Softening Splits in Decision Trees Using Simulated Annealing

Jakub Dvořák and Petr Savický

Institute of Computer Science, Academy of Sciences of the Czech Republic
{dvorak,savicky}@cs.cas.cz

Abstract. *Predictions computed by a classification tree are usually constant on axis-parallel hyperrectangles corresponding to the leaves and have strict jumps on their boundaries. The density function of the underlying class distribution may be continuous and the gradient vector may not be parallel to any of the axes. In these cases a better approximation may be expected, if the prediction function of the original tree is replaced by a more complex continuous approximation. The approximation is constructed using the same training data on which the original tree was grown and the structure of the tree is preserved.*
The current paper uses the model of trees with soft splits suggested by Quinlan and implemented in C4.5, however, the training algorithm is substantially different. The method uses simulated annealing, so it is quite computationally expensive. However, this allows to adjust the soft thresholds in groups of the nodes simultaneously in a way that better captures interactions between several predictors than the original approach. Our numerical test with data derived from an experiment in particle physics shows that besides the expected better approximation of the training data, also smaller generalization error is achieved.

Keywords: decision trees, soft splits, classification, simulated annealing.

1 Introduction

Classification trees are suitable for predicting the class in complex distributions, if a large sample from the distribution is available. The classical parametrical methods may not succeed in such situations, if they work with a closed formula describing the density in the whole predictor space. Decision trees and ensambles of trees are comparable to neural networks and SVM in classification accuracy. The predictor vector for a tree consists of a fixed number of numerical and categorical variables. In this paper, we consider single univariate decision trees with numerical predictors.

A trained classification tree usually does not only provide a discrete classification, but also an estimate of the confidence for it on a continuous scale. This confidence may be an estimate of the conditional probability of the classes, but this is not necessary. Even if it is not a good estimate of the probabilities, it may be a reasonable information. In the real world problems, it is frequently plausible to assume that the function which assigns such a confidence to each point of the predictor space is continuous. Even if the true distribution has a sharp boundary between the classes, a limited sample from the distribution does not provide enough information for a justified construction of a prediction function with a strict jump.

Classification trees constructed using the traditional methods like CART [2] or C4.5 [4] generate trees, whose internal nodes contain conditions of the form $x_{k_j} \leq c_j$, where x_{k_j} is one the predictors and c_j is a threshold value. The result of the use of such sharp threshold conditions is that the predictions computed by a classification tree are constant on axis-parallel hyperrectangles corresponding to the leaves and have strict jumps on their boundaries. Such functions may badly approximate boundaries of different type. This is the cost, which is paid for the simplicity of the classifier.

A soft threshold condition means that if the actual value of x_{k_j} is close to c_j, then both branches of the tree are evaluated and their results are combined using weights changing continuously with $x_{k_j} - c_j$. Soft splits were suggested first by Quinlan [4] including the implementation in C4.5. We use exactly the same extension of the decision tree classifiers, but use a substantially different technique for training.

Quinlan's technique determines the soft thresholds in each node separately using statistical estimation. In our approach, several thresholds corresponding to a group of nodes close to each other in the tree are adjusted simultaneously. Hence, the choice of the final thresholds is influenced also by the interactions between predictors. A nonsoft tree together with the training data used for its construction, are used as the input to an optimization phase, which tries to find the values of the parameters of the soft thresholds, which yield the best possible approximation of the training data. Since the structure of the tree, in particular, its number of nodes, is fixed during the optimization, overfitting was not observed in our experiments, although a computationally intensive optimization is used to tune the soft thresholds.

The goal of the postprocessing of the tree is to reach a smoother function that fits better the training data. Since the complexity of the classifier does not increase too much, one may expect to achieve also

smaller generalization error. Besides better approximation in cases, where the unknown conditional probability function is continuous, we may obtain a better approximation even if the true value of the conditional probability makes a jump on a boundary between the regions of different classification, if the boundary is not in axis-parallel direction. Soft tree may represent a more complex function than a nonsoft tree. In particular, as a consequence of interactions of several predictors, the prediction function of a soft tree may have gradient vector in a general direction, while keeping the small number of nodes of the original tree. Approximating this using a nonsoft tree requires to use a stair like boundary with a large number of nodes.

