PRUNING DECISION TREES TO REDUCE TREE SIZE
LĒMUMU KOKU APGRIEŠANA TO IZMĒRU SAMAZINĀŠANAI

Ieva Bolakova
Department of Computer Science, Daugavpils University
Parades 1 – 412, Daugavpils LV 5400, Latvia
Phone: 54 25321, E-mail: ievina@dpu.lv

Abstract. *Many decision tree construction algorithms involve a two – step process: first, a very large decision tree is grown. To reduce large size and overfitting the data, in the second step, the given tree is pruned using one of several available methods. A variety of pruning methods has come to existence, and the question is which of them is best suited to solving the problem. There are investigated and compared some decision tree pruning methods in this paper.*

Keywords: *decision trees, post-pruning, pruning methods.*

1. Introduction

Decision tree learning is one of the most widely used and practical methods for inductive inference. Many variants of decision tree algorithms have been introduced in the last time and new more methods for data classification are being developed still.

Decision trees classify instances by sorting them down the tree from the root node to some leaf node, which provides the classification of the example. Each node in the tree checks the value of a single attribute. The attributes in the data set may be both symbolic – valued and numeric.

With deterministic data, an example in the training set can always be correctly classified from its known attributes. However, in many real problems there may be a degree of uncertainty present in the data. This uncertainty may arise from two different sources [1]:

1) the value of an attribute or class may be incorrectly measured or may be missing;
2) the occurrence of extraneous factors which are not recorded, but which affect the results.

When some decision tree constructing algorithm classifies such data, the resulting tree tends to be very large.

Once the decision tree has been constructed, it is easy to convert it into equivalent set of rules that are comprehensible for human beings. But if the received decision tree is very large – that means that the rule set will be large too. Unfortunately, large tree sizes do not mean better expert systems [3].

There are various tree pruning methods designed to reduce tree sizes and to increase the expected tree accuracy.

There are two different ways to apply tree pruning. The first: stop growing the tree e.g., when data split not statistically significant or too few examples are in a split. This kind of tree pruning is called *pre-pruning* or forward pruning.

The second way is called *post-pruning*: retrospectively reduce the size of a fully expanded tree by pruning some branches.

The aim of the given paper is to describe more popular and well known post-pruning methods, as well as to give a comparison of these methods by testing them on some real decision tree.

The decision tree(see fig.1.) given below is just a training example and as we can see – a partial pruning has already been made.

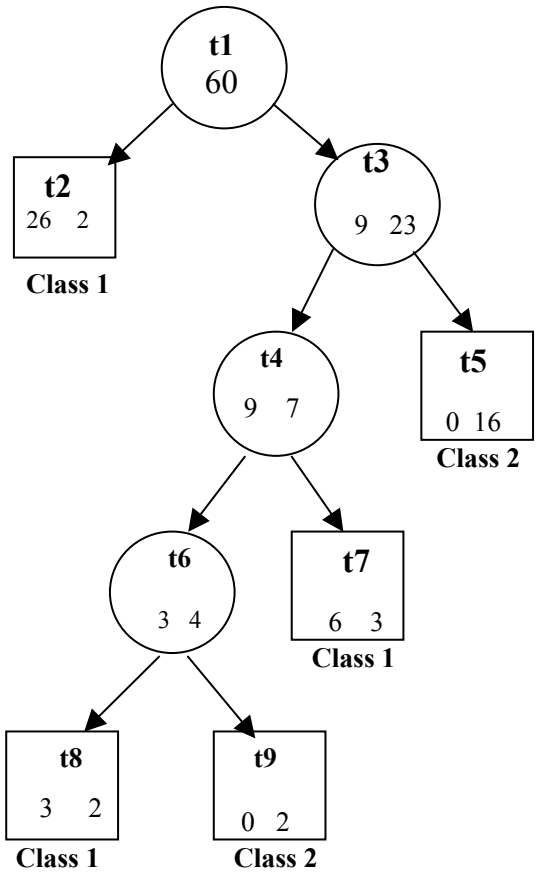

Figure 1. Example of a partially pruned decision tree

2. Cost – Complexity Pruning

This method is also known as CART (Breinman et al.) pruning algorithm. It consists of two basic steps:

1. Selection of a series of subtrees according to some heuristics.
2. Choice of the best tree T_i according to an estimate of the true error rates of these trees.

