Regularizing Soft Decision Trees

Olcay Taner Yıldız and Ethem Alpaydın

Abstract Recently, we have proposed a new decision tree family called soft decision trees where a node chooses both its left and right children with different probabilities as given by a gating function, different from a hard decision node which chooses one of the two. In this paper, we extend the original algorithm by introducing local dimension reduction via L_1 and L_2 regularization for feature selection and smoother fitting. We compare our novel approach with the standard decision tree algorithms over 27 classification data sets. We see that both regularized versions have similar generalization ability with less complexity in terms of number of nodes, where L_2 seems to work slightly better than L_1.

1 Introduction

A decision tree is an hierarchical structure made up of internal decision nodes and terminal leaves. For classification, the leaves carry the label of one of K classes. The input vector is composed of d attributes, $x = [x_1, \ldots, x_d]^T$. Each decision node m implements function $v_m(x)$ and chooses one of the children accordingly. Let $F_m(x)$ be the output generated by the subtree whose root is m and in a binary tree, let $F_m^L(x)$ and $F_m^R(x)$ denote respectively its left and right children and $v_m(x)$ hence has two outcomes:

$$F_m(x) = \begin{cases} F_m^L(x) & \text{if } v_m(x) > 0 \text{ /* true */} \\ F_m^R(x) & \text{otherwise /* false */} \end{cases} \quad (1)$$

Olcay Taner Yıldız
Department of Computer Engineering, Işık University, TR-34980, Istanbul, Turkey e-mail: olcaytaner@isikun.edu.tr

Ethem Alpaydın
Department of Computer Engineering, Boğaziçi University, TR-34342, Istanbul, Turkey e-mail: alpaydin@boun.edu.tr

Given an input to classify, starting from the root node, one applies the function at each internal node and the input is forwarded to one of the two branches depending on the outcome. This process is repeated recursively until a leaf node is hit at which point the class label of the leaf constitutes the output. Depending on the model they assume for $F_m(x)$, decision trees are subcategorized into univariate decision trees [1], multivariate linear decision trees [2], multivariate nonlinear decision trees [3], and omnivariate decision trees [4].

In our recent work [5], we generalized decision trees and proposed soft decision trees. A node at a hard decision tree forwards the input to either its left or the right subtree, whereas a soft decision tree node utilizes a gating function to assign probabilities to its children and merges the decision of its children by these probabilities. That is, we follow all the paths to all the leaves and all the leaves contribute to the final decision but with different probabilities.

In this paper, we extend soft decision trees by adding a regularization term (linear for L_1 regularization, quadratic for L_2 regularization) to handle a localized dimensionality reduction in the nodes, since as we go deeper into the tree, the scope of a node becomes more localized. This paper is organized as follows: In Section 2, we briefly review the original soft tree algorithm. We give the details of our regularized version of the original approach in Section 3. We give our experimental results in Section 4 and conclude in Section 5.

2 Soft Decision Trees

As opposed to the (hard) decision mode which redirects instances to its left or right subtree depending on the node function $F_m(x)$, soft decision node redirects instances both to the left and right subtree with probabilities calculated by the gating function $v_m(x)$:

$$F_m(x) = F_m^L(x)v_m(x) + F_m^R(x)(1 - v_m(x)) \tag{2}$$

and to choose among two children, we take $v_m(x) \in [0,1]$ as the *sigmoid function*:

$$v_m(x) = \frac{1}{1 + \exp[-(w_m^T x + w_{m0})]} \tag{3}$$

Learning the tree is incremental and recursive, as with the hard decision tree. The algorithm starts with one node and fits a constant model. Then, as long as there is improvement, it replaces the leaf by a subtree. This involves optimizing the gating parameters and the values of its children leaf nodes by gradient-descent over an error function. The error function is cross-entropy for classification, and the final output of the tree is filtered through a sigmoid at the root to convert it to a probability:

$$E = r \log y + (1 - r) \log(1 - y)$$

3 Regularized Soft Decision Trees

In general, model selection problem in decision trees have two faces. On the one hand, keeping the node model fixed, one can delve into the optimization of the tree structure and use either pre or post-pruning techniques. On the other hand, one can try to solve the model selection problem at the node level and select one among L candidate models based on both complexity and performance [4].