A situation, where soft thresholds are well justified directly from an application domain, may be demonstrated for example on evaluating custumers asking for a credit card in a bank. The bank investigates several parameters of each customer, for example the current amount on his account, his regular monthly deposit amount, and some other criteria. Insufficiency in one of these criteria may be compensated by good values in some other. Instead of designing a tree with a large number of nodes, an approximating smaller tree with soft splits may be sufficient.

The current paper investigates classification accuracy on data from particle physics (MAGIC gamma telescope) considered already in [1]. In this comparison, ensambles of trees, in particular random forests, provided the best classifiers for these data. Our experimental results on these data demonstrate that the trees obtained by introducing soft splits may have substantially smaller generalization error than individual nonsoft trees.

We also compare the soft trees obtained by simulated annealing with soft trees obtained by C5.0[1]. There is no significant difference in test classification error between these two types of trees, see section Results, although the trees are obtained using substantially different principles. C5.0 finds the soft thresholds as bounds of confidence intervals based on a statistical model. It does not take into account whether the error on the training data changes or not. In fact, using thresholds constructed in this way frequently increases the training error. On the other hand, our approach optimizes the soft threshold purely by minimizing the training error. Our result shows that minimizing training error leads to similar results as the satistical estimation used in C5.0, at least on the MAGIC data.

[1] http://www.rulequest.com, commercial version of C4.5.

2 Decision tree with soft splits

Let T be a decision tree with nodes v_j for $j = 1, \ldots, s$. We assume that if v_{j_1} and v_{j_2} are left and right successor of v_j, then $j < j_1$ and $j < j_2$. In particular, v_1 is the root. Let V be the set of indices of the internal nodes and U the set of indices of the leaves. The variable tested in node v_j is denoted x_{k_j} and the corresponding threshold value is denoted c_j. If C is the number of classes, then the label (response vector) $G(v)$ of a leaf v is a nonnegative real valued vector of dimension C, whose coordinates sum up to one. Let $T(x)$ be the function $\mathrm{R}^d \to \mathrm{R}^C$ computed by the tree, where d is the number of predictors. More generally, let $T_j(x)$ be the function computed by the subtree starting at v_j. In particular, $T_1(x) = T(x)$. Note that $T_j(x)$ is defined even in cases, when the computation of the whole tree T for x does not reach the node v_j.

If v_{j_1} and v_{j_2} are left and right successor of v_j, then we have $T_j(x) =$ if $x_{k_j} \le c_j$ then $T_{j_1}(x)$ else $T_{j_2}(x)$. If we define $I(\mathit{condition})$ equal to 1 if *condition* is true and equal to 0 if *condition* is false, then this is equivalent to

$$T_j(x) = I(x_{k_j} \le c_j) T_{j_1}(x) + I(x_{k_j} > c_j) T_{j_2}(x).$$

A tree with soft splits is obtained by replacing this by

$$T_j(x) = L_j(x_{k_j} - c_j) T_{j_1}(x) + R_j(x_{k_j} - c_j) T_{j_2}(x)$$

for appropriate continuous functions $L_j, R_j : \mathrm{R} \to [0, 1]$. It is required that if both subtrees of T_j return the same output vector, then T_j returns the same vector as well. Hence, we require $L_j(t) + R_j(t) = 1$ for all $t \in \mathrm{R}$. A natural further requirement is that L_j be nonincreasing with limits $L_j(-\infty) = 1$ and $L_j(\infty) = 0$. Hence, we also have that R_j is nondecreasing with limits $R_j(-\infty) = 0$ and $R_j(\infty) = 1$.