The basic idea of the first step is that T_{i+1} is obtained from T_i by pruning those branches that show the lowest increase in apparent error rate per pruned leaf.

When a tree T is pruned in a node t, its apparent error rate increases by the amount $R(t) - R(T_t)$, while its number of leaves decreases by $|N_T| - 1$ units. Thus, the following ratio

$$\alpha = \frac{R(t) - R(T_t)}{|N_T| - 1} \qquad (1)$$

measures the increase in apparent error rate per pruned leaf [4]. The algorithm calculates α for each subtree (except the first) and selects the subtree with the smallest value of α for pruning.

To illustrate that, let's consider node t4 in Figure 1.

N_T – number of leaves in the subtree T, $N_T=3$
r(t) – the error rate of node t, r(t4) = 7/16
p(t) – the proportion of data at t, p(t4) = 16/60
R(t) – the error cost of node t, if the subtree is pruned
$R(T_t)$ – the error cost for the subtree T, if the node is not pruned
i – subtree leaves

R(t4) = r(t4) * p(t4) = 7/16 * 16/60 = 7/60
$R(T_{t4})$= ΣR(i) = (2/5*5/60)+(0/2*2/60)+(3/9*9/60) = 5/60

$$\alpha = \frac{7/60 - 5/60}{3-1} = 1/60 \approx 0{,}0166$$

If we chose to prune the given tree in the node t6:

N_T=2
R(t6)= 3/7 * 7/60 = 3/60
$R(T_{t6})$= ΣR(i) = (2/5*5/60)+(0/2*2/60) = 2/60

$$\alpha = \frac{3/60 - 2/60}{2-1} = 1/60 \approx 0{,}0166$$

But now, let's see, how apparent error rate per pruned leaf will increase, if the tree is pruned in the node t3.

N_T=4
R(t3)= 9/32 * 32/60 = 9/60
$R(T_{t3})$= ΣR(i) = (2/5*5/60)+(0/2*2/60)+(3/9*9/60)+(0/16*16/60) = 5/60

$$\alpha = \frac{9/60 - 5/60}{4-1} = 4/180 \approx 0{,}0222$$

The best tree is when the subtree is pruned in the node **t4** (Figure 2), because the α is the smallest and the tree size is more reduced than if the tree is pruned in the node t6:

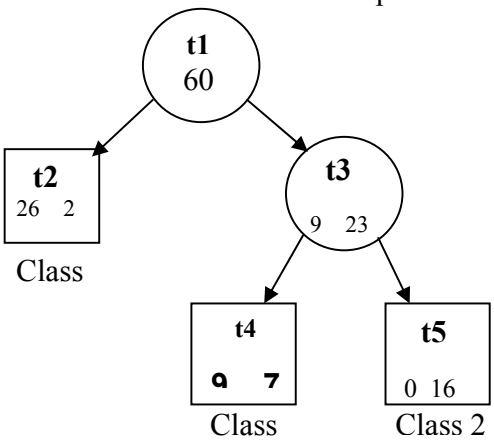

Figure 2. Pruned decision tree

3. Minimum Error Pruning

Niblett and Bratko proposed an approach seeking for a single tree that minimizes the expected error rate when classifying independent data sets [4].

The number of classes in the given set will be denoted by *k*, the total number of training examples in the node *t* will be denoted by *n(t)*, but the number of examples belonging to the dominate class *C* in the node *t* – by $n_c(t)$. Then the expected error rate is calculated by this formula:

$$E_k = \frac{n(t) - n_c(t) + k - 1}{n(t) + k} \qquad (2)$$

The method of pruning is as follows. At each non-terminal node in the tree calculate the expected error rate E_k if that subtree is pruned. Then calculate the expected error rate if the node is not pruned using the error rates for each branch, combined by weighting according to the proportion of observations along each branch. If pruning the node leads to a greater expected error rate, then keep the subtree; otherwise, prune it [3].

For the node **t6** the expected error rate will be:

* if the subtree is pruned

$$E_k = \frac{7-4+2-1}{7+2} = \frac{4}{9} \approx 0{,}4444$$

* if the subtree is not pruned

$$E_k = \frac{5}{7}\left(\frac{5-3+2-1}{5+2}\right) + \frac{2}{7}\left(\frac{2-2+2-1}{2+2}\right) \approx 0{,}3775$$

The expected error from pruning is greater, so the decision is – don't prune.