Our approach in this paper falls into the second category and we use L_1 and L_2-norm regularization techniques to combine the model complexity and error term into one single value. The function we want to minimize is

$$E_{L_1} = \frac{1-\lambda}{N}\sum_t (r^{(t)}\log y^{(t)} + (1-r^{(t)})\log(1-y^{(t)})) + \frac{\lambda}{d}\sum_{i=0}^{d}|w_{mi}|$$

for L_1 regularization, and

$$E_{L_2} = \frac{1-\lambda}{N}\sum_t (r^{(t)}\log y^{(t)} + (1-r^{(t)})\log(1-y^{(t)})) + \frac{\lambda}{d}\sum_{i=0}^{d}w_{mi}^2$$

for L_2 regularization, where $1-\lambda$ and λ correspond to the weight factor of cross-entropy and model complexity respectively. From a Bayesian perspective, L_1 and L_2 regularization corresponds to Laplacian and Gaussian priors on w_{mi} and the equations above correspond to maximum a posteriori estimates.

In gradient-descent, we use the following update equations:

$$\alpha_m = \prod_{p=m.parent}^{p!=root} \delta_{p,p.parent.left} v_p(x) + \delta_{p,p.parent.right}(1-v_p(x))$$

$$\frac{\partial E_{L_1}}{\partial w_{mi}} = \frac{1-\lambda}{N}(r-y)(F_m^L(x)-F_m^R(x))\alpha_m v_m(x)(1-v_m(x))x_i + \frac{sgn(w_{mi})\lambda}{d} \quad (L_1)$$

$$\frac{\partial E_{L_2}}{\partial w_{mi}} = \frac{1-\lambda}{N}(r-y)(F_m^L(x)-F_m^R(x))\alpha_m v_m(x)(1-v_m(x))x_i + \frac{\lambda w_{mi}}{d} \quad (L_2)$$

where $\delta_{x,y}$ is the Kronecker delta and sgn(x) is the sign function.

4 Experiments

To compare the generalization error and model complexity of our regularized soft trees with both soft trees (without regularization) and hard C4.5 trees, we use 27 two-class data sets from UCI repository [6]. We also compare with a multivariate linear tree algorithm (Ldt) [7]. We first separate one third of the data set as the test set over which we evaluate the final performance. On the remaining two thirds, we apply 5×2-fold cross validation, which gives a total of 10 folds for each data set. We

Table 1 On the classification data sets, the average error of soft, hard, linear discriminant trees (Ldt). Pairwise comparisons of error rate of soft and hard decision tree algorithms are shown in the second table.

Dataset	Hard	Ldt	Soft	Soft(L_1)	Soft(L_2)
acceptors	16.1 ± 2.0	9.6 ± 0.8	8.7 ± 0.7	7.5 ± 0.3	7.3 ± 0.3
artificial	1.1 ± 1.8	1.5 ± 1.9	1.1 ± 1.8	0.7 ± 1.6	0.4 ± 1.2
breast	6.7 ± 1.1	4.9 ± 0.6	3.5 ± 0.7	4.1 ± 0.8	3.5 ± 0.7
bupa	38.6 ± 4.1	39.1 ± 3.4	39.7 ± 4.2	41.4 ± 3.0	39.0 ± 2.4
donors	7.7 ± 0.4	5.4 ± 0.3	5.7 ± 0.4	5.6 ± 0.3	5.5 ± 0.3
german	29.9 ± 0.0	25.8 ± 2.0	24.0 ± 3.0	25.9 ± 2.4	24.3 ± 1.4
haberman	26.6 ± 0.3	27.2 ± 1.5	25.9 ± 1.8	25.0 ± 2.5	24.7 ± 2.4
heart	28.3 ± 4.7	18.4 ± 2.3	19.7 ± 3.4	18.1 ± 2.5	18.3 ± 2.5
hepatitis	22.1 ± 4.4	20.4 ± 2.9	20.2 ± 2.4	20.4 ± 4.2	19.0 ± 3.6
ionosphere	13.1 ± 1.9	12.3 ± 2.2	11.5 ± 2.0	11.5 ± 1.9	12.6 ± 1.1
krvskp	1.2 ± 0.4	4.5 ± 0.7	1.8 ± 0.6	3.6 ± 0.6	3.0 ± 0.7
magic	17.5 ± 0.6	16.9 ± 0.1	14.7 ± 0.5	21.6 ± 0.2	20.8 ± 0.2
monks	12.8 ± 7.8	23.8 ± 8.2	0.0 ± 0.0	3.4 ± 5.3	3.5 ± 7.4
mushroom	0.0 ± 0.1	1.8 ± 0.5	0.1 ± 0.0	0.1 ± 0.0	0.0 ± 0.0
musk2	5.5 ± 0.6	6.4 ± 0.3	4.3 ± 0.7	6.3 ± 0.7	5.7 ± 0.7
parkinsons	13.8 ± 2.3	13.5 ± 2.5	14.3 ± 2.7	13.7 ± 2.7	12.3 ± 2.4
pima	27.9 ± 3.4	23.1 ± 1.4	24.9 ± 2.0	23.1 ± 0.8	23.1 ± 1.0
polyadenylation	30.5 ± 1.3	22.6 ± 0.6	22.9 ± 0.5	22.2 ± 0.5	22.3 ± 0.5
promoters	26.1 ± 9.9	34.4 ± 9.4	15.3 ± 6.7	13.1 ± 7.6	16.7 ± 9.3
ringnorm	12.2 ± 1.1	22.8 ± 0.3	9.9 ± 1.7	22.4 ± 0.3	22.4 ± 0.4
satellite47	15.4 ± 1.5	16.7 ± 1.4	12.4 ± 1.4	16.4 ± 1.4	15.4 ± 0.9
spambase	9.9 ± 0.7	10.1 ± 0.7	7.5 ± 0.5	8.2 ± 0.3	7.6 ± 0.3
spect	19.1 ± 2.8	20.1 ± 2.4	19.6 ± 2.4	16.9 ± 2.6	16.6 ± 1.8
tictactoe	23.8 ± 2.2	31.9 ± 2.4	1.8 ± 0.3	1.8 ± 0.4	1.7 ± 0.3
titanic	21.8 ± 0.5	22.4 ± 0.4	21.5 ± 0.2	21.6 ± 0.3	22.1 ± 0.2
twonorm	17.0 ± 0.7	2.0 ± 0.1	2.1 ± 0.2	2.1 ± 0.2	2.1 ± 0.2
vote	5.2 ± 0.7	6.7 ± 2.6	5.1 ± 0.9	6.7 ± 0.9	5.6 ± 1.1

	Hard	Ldt	Soft	Soft(L_1)	Soft(L_2)
Hard		6	0	5	4
Ldt	5		0	1	1
Soft	9	11		5	4
Soft(L_1)	7	4	1		0
Soft(L_2)	8	4	1	2	

use parametric 5×2 paired F-test [8] and nonparametric Nemenyi's test to compare algorithms.

Table 1 shows the average and standard deviation of errors of soft and hard decision trees. The table below it shows the pairwise comparison results of error rates—entry (i, j) in this second table gives the number of datasets (out of 27) on which method i is statistically significantly better than method j with at least 95% confidence using 5×2 paired F-test. Figure 1 shows the result of post-hoc Nemenyi's test applied on the error rates of the algorithms.

We see that the soft tree and its extensions are significantly more accurate than both hard and linear discriminant trees. Soft tree variants are better than hard trees and Ldt on eight and six datasets respectively; hard trees are only better on three

datasets and Ldt are better only on one dataset. L_2 regularization is significantly better than L_1 on two datasets. If we compare nonparametrically, on 18 datasets out of 27, L_2 normalization has smaller error rate, whereas L_1 has smaller error rate only on 9 datasets. Overall, L_2 regularization is slightly better than L_1 in terms of error rate (p-value = 0.06).