The functions L_j and R_j used in the current paper are piecewise linear functions interpolating the points in the table

t	$L_j(t)$	$R_j(t)$
$-\infty$	1	0
$-a_j$	1	0
0	1/2	1/2
b_j	0	1
∞	0	1

where the values $a_j \ge 0$ and $b_j \ge 0$ are parameters of the soft splits. If $a_j > 0$ and $b_j > 0$, then the functions L_j, R_j are uniquely determined by the above table. If some of the values a_j, b_j is zero, the corresponding function L_j, R_j is defined as a pointwise limit, which is noncontinuous. The limit function satisfies $L_j(0) = 1/2$ or $R_j(0) = 1/2$ as in any other case.

The tree with soft splits is obtained by replacing the evaluation function in each internal node by the

above one. This requires to specify the parameters (a_j, b_j) in all internal nodes. Let $\theta = \{(a_j, b_j)\}_{j \in V}$. The functions L_j, R_j defined above depend on θ. So, the correct notation for them is $L_j(\theta, t), R_j(\theta, t)$. Moreover, let $T(\theta, x)$ and $T_j(\theta, x)$ denote the functions computed by the whole tree and the tree starting at node v_j, respectively, if the soft splits determined by θ are used. We again have $T(\theta, x) = T_1(\theta, x)$.

If $x_{k_j} = c_j$, then the soft split always evaluates both subtrees and returns their arithmetic mean. If $x_{k_j} \neq c_j$ the situation depends on x_{k_j} as follows. If $x_{k_j} \leq c_j - a_j$ or $x_{k_j} \geq c_j + b_j$, then the evaluation function in node v_j behaves in the same way as in the original tree and returns the value of one of the two subtrees. If $x_{k_j} \in (c_j - a_j, c_j + b_j)$, then evaluating $T_j(x, \theta)$ requires to compute both subtrees and the output is a combination of their values.

Note that $T(0, x)$ behaves similarly to $T(x)$, but there is a difference. If the computation of $T(x)$ never reaches a node, where $x_{k_j} = c_j$, then we have $T(0, x) = T(x)$. However, if $x_{k_j} = c_j$ is satisfied at some step of the computation, the results may differ, since evaluation of $T(0, x)$ combines both subtrees, while evaluation of $T(x)$ uses only one of them.

For every pair of nodes v_j and v_{j_1} such that v_{j_1} is one of the two successors of v_j, let $H(\theta, v_j, v_{j_1})(x)$ be the following function defined on the predictor vector x.

$$H(\theta, v_j, v_{j_1})(x) = \begin{cases} L_j(\theta, x_{k_j}) & \text{if } v_{j_1} \text{ is the left} \\ & \text{successor of } v_j \\ R_j(\theta, x_{k_j}) & \text{if } v_{j_1} \text{ is the right} \\ & \text{successor of } v_j \end{cases}$$

For every leaf v, let $Path(v)$ be the uniquely determined path from the root to v. Then an explicit formula for the function computed by a tree with soft splits is

$$T(\theta, x) = \sum_{\substack{j \in U \\ (u_1, \ldots, u_k) = Path(v_j)}} G(v_j) \prod_{i=2}^{k} H(\theta, u_{i-1}, u_i)(x).$$

The formula may be verified by induction starting at the leaves.

3 Optimization of the soft splits

The method described in this paper assumes that a nonsoft classification tree T with two classes 0 and 1 is available. Such a tree may be obtained using a method like CART or C4.5 without softening. We used the R implementation of CART. The response vector is two dimensional in this case and the two coordinates are assumed to sum up to one. Hence, each of the coordinates alone carries the full information on the prediction. Let us denote $T^*(x)$ the component of the response vector computed by tree T, which corresponds to class 1. The method of the current paper assumes that it is possible to find a threshold h such that the rule "predict 1 iff $T^*(x) \geq h$" provides a reasonable classification.