For the node **t4** the expected error rate will be:

* if the subtree is pruned

$$E_k = \frac{16-9+2-1}{16+2} = \frac{8}{18} \approx 0{,}4444$$

* if the subtree is not pruned

$$E_k = \frac{7}{16} * 0{,}3775 + \frac{9}{16}\left(\frac{9-6+2-1}{9+2}\right) \approx 0{,}3697$$

The expected error from pruning is greater again, so the decision is – do not prune.

One of the disadvantages of this method is the dependence of the expected error rate on the number of classes.

4. Pessimistic Error Pruning

This pruning method, proposed by Quinlan and used currently in C4.5, aims to avoid the necessity of a separate test data set [3]. Pessimistic pruning is based on the number of errors and the size of the training sample.

If N(t) is the number of training set examples at node t, and e(t) is the number of examples mis-classified at node t, then an estimate of the mis-classification rate is

$$r(t) = \frac{e(t)}{N(t)}. \tag{3}$$

The rate with the continuity correction is

$$r'(t) = \frac{e(t)+1/2}{N(t)}. \tag{4}$$

Accordingly, for a subtree T_t the mis-classification rate will be

$$r(T_t) = \frac{\sum e(i)}{\sum N(i)} \tag{5}$$

where i covers the leaves of the subtree. Thus the corrected mis-classification rate will be

$$r'(T_t) = \frac{\sum (e(i)+1/2)}{\sum N(i)} = \frac{\sum e(i) + N_T/2}{\sum N(i)} \tag{6}$$

where N_T is the number of leaves [3].

With the training data, the subtree will always make fewer errors than the corresponding node, but this is not so when the corrected figures are used, since they depend on the number of

leaves, not just the number of errors. The algorithm only keeps the subtree if its corrected figure is more than one standard error better than the figure for the node [3].

The standard error is calculated in this way:

$$SE(n'(T_t)) = \sqrt{\frac{n'(T_t) * (N(t) - n'(T_t))}{N(t)}} \quad (7)$$

where n'(t) = e(t) + ½ for a node and

$n'(T_t) = \Sigma e(i) + N_T/2$ for a subtree.

So this pruning method suggests pruning the subtree if its corrected number of mis-classifications is greater than that for the node.

For example, the number of corrected mis-classifications at node t4 is

n'(t4) = 7 +1/2 = 7.5,

and the number of corrected mis-classifications for subtree is

$n'(T_{t4}) = (2+0+3) + 3/2 = 6.5$.

$$SE = \sqrt{\frac{6{,}5 * (16 - 6{,}5)}{16}} \approx 1{,}96 .$$

Since 6.5 + 1.96 = 8.46, which is greater than 7.5, the subtree will be pruned.

This method have some advantages: the same training set is used for both growing and pruning a tree; and it is very quick because it only has to make one pass and looks at each node only once [3].

5. Critical Value Pruning

Critical Value Pruning method, proposed by Mingers, consists of two basic steps [4]:

1) prune T_{max} for increasing critical values;
2) choose the best tree among the sequence of pruned trees, by measuring the significance of the tree as a whole and its predictive ability.

Although this method is post-pruning method, it is very similar to a pre-pruning technique. In creating the original tree, a goodness of split measure determines the attribute at a node. The value of this measure reflects how well the attribute is chosen.

In practice, for each node this method computes the maximum 'information - gain' in the subtree, and prunes a node if this value is less than a certain threshold [2].

6. Reduced Error Pruning

This is another method proposed by Quinlan, which uses an independent sample to test the accuracy of each subtree compared to the case when it is pruned.

The method is as follows: start with a complete tree and run the test data through it. For each non-terminal node, count the number of errors if the subtree is kept and the number if it becomes a leaf through pruning. The pruned node will often make fewer errors on the test data than the subtree makes. The difference between the numbers of errors is a measure of the gain from pruning the subtree. From all the nodes, choose the one with the largest difference as the subtree to prune [3].

The positive property of this method is that each node is visited only once to evaluate the opportunity of pruning it.

7. Conclusion

The decision tree that we could see in Figure 1 is acquired by using CART algorithm. The tree was tested on 60 examples data set after that was growing. That decision tree was used to review several decision tree pruning methods too.

The decision tree pruning is regarded as a logical continuation after its construction because of two important reasons: to reduce the decision tree size and to increase its accuracy.