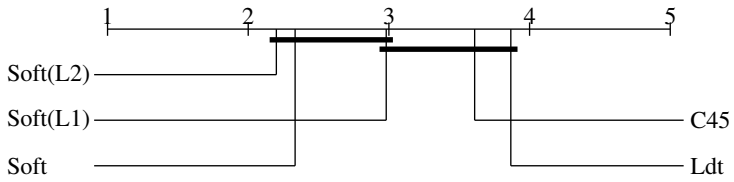

Fig. 1 The result of post-hoc Nemenyi's test applied on the error rates of soft, hard, linear discriminant trees (Ldt).

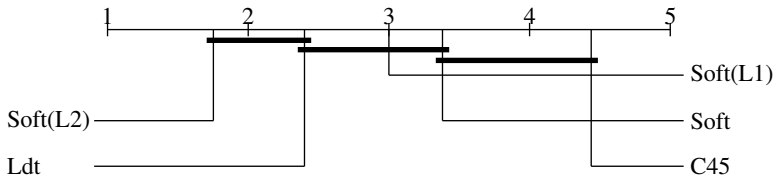

Fig. 2 The result of post-hoc Nemenyi's test applied on the number of nodes of trees generated by soft, hard, linear discriminant trees (Ldt).

Table 2 shows the average and standard deviation of number of nodes of soft and hard decision trees. Again the table below shows the pairwise comparison results in terms of number of nodes. Figure 2 shows the result of post-hoc Nemenyi's test applied on the number of nodes of tree generated by the algorithms.

As expected, soft tree variants are simpler than hard trees. On the average, soft tree variants are significantly smaller than hard trees on 12 datasets, whereas hard trees are smaller only on one dataset. The results also show that regularization pays off. Regularized soft trees are simpler than the original soft trees on three datasets. L_1 regularization is significantly better than L_2 on one dataset according to 5×2 paired F-test. If we compare nonparametrically, on 24 datasets out of 27, L_2 regularization generates smaller trees than L_1 whereas the opposite is true on only one dataset. We can conclude that L_2 regularization leads to significantly smaller trees than L_1 (p-value $< 10^{-6}$).

Table 2 On the classification data sets, the average number of decision nodes of soft, hard, linear discriminant trees (Ldt). Pairwise comparisons are shown in the second table.

Dataset	Hard	Ldt	Soft	Soft(L_1)	Soft(L_2)
acceptors	7.1 ± 6.9	1.1 ± 0.3	7.0 ± 3.9	4.0 ± 1.4	2.2 ± 0.8
artificial	4.4 ± 1.0	2.0 ± 0.5	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0
breast	4.1 ± 2.2	1.6 ± 0.7	1.3 ± 0.5	1.7 ± 0.7	1.3 ± 0.5
bupa	5.4 ± 3.7	1.7 ± 0.7	4.0 ± 2.3	3.5 ± 1.2	2.4 ± 0.8
donors	21.0 ± 3.7	3.7 ± 1.9	6.4 ± 3.2	3.2 ± 1.0	1.8 ± 0.9
german	0.0 ± 0.0	3.1 ± 3.5	4.1 ± 2.0	4.1 ± 1.8	1.7 ± 0.5
haberman	1.0 ± 3.2	1.2 ± 1.8	2.0 ± 1.6	2.5 ± 1.1	1.2 ± 1.0
heart	3.5 ± 2.7	1.0 ± 0.0	1.6 ± 0.8	1.7 ± 0.8	1.2 ± 0.4
hepatitis	0.7 ± 0.9	0.9 ± 0.6	2.0 ± 0.8	1.8 ± 0.9	1.2 ± 0.4
ionosphere	3.8 ± 1.9	2.2 ± 0.8	2.5 ± 1.4	2.1 ± 1.4	1.5 ± 0.8
krvskp	23.7 ± 4.2	6.2 ± 2.9	6.8 ± 2.5	3.5 ± 1.2	2.7 ± 1.5
magic	30.1 ± 16.5	19.4 ± 8.1	26.4 ± 6.1	5.3 ± 1.4	2.5 ± 0.7
monks	11.2 ± 2.4	3.5 ± 3.3	3.0 ± 0.0	4.8 ± 1.8	3.5 ± 1.4
mushroom	4.9 ± 0.3	10.9 ± 2.5	1.0 ± 0.0	1.3 ± 0.7	1.0 ± 0.0
musk2	28.5 ± 7.0	6.8 ± 3.8	16.5 ± 4.0	7.4 ± 4.0	4.7 ± 3.0
parkinsons	3.3 ± 1.7	1.2 ± 0.4	3.6 ± 1.4	1.8 ± 0.8	1.4 ± 0.5
polyadenylation	22.5 ± 18.4	2.2 ± 2.4	8.9 ± 3.6	5.5 ± 2.7	3.1 ± 2.5
pima	3.8 ± 2.6	2.2 ± 1.2	3.4 ± 2.4	2.4 ± 1.0	1.7 ± 0.8
promoters	2.0 ± 1.3	0.8 ± 0.4	1.4 ± 0.5	1.9 ± 0.7	1.5 ± 0.5
ringnorm	45.7 ± 5.2	1.3 ± 0.5	39.9 ± 7.2	2.8 ± 0.9	2.5 ± 1.2
satellite47	11.9 ± 4.5	4.0 ± 1.9	12.7 ± 4.3	5.6 ± 3.1	3.6 ± 2.3
spambase	20.3 ± 8.4	6.0 ± 2.7	5.5 ± 2.1	4.9 ± 2.0	3.1 ± 1.4
spect	5.2 ± 5.8	1.1 ± 1.6	1.4 ± 1.3	1.8 ± 0.6	1.3 ± 0.5
tictactoe	22.2 ± 6.8	6.2 ± 3.4	1.3 ± 0.5	1.2 ± 0.4	1.1 ± 0.3
titanic	4.2 ± 0.6	1.7 ± 0.5	1.1 ± 0.3	1.3 ± 0.5	1.3 ± 0.5
twonorm	79.9 ± 8.0	1.1 ± 0.3	3.1 ± 1.2	1.9 ± 0.6	2.2 ± 1.1
vote	2.9 ± 1.7	1.8 ± 0.9	1.7 ± 0.7	1.6 ± 0.5	1.4 ± 0.5

	Hard	Ldt	Soft	Soft(L_1)	Soft(L_2)
Hard		0	1	1	1
Ldt	11		4	2	0
Soft	10	2		1	0
Soft(L_1)	12	3	2		0
Soft(L_2)	13	4	4	1	

5 Conclusions

We extend the soft decision tree model by adding L_1 and L_2 regularization to penalize unnnecessary complexity. The extended model is evaluated on 27 classification data sets. We see that both versions improve accuracy slightly and decrease complexity significantly; overall L_2 regularization seems to work slightly better than L_1.

References

1. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Meteo, CA (1993)
2. Murthy, S.K., Kasif, S., Salzberg, S.: A system for induction of oblique decision trees. Journal of Artificial Intelligence Research **2** (1994) 1–32
3. Guo, H., Gelfand, S.B.: Classification trees with neural network feature extraction. IEEE Transactions on Neural Networks **3** (1992) 923–933
4. Yıldız, O.T., Alpaydın, E.: Omnivariate decision trees. IEEE Transactions on Neural Networks **12**(6) (2001) 1539–1546
5. Irsoy, O., Yildiz, O.T., Alpaydin, E.: Soft decision trees. In: Proceedings of the International Conference on Pattern Recognition, Tsukuba, Japan (2012) 1819–1822
6. Blake, C., Merz, C.: UCI repository of machine learning databases (2000)
7. Yıldız, O.T., Alpaydın, E.: Linear discriminant trees. International Journal of Pattern Recognition and Artificial Intelligence **19**(3) (2005) 323–353
8. Alpaydın, E.: Combined 5×2 cv F test for comparing supervised classification learning classifiers. Neural Computation **11** (1999) 1975–1982