The goal is to find θ such that the error of classification using $T^*(\theta, x) \geq h$ on unseen cases is smaller than in the original tree. Since the algorithm has access only to the training data, the method tries to achieve the above goal by minimizing the error of $T^*(\theta, x)$ on the training data. This error may be measured in different ways. We tested first simply the classification error, but it appeared to be better to use a continuous error defined on the data (x_i, y_i), $i = 1, \ldots, m$, $y_i \in \{0, 1\}$, by the formula

$$f(\theta) = \sum_{i=1}^{m} \mathrm{e}^{\alpha(|T^*(\theta, x_i) - y_i| - 1)}, \qquad (1)$$

where α was chosen to be 4. Experiments show that an improvement on unseen cases is indeed achieved.

Since the minimized function $f(\theta)$ is not a smooth function and has a large number of local minima, we used simulated annealing available in R Statistical Package [5] using method SANN of function "optim". Since this does not allow to restrict the range of θ to nonnegative values, we used a large penalty (number of errors larger than the number of training cases) for θ, which contain a negative value. The initial value of θ was chosen to be 0, i.e. the optimization starts approximately at the original nonsoft tree.

The dimension of the optimization problem (the number of parameters of the minimized function) is two times the number of internal nodes. In order to make the optimization process independent on the scaling of the data, the optimization function uses a normalized vector of parameters $\theta' = \{(a'_j, b'_j)\}_{j \in V}$, where $a'_j = a_j / a_{j,0}$, $b'_j = b_j / b_{j,0}$. The normalizing factors $a_{j,0}, b_{j,0}$ are defined using the original nonsoft tree as follows.

In each node of the tree, we find the hyperrectangle that is guaranteed to contain all training points that go through the node during classification. For the root, the hyperrectangle is the cartesian product of the smallest closed intervals, which contain all the values of the corresponding predictor in the training data. If v_{j_1}, v_{j_2} are the two successors of v_j, then the hyperrectangles assigned to them are obtained by splitting the hyperrectangle assigned to v_j by the hyperplane $x_{k_j} = c_j$. Then, the values $a_{j,0}$ and $b_{j,0}$ are chosen so that the interval $[c_j - a_{j,0}, c_j + b_{j,0}]$ is exactly the range of x_{k_j} within the hyperrectangle assigned to v_j.

4 Iteration of simulated annealing

In this section, we discuss the strategy to look for θ that minimizes the function (1). For the purpose of this section, we denote θ as $x \in \mathrm{R}^n$, where n is the length of θ. It appeared to be better to split the minimization into phases, in each of which, only a small randomly chosen subset of arguments is modified. Let us introduce the following notation for this purpose. Let $S \subseteq \{1, \ldots, n\}$ and $z \in \mathrm{R}^n$. By R^S we mean the set of vectors $\{x_i\}_{i \in S}$, i.e. the vectors from $\mathrm{R}^{|S|}$ whose coordinates are indexed by elements of S instead of consecutive integers. Then, let $f[S, z] : \mathrm{R}^S \to \mathrm{R}$ be the function defined for every $x \in \mathrm{R}^S$ by $f[S, z](x) = f(y)$, where

$$y_i = \begin{cases} x_i & \text{if } i \in S \\ z_i & \text{otherwise} \end{cases}$$

Optimization is performed by a sequence of calls of method SANN of optim, which is an implementation of simulated annealing. The initial approximation of each call is the best solution found during the previous call. The initial temperature for all calls is $temp = 10$ and the bound on the number of iterations is $maxit = 101$ for all calls.

For each call of optim, a set S of k indices of variables is selected, see below. In the given call, $f(x)$ is minimized by modifying only variables with indices in S. Formally, the minimized function is the function $f[S, x_0](x)$ with $k = |S|$ arguments, where x_0 is the result of the previous call. The restriction of x_0 to the selected set S of indices is also the initial value for x in the current call.

One call of optim is successful, if it succeeds to find a better solution than the initial one. The whole process stops, when 50 consecutive calls are unsuccessful.