There are three pruning methods considered more particularly in this paper: Cost – Complexity Pruning, Minimum Error Pruning and Pessimistic Error Pruning. Two methods (Cost – Complexity Pruning and Pessimistic Error Pruning) gave the same advice – prune the given tree in the node t4, but Minimum Error Pruning method advise to keep this subtree.

In the literature concerning decision tree pruning the results attained are different too: some methods gave more accurate decision trees; some methods gave trees with smaller sizes. It is conclused that pruning can improve the accuracy of induced decision trees by up to 25% [1].

Kopsavilkums

Lēmumu kokus uzskata par praktiskāko un vienkāršāko datu klasificēšanas veidu. Mūsdienās ir pieejami daudz un dažādi lēmumu koku konstruēšanas algoritmi, no kuriem var izvēlēties vispiemērotāko konkrēta uzdevuma risināšanai. Kad attiecīgais lēmumu koks ir iegūts, no tā var izveidot likumu kopu. Likumi vienkāršam lietotājam ir vieglāk saprotami. Taču, ja lēmumu koks ir ļoti liels – arī likumu kopa būs apjomīga izmēra. Tas savukārt nepavisam neatvieglos cilvēka darbu ar šādu kopu. Lai novērstu šo problēmu, t.i., lai samazinātu lēmumu koku izmērus un lai paaugstinātu to precizitāti, tiek piedāvātas dažādas lēmumu koku apgriešanas metodes.

Šajā referātā detalizētāk aplūkotas trīs patlaban populārākās lēmumu koku apgriešanas metodes, kā arī sniegts vēl divu metožu vispārīgs apraksts.

Lēmumu koku apgriešanas algoritmus var sadalīt divās lielākās grupās:

1) metodes, kuras izmanto lēmumu koka konstruēšanas laikā – tas ir: katrā koka neterminālā mezglā tiek izskatīts jautājums, vai ir/nav vērts sadalīt tālāk šo mezglu;
2) metodes, kuras pielieto koka apgriešanai pēc tā pilnīgas ‘uzaudzēšanas’, konstruēšanas.

Apgriešana, izmantojot kļūdas sarežģītību

Šo algoritmu piedāvājis L.Breimans un to galvenokārt izmanto konstruēšanas algoritmā CART. Tā darbību var sadalīt divos soļos:

1) tiek izveidota virkne apgrieztu koku, kuri atšķiras viens no otra ar apgriezto mezglu skaitu;
2) no virknes izvēlas labāko koku, veicot testus uz neatkarīgas datu kopas.

Apgriešana, izmantojot kļūdas minimumu

T.Nibletts un I.Bratko piedāvāja metodi tāda koka meklēšanai, kurš pieļautu mazāko kļūdu neatkarīgo datu klasificēšanā.

Apgriešanas algoritms darbojas sekojoši: katrā neterminālā koka mezglā tiek skaitļota kļūdas vērtība E_k gadījumam, ja apakškoku apgrieztu. Tad skaitļo kļūdas vērtību gadījumam, ja apakškoku neapgriež, atrodot kļūdu katram zaram un pielabojot to, sareizinot ar piemēru skaitu dotajā zarā. Ja, apgriežot koku apskatītajā mezglā, kļūda ir lielāka nekā kļūda apakškokam, tad lēmums ir – negriezt apakškoku.

Šajā metodē aprēķinos tiek ņemts vērā klašu skaits *k* datu kopā.

Apgriešana, izmantojot pesimistisko kļūdu

Šīs metodes autors ir J.R.Kvinlans. Tās mērķis ir izvairīties no nepieciešamības izmantot atsevišķu testa datu kopu – viena un tā pati apmācošā datu kopa tiek izmantota gan koka audzēšanai, gan, pēc tam, tā apgriešanai. Taču ir skaidrs, ka kļūdas vērtība, kuru aprēķina uz apmācošās datu kopas, būs neobjektīva, lai izvēlētos labāko apgriezto lēmumu koku. Tāpēc autors šajā algoritmā piedāvā skaitļot ‘reālāku kļūdas vērtību’ [4], kur sauc par standarta kļūdu.

Algoritma aprēķini balstās uz apmācošās datu kopas izmēru.

Pesimistiskās kļūdas algoritms tiek uzskatīts par ātrdarbīgu apgriešanas metodi, jo tajā katrs mezgls tiek aplūkots tikai vienu reizi.

Apgriešana, izmantojot kritisko vērtību

Algoritma autors ir J.Mingers. Tā pamatā ir divi soļi:

