



Repositorio Institucional de la Universidad Autónoma de Madrid

<https://repositorio.uam.es>

Esta es la **versión de autor** de la comunicación de congreso publicada en:
This is an **author produced version** of a paper published in:

Intelligent Data Engineering and Automated Learning – IDEAL 2006: 7th
International Conference, Burgos, Spain, September 20-23, 2006. Proceedings.
Lecture Notes in Computer Science, Volumen 4224. Springer, 2006. 995-1002.

DOI: http://dx.doi.org/10.1007/11875581_119

Copyright: © 2006 Springer-Verlag

El acceso a la versión del editor puede requerir la suscripción del recurso
Access to the published version may require subscription

Evaluation of Decision Tree Pruning with Subadditive Penalties

Sergio García-Moratilla, Gonzalo Martínez-Muñoz and Alberto Suárez

Universidad Autónoma de Madrid,
Avenida Francisco Tomás y Valiente, 11,
Madrid 28049, Spain,
`sergio.garciamoratilla@estudiante.uam.es`,
`gonzalo.martinez@uam.es`, `alberto.suarez@uam.es`

Abstract. Recent work on decision tree pruning [1] has brought to the attention of the machine learning community the fact that, in classification problems, the use of subadditive penalties in cost-complexity pruning has a stronger theoretical basis than the usual additive penalty terms. We implement cost-complexity pruning algorithms with general size-dependent penalties to confirm the results of [1]. Namely, that the family of pruned subtrees selected by pruning with a subadditive penalty of increasing strength is a subset of the family selected using additive penalties. Consequently, this family of pruned trees is unique, it is nested and it can be computed efficiently. However, in spite of the better theoretical grounding of cost-complexity pruning with subadditive penalties, we found no systematic improvements in the generalization performance of the final classification tree selected by cross-validation using subadditive penalties instead of the commonly used additive ones.

1 Introduction

Decision trees are one of the most extended types of classifiers. The reasons for their wide use are the availability of efficient algorithms for the automatic induction of decision trees from labeled data (CART [2], C4.5 [3]), the high processing speed and accuracy that can be obtained in many classification problems of practical interest, and the interpretability of the classification models generated.

A decision tree is a hierarchical questionnaire that partitions the data into disjoint subsets according to the result of tests associated to each of the non-terminal nodes of the tree. By applying the sequence of tests at the internal nodes, an example is associated to a single leaf node on the fringe of the decision tree. The classification given by the tree is the class label of the leaf node to which the example is assigned. Assuming that only Boolean tests are used, as in CART, the decision tree is a rooted binary tree. The root node has all the examples associated to it and yields as a classification the majority class of the examples in the whole training set. The binary decision tree is grown from the root node by performing a test that splits the data into two disjoint subsets.

Each of these subsets is associated to one of the two child nodes of the root node, which becomes an internal node of the tree. Each of the newly generated child nodes becomes labeled with the majority class of the training examples associated to it. The split is chosen to maximize a quantity that is correlated with the classification accuracy of the tree (for instance impurity reduction [2], information gain, information gain ratio [3], etc.) This divide-and-conquer process is repeated for each of the newly generated nodes until either all examples are correctly classified or a termination criterion is satisfied.

Trees grown with this greedy algorithm need not be globally optimal. Furthermore, if they are grown until they reach the minimum classification error on the training data, they typically exhibit poor generalization performance and yield overly optimistic estimates of the true classification error. To avoid overfitting to the training data, the tree growing process could be stopped when a properly designed termination criterion is fulfilled. However, it has proved difficult to design appropriate stopping rules [2]. Instead, the strategy that is commonly used is to overgrow the decision tree until all training examples are correctly classified, and then to prune the fully-grown tree upward, in an appropriate order, until the optimal tree is found. Biases can be avoided if the pruning process is guided by using classification error rates estimated on a validation set, an independent collection of labeled examples, which have not used in the construction of the tree [4]. While this may be an appropriate strategy for problems in which labeled training data is either abundant or easy to obtain, in data-poor problems, or when the labeling process is costly, one should avoid using separate portions of the training set for the growing and for the pruning process [5]. An alternative to this procedure is the cost-complexity pruning proposed in CART, where the goal is to minimize a function that considers both misclassification costs and a measure of the tree complexity [2].

The complexity penalty used in most cost-complexity pruning methods for decision trees, and in particular in CART, is additive. Additive penalties increase linearly with the size of the decision tree. Recent results in statistical learning theory suggest that subadditive penalties, and in particular a penalty term that varies as the square root of the size of the tree, may be more appropriate for classification problems [1, 6–9]. A subadditive penalty is monotonic but its increase with the size of the tree is slower than linear. The theoretical support for subadditive penalty terms comes from complexity regularization theory [7] and from structural risk minimization formulas that provide bounds for the generalization error of a decision tree [6]. Generally, additive penalties are used because one can design pruning algorithms that are fast, efficient in memory use, and easy to implement [1]. Little or no theoretical justification is given for the choice of a linear penalty term.

The goal of the present work is to investigate whether cost-complexity pruning with subadditive penalties, which seems to be theoretically well grounded, improves the results of other pruning strategies in a collection of benchmark problems. Previous research devoted to performing extensive comparisons between different pruning strategies [5, 10] did not consider the possibility of subad-

ditive penalties. To carry out this comparison we implement the efficient pruning algorithms designed in [1] and confirm the main results of this work. In particular we corroborate that in all cases the family of trees selected by pruning with a subadditive penalty is a subset of the nested family of trees obtained by pruning with additive penalties. Following CART, we select a single tree from the family of pruned trees using cross-validation error estimates. The performance of the selected tree is then compared with the corresponding standard CART tree, which is induced using an additive penalty.

2 Cost-complexity pruning using subadditive penalties

Consider T , a binary decision tree. Any subtree S of T containing the root node is a pruned subtree of T . This relation is denoted by $S \preceq T$. The set of terminal nodes of T is \tilde{T} and the number of terminal nodes is $|T|$.

The idea behind cost-complexity pruning is to avoid overfitting by balancing the performance on the training data and the complexity of the generated model. The performance is quantified by a cost function $\rho(S)$. In classification trees the cost function typically used is the training classification error. The complexity of the model is measured by a penalty function $\Phi(S)$, which, for decision trees, is a function of the size of the tree or, equivalently, of the number of terminal nodes. The following assumptions can be made about these functions: (i) $\rho(S)$ is a monotonically non-increasing function. That is, if $S_1 \preceq S_2$ then $\rho(S_1) \geq \rho(S_2)$, (ii) $\rho(S)$ is additive; i.e. it can be calculated by $\rho = \sum_{t \in \tilde{T}} \rho(t)$ and (iii) the penalty function $\Phi(|S|)$ is a monotonically increasing function of the tree size. That is, if $S_1 \prec S_2$ then $\Phi(|S_1|) < \Phi(|S_2|)$.

To balance the importance of $\rho(S)$ with respect to $\Phi(|S|)$ a tuning parameter α is introduced. For a given value of α the optimal tree according to the cost-complexity function is

$$T^*(\alpha) = \underset{S \preceq T}{\operatorname{argmin}} [\rho(S) + \alpha\Phi(|S|)]. \quad (1)$$

The final classification tree is selected by estimating the value of α using cross-validation. Given that $|T| < \infty$, the solutions of Eq. (1) are a finite set of pruned subtrees $R_l \preceq T$, $l = 1, \dots, m$ such that $|R_1| > |R_2| > \dots > |R_m| = 1$. Each of these trees is optimal for a range of values of α

$$\alpha \in [\alpha_{l-1}, \alpha_l) \Rightarrow T(\alpha) = R_l, \quad 0 = \alpha_0 < \alpha_1 < \dots < \alpha_m = \infty \quad (2)$$

The goal is to select the optimally pruned tree from R_l , $l = 1, \dots, m$ by determining the correct value of α . For both additive and subadditive penalties the family of pruned subtrees is unique and nested [1, 2]

$$\text{root} = R_m \prec \dots \prec R_2 \prec R_1 \preceq T. \quad (3)$$

Another important result demonstrated in [1] is that the family of subtrees obtained using subadditive penalties is a subset of the family generated using