As mentioned above, for each call of optim, a set S of indices of variables is selected. Let s be a variable from the vector θ. This means that s is a_j or b_j for some j. Then, let T_s be the maximal subtree of the tree T, such that the root of T_s is the left son of the node v_j in the case that s is a_j and the right son of v_j if s is b_j. The selection of S starts with selecting randomly a variable s such that the root of T_s is not a leaf. Then S contains s and variables a_i, b_i where i traces through all indexes of nodes in the two top levels of T_s. Depending on the structure of T and selection of s the set S contains 3, 5 or 7 variables.

5 Experimental setup

In a single run of the experiment, the available data D were split at random in ratio 2:1 into a training set D_1 and a test set D_2 and the four classifiers obtained by the following methods were constructed:

1. CART.
2. Soft tree obtained by the method described in the prevoius section from CART trees.
3. C5.0 without softenning.
4. C5.0 with softenning (option "`-p`").

More detail is given in subsections below.

5.1 CART

The trainig set D_1 was further split in ratio 2:1 into D_{11} and D_{12}. The larger part D_{11} was used for growing the tree, the smaller D_{12} was used for pruning as the validation set (cost complexity pruning in CART method). The result of the pruning is a sequence of trees of different sizes. Accuracy of these trees is reported for comparison with the other methods. In order to show that the comparison result does not depend on the selection of the tree in the sequence, we select the best tree on the test set D_2 and report its accuracy. Even such trees are worse than the soft trees, whose error is measured using standard methodology.

5.2 Softening trees from CART

We use the sequence of pruned trees constructed by CART. Trees without split nodes are not considered. When interpreting the soft tree as a classifier, we used threshold $h = 0.5$. This means that the class with the larger confidence (we have two classes) in the response vector is predicted.

The error of the resulting soft tree is never worse than in the original tree, however, it is sometimes close to it. Such soft trees are discarded. The optimization is considered unsuccessful, if the ratio of the error of the original tree over the error of the soft tree is less than 1.01. The sequence of trees from CART contains trees of different sizes. Smaller trees have higher chance to be improved by one run of the softening procedure. On the other hand, if the softening procedure succeeds to improve a large tree, the result is usually better than for small trees. In order to balance between these two effects, we used a strategy which is splitted into steps numbered by $i = 1, 2, \ldots$. In step i, the softening procedure tries to improve all of the i largest trees in the sequence. The process terminates, when 10 trees successfuly improved by softening are collected. The resulting classifier is determined as the tree with the smallest classification error on D_1 among the 10 trees obtained by the above strategy. The error of this tree on D_2 is reported.

5.3 C5.0

We used C5.0 release 1.15. The confidence level, which determines the amount of pruning was chosen 0.1 (option "`-c 10`"), which appeared to be the best among

several values that we tried. For each split of the data, C5.0 was run twice, with and without the option "-p", which forces that a softened tree is constructed.

The reason for choosing the confidence level equal to 0.1 was the following: We computed for each split of the data error rates of C5.0 softened trees for the values of confidence level 0.01, 0.02, 0.05, 0.10, 0.15, 0.20, 0.25 and 0.30. Then for each of these values we compared the error rate of C5.0 and the error rate of the tree from CART softened. The value 0.10 is the one, for which the error rates of C5.0 softened trees are lower than the error rates of softened trees from CART in the highest number of data splits.

6 Results

We used data simulating registration of gamma and hadron particles in Cherenkov imaging gamma ray telescope MAGIC [1]. There are 10 numerical predictors and 2 classes. The predictors are numerical values that are produced by the registration device and characterize the registered particle. Class signal represents cases, where the registered particle is gamma. Class background corresponds to hadrons, mostly protons. The number of cases in the dataset is 19020.

The data were created by a complex Monte Carlo simulation [3] that approximates the development of a shower of particles generated by a high energy primary particle that reaches the atmosphere. The result of the simulation is an estimate of the number of Cherenkov ultraviolet photons that reach different pixels in the focus of an antenna at the ground and form a single registered event. The 10 predictors are numerical parameters of the geometric form of the obtained image. Generating each case in the dataset required several seconds of CPU time.

Creating the dataset was a part of the project of constructing the telescope and was used to support the decision, which classification technique to use in regular observations using the telescope. On the basis of [1], random forest was selected and is still used.

We used 7 random splits of this set in the ratio 2:1 into D_1 and D_2. For each of these splits, four classifiers were constructed using the methods described in the previous section. Besides the nonsoft CART trees, the classifier was constructed using only D_1 and its accuracy was estimated using D_2. Due to the large size of both training and testing set, there is a strong relationship between the training error and test error. The test errors on D_2 are presented in the following tables.

	CART non-soft	CART softened	ratio soft/non-soft
1	0.1677	0.1377	0.8213
2	0.1610	0.1371	0.8511
3	0.1598	0.1366	0.8549
4	0.1659	0.1393	0.8394
5	0.1591	0.1377	0.8652
6	0.1550	0.1380	0.8901
7	0.1533	0.1371	0.8940

The next table allows to compare the error rates of trees from CART softened and trees from C5.0 softened. The ratio of these two errors is included.

	CART softened	C5.0 non-soft	C5.0 softened	ratio CART s./C5.0 s.
1	0.1377	0.1473	0.1391	0.9898
2	0.1371	0.1535	0.1438	0.9529
3	0.1366	0.1456	0.1323	1.0322
4	0.1393	0.1508	0.1380	1.0091
5	0.1377	0.1478	0.1366	1.0081
6	0.1380	0.1516	0.1483	0.9309
7	0.1371	0.1479	0.1410	0.9720

In our experiments, softening using simulated annealing led to soft trees with similar accuracy on the MAGIC dataset compared to that of C5.0 softened tree. In order to formulate the result in terms of statistical significance, let p_1 be the probability of the event that the error rate of a tree softened using simulated annealing on the MAGIC dataset is lower than error rate of C5.0 softened tree. The two sided 0.95 confidence interval for p_1 obtained using the exact binomial test is approximately $[0.18, 0.9]$, since we have 4 successes out of 7 trials. This means that the result of the experiment does not imply any significant difference between the two methods.

The comparison of trees softened using simulated annealing with the non-soft trees from the CART method gives much better result. The error of the soft tree obtained by SANN was always by at least 10% better than that of the corresponding nonsoft tree. In order to formulate the result in terms of statistical significance, let p_2 be the probability of the event that the error rate of a tree softened using simulated annealing on the MAGIC dataset is lower than 90 % of error rate of CART tree. The one-sided lower 0.95 confidence limit for p_2 obtained using the exact binomial test is approximately 0.65, since we have 7 successes out of 7 trials.

Acknowledgement. The first author was supported by Czech Science Foundation (GACR) under the grant No. GD201/05/H014. The second author was partially supported by the "Information Society" project 1ET100300517. Both authors were par-

tially supported also by the Institutional Research Plan AV0Z10300504.

References

1. R.K. Bock, A. Chilingarian, M. Gaug, F. Hakl, T. Hengstebeck, M. Jiřina, J. Klaschka, E. Kotrč, P. Savický, S. Towers, A. Vaicilius, "Methods for multidimensional event classification: a case study using images from a Cherenkov gamma-ray telescope." *Nuclear Instruments and Methods in Physics Research*, Section A, Volume 516, Issue 2-3, p. 511-528, 2004.
2. L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression Trees*, Belmont CA: Wadsworth, 1993.
3. D.Heck et al., CORSIKA, *A Monte Carlo code to simulate extensive air showers*, Forschungszentrum Karlsruhe FZKA 6019, 1998.
4. J.R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, San Mateo — California, 1993.
5. R Development Core Team (2005). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.r-project.org.