, and G. John, "Wrappers for Feature Subset Selection," Art. Intelligence, Vol. 1-2, pp. 273-324, 1997.
[9] K. Bennett, M. Momma, and M. Embrechts, "A Boosting Algorithm for Heterogeneous Kernel Models," SIGKDD '02, Edmonton, Canada, 2002.
[10] Q. Ding, W. Perrizo, and Q. Ding, “On Mining Satellite and Other Remotely Sensed Images,” Workshop on Data Mining and Knowledge Discovery (DMKD-2001), Santa Barbara, CA, pp. 33-40, 2001.
[11] C. Shannon, "A Mathematical Theory of Communication," Bell Systems Tech. Journ. l, Vol. 27, pp. 379-423 and 623-656, July and October, 1948.
[12] M. Khan, Q. Ding, and W. Perrizo, "K-nearest Neighbor Classification of Spatial Data Streams Using P-trees," PAKDD-2002, Taipei, Taiwan, May 2002.
[13] W. Perrizo, W. Jockheck, A. Perera, D. Ren, W. Wu, and Y. Zhang, "Multimedia Data Mining Using P-trees," Multimedia Data Mining Workshop, KDD’02, Edmonton, Canada, Sept. 2002.
[14] W. Ledermann, "Handbook of Applicable Mathematics," Vol. 6, Wiley, Chichester, 1980.
[15] L. Breiman, "Bagging Predictors," Machine Learning, Vol. 24, No. 2, pp. 123-140, 1996.
[16] C.L. Blake, and C.J. Merz, "(UCI) Repository of Machine Learning Databases," Irvine, CA, 1998. http://www.ics.uci.edu/~mlearn/MLSummary.html,
[17] MIPS,"Comprehensive Yeast Genome Database," http://mips.gsf.de/genre/proj/yeast/index.jsp.