Input: Fully developed tree T

Output: Minimum cost trees $T^k = T_1^k$, $k = 1, 2, \dots, |T|$

```

1. for  $t = 2|T| - 1$  to 1 {
2.   set  $T_t^1 = t$ 
3.   if ( $t$  is not a terminal node) {
4.     for  $k = 2$  to  $|T_t|$  {
5.       set  $mincost = \infty$ 
6.       for  $i = \max(1, k - |T_{r(t)}|)$  to  $\min(|T_{l(t), k-1}|)$  {
7.         set  $j = k - i$ 
8.         set  $cost = \rho(T_{l(t)}^i) + \rho(T_{r(t)}^j)$ 
9.         if  $cost < mincost$  {
10.          set  $mincost = cost$ 
11.          set  $T_t^k = merge(t, T_{l(t)}^i, T_{r(t)}^j)$ 
12.        }
13.      }
14.    }
15.  }
16. }
```

Fig. 1. Pseudocode for computing minimum cost trees

additive penalties. This implies that there are less trees available for selection using cross-validation. Despite having a smaller range of trees to choose from, the extra trees that appear when additive penalties are used may actually have poorer generalization performance, in which case it would be better not to consider them for selection [1].

The algorithms presented in Figs. 1 and 2 generate the family of subtrees pruned with a general size-based penalty term of increasing strength [1]. The algorithm detailed in Fig. 1 constructs the minimum cost pruned subtrees of sizes 1 to $|T|$ from the fully grown tree T . Tree nodes are indexed by numbers 1 to $2|T| - 1$ in such a way that children nodes always have a larger index than their parents. The root node has index 1. The expression T_t denotes the full tree rooted at node t (hence $T = T_1$) and T_t^k denotes the lowest cost pruned subtree of T_t of size k (i.e. $|T_t^k| = k$). The expressions $l(t)$ and $r(t)$ refer to the left child and right child of t , respectively. The algorithm given in Fig. 2 generates the family of pruned subtrees R_l and thresholds α_l of Eq. (2) from the minimum cost tree set ($T^k = T_1^k$, $k = 1, 2, \dots, |T|$) returned from the algorithm in Fig. 1.

3 Experiments

In order to compare the performance of pruning strategies using either linear (additive) or square-root (subadditive) penalties we carry out experiments in eight datasets from the UCI repository [11] and in two synthetic datasets proposed by Breiman *et al.* [2]. The datasets are selected to sample a variety of

Input: Minimum cost trees T_1^k , $k = 1, 2, \dots, |T|$

Output: Family of prunings R_l and thresholds α_l

```

1. set  $k_1 = \operatorname{argmin} (\rho(T^k) = \rho(T))$ 
2. set  $R_1 = T^{k_1}$ 
3. set  $l = 1$ 
4. while  $k_l > 1$  {
5.   set  $\alpha_l = \infty$ 
6.   for  $k = k_l - 1$  to 1 {
7.     set  $\gamma = (\rho(T^k) - \rho(T^{k_l})) / (\Phi(k_l) - \Phi(k))$ 
8.     if  $\gamma < \alpha_l$  {
9.       set  $\alpha_l = \gamma$ 
10.      set  $k_{l+1} = k$ 
11.    }
12.  }
13.  set  $l = l + 1$ 
14.  set  $R_l = T^{k_l}$ 
15. }
```

Fig. 2. Pseudocode for computing the family of pruned subtrees

problems from different fields of application. The characteristics of the selected datasets and the testing method are shown in Table 1.

The experiments consist in 100 executions for each dataset. For real-world datasets we perform a 10×10 -fold cross-validation. For the synthetic datasets (*Led24* and *Waveform*) random sampling was applied to generate each of the 100 training and testing sets. Each experiment involves the following steps:

1. Obtain the training and testing sets (by 10-fold-cv or by random sampling) and build a fully grown tree T with the training dataset using the CART tree growing algorithm [2].
2. Compute the family of pruned subtrees R_l of T and thresholds α_l , using a square-root penalty (i.e. $\Phi(|S|) = \sqrt{|S|}$) in the algorithms of Figs. 1 and 2.
3. Obtain by V-fold-cv on the training dataset V trees $(T^{(1)}, \dots, T^{(V)})$ and their respective families of pruned subtrees $R_m^{(v)}$ for $v = 1, \dots, V$. The value $V = 10$ is used. Select one subtree from each of the V families for each of the following α values: $\alpha_{geom}^l = \sqrt{\alpha_{l-1}\alpha_l}$, $l = 1, \dots, m$. For each value of α_{geom}^l calculate the error of the selected subtrees using the independent set and obtain the average cv-error (e_{cv}^l) and standard deviation (se_{cv}^l).
4. Select the pruned subtree from $(R_l, l = 1, \dots, m)$ corresponding to the α_{geom}^l value producing the smallest cv-error e_{cv}^{l*} . Denote this tree by CV-0SE. We also select the smallest tree corresponding to cv-error such that $e_{cv}^l < e_{cv}^{l*} + se_{cv}^{l*}$ and denote it by CV-1SE. Breiman *et al.* advocate the selection of CV-1SE (the 1 SE rule in [2]) because it is the simplest tree whose accuracy is comparable to CV-0SE (the optimal tree according to

Table 1. Characteristics of the datasets and testing method

Dataset	Instances	Test	Attrib.	Classes
Australian	690	10-fold-cv	14	2
Breast W.	699	10-fold-cv	9	2
Diabetes	768	10-fold-cv	8	2
German	1000	10-fold-cv	20	2
Heart	270	10-fold-cv	13	2
Ionosphere	351	10-fold-cv	34	2
Led24	200	5000 cases	24	10
New-thyroid	215	10-fold-cv	5	3
Tic-tac-toe	958	10-fold-cv	9	2
Waveform	300	5000 cases	21	3

the cross-validation procedure) when the uncertainty in the cross-validation error estimates is taken into account.

5. Repeat steps 2-4 using additive penalties $\Phi(|S|) = |S|$. This configuration results in standard CART trees.

The results of the experiments performed are in agreement with the theoretical results demonstrated in [1]. In particular, the family of pruned subtrees obtained when applying a square-root subadditive penalty term is a subset of the family of trees obtained when considering additive penalties.

Table 2 displays the average test error and average size of the selected trees for the different pruning configurations and datasets. The best average test error for each classification task is highlighted in boldface. The test errors are very similar in trees selected with either subadditive or additive penalties. The differences between CV-0SE and CV-1SE (and between these and unpruned trees) are actually larger. Another important observation is that there does not seem to be a systematic trend in the variations in performance. In some datasets the more complex trees perform better (*New-thyroid*, *Tic-tac-toe*). In other problems the minimum is obtained for medium-sized trees (*Breast W.*, *Heart*, *Led24*, *Waveform*). Finally, there are some datasets where the smaller trees are slightly better (*Australian*, *Diabetes*, *German*). The lack of a clear tendency in the results is apparent in the *Ionosphere* dataset, where the highest error corresponds to a tree of intermediate size.

The average sizes of the trees selected using subadditive and additive penalties are very similar. This can be seen in the second and third columns of Table 3, which display the average size of the families of subtrees for subadditive and additive penalties, respectively. The differences are small for most datasets. In fact, they are less than one on average for half of the studied datasets (*Breast*, *Heart*, *Ionosphere*, *New-thyroid*, *Waveform*). This indicates that the size of the final tree selected with both types of penalties is very similar. The fourth and fifth columns of Table 3 show the number of times (out of the 100 executions) in which both penalties actually selected the same final pruned subtree for CV-0SE and

Table 2. Test errors and sizes of the decision trees selected

Dataset	Unpruned		CV-0SE				CV-1SE			
	error	size	Subadd.		CART		Subadd.		CART	
	error	size	error	size	error	size	error	size	error	size
Australian	18.7	151.2	15.1	9.8	15.2	9.9	14.7	3.6	14.7	3.8
BreastW.	5.6	67.4	5.3	31.3	5.5	29.7	6.0	12.3	6.2	12.5
Diabetes	29.9	247.1	26.0	19.1	25.8	22.5	25.9	8.1	25.8	6.8
German	30.8	329.2	26.0	26.9	26.0	23.6	26.1	10.8	25.8	11.1
Heart	26.8	83.9	22.7	15.7	22.7	16.4	23.4	9.0	23.3	8.5
Ionosphere	10.4	44.3	11.3	16.0	11.4	15.5	10.8	5.9	10.7	5.7
Led24	43.6	146.3	32.7	21.7	32.9	24.7	33.7	18.6	33.6	18.6
New-thyroid	6.6	22.9	7.6	17.6	7.6	17.5	9.2	10.6	9.1	10.7
Tic-tac-toe	5.5	143.1	5.9	105.4	5.9	106.1	6.7	69.1	6.7	71.0
Waveform	29.6	78.9	28.9	31.7	29.0	28.3	30.2	16.0	30.3	15.7

CV-1SE, respectively. The fact that in most instances the same tree is selected, irrespective of the type of complexity penalty used, accounts for the similarity of the values of test errors.

Table 3. Family size and number of coincidences in the tree selected

Dataset	Average m		# same tree is selected	
	Subadd.	CART	CV-0SE	CV-1SE
Australian	10.74	13.32	79	96
Breast W.	10.01	10.54	59	70
Diabetes	11.24	16.02	64	82
German	13.52	19.52	44	65
Heart	9.47	10.19	64	71
Ionosphere	7.59	8.25	92	97
Led24	11.19	15.76	76	94
New-thyroid	5.85	6.60	99	95
Tic-tac-toe	13.40	17.22	78	75
Waveform	11.23	12.20	67	70

4 Conclusions

Despite the large body of work on decision trees, there has been little research into the problem of how to prune fully grown trees to their optimal size using complexity penalty terms. This paucity of theoretical investigations may be ascribed to the fact that pruning with additive penalties, which are commonly used

in cost-complexity pruning, can be readily and efficiently implemented. Furthermore, classification trees that are selected using cross-validation from a family of trees pruned with additive penalties seem to perform well in many problems of practical interest. Recent work on statistical learning theory for classification problems indicates that subadditive penalties may have a sounder theoretical basis than the additive penalty terms commonly used in cost-complexity pruning. In this research we implement the efficient algorithms designed in [1] to generate families of decision trees pruned with nonadditive penalty terms. The family of pruned trees is generated using a subadditive complexity penalty that increases with the square root of the size of the tree. From this family a tree is selected as the final classifier using cross-validation error estimates.

Experiments on benchmark problems from the UCI repository show that, in the datasets investigated, there is no systematic improvement of the classification performance of decision trees selected by cross-validation from a family of pruned trees induced with a square-root penalty. Since the family of trees pruned using subadditive penalties is necessarily smaller than or equal to the family pruned using additive ones, there is little room for improvement and, in fact, the decision trees selected using either a square-root penalty or linear penalty are often equal. This conclusion should also obtain for other subadditive penalties. In summary, despite the implications of recent theoretical work, we have found no evidence in the classification problems analyzed of systematic improvements in generalization performance by using subadditive penalties instead of the usual additive ones.

References

1. Scott, C.: Tree pruning with subadditive penalties. *IEEE Transactions on Signal Processing* **53**(12) (2005) 4518–4525
2. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: *Classification and Regression Trees*. Chapman & Hall, New York (1984)
3. Quinlan, J.R.: *C4.5 programs for machine learning*. Morgan Kaufmann (1993)
4. Quinlan, J.R.: Simplifying decision trees. *Int. J. Man-Mach. Stud.* **27**(3) (1987) 221–234
5. Esposito, F., Malerba, D., Semeraro, G., Kay, J.: A comparative analysis of methods for pruning decision trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(5) (1997) 476–491
6. Mansour, Y., McAllester, D.A.: Generalization bounds for decision trees. In: *COLT '00: Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2000) 69–74
7. Nobel, A.: Analysis of a complexity-based pruning scheme for classification trees. *IEEE Transactions on Information Theory* **48**(8) (2002) 2362–2368
8. C., S., R., N.: Dyadic classification trees via structural risk minimization. *Advances in Neural Information Processing Systems* 15 (2003)
9. Scott, C., Nowak, R.: Minimax-optimal classification with dyadic decision trees. *IEEE Transactions on Information Theory* **52**(4) (2006) 1335–1353
10. Mingers, J.: An empirical comparison of pruning methods for decision tree induction. *Machine Learning* **4**(2) (1989) 227–243
11. Blake, C.L., Merz, C.J.: *UCI repository of machine learning databases* (1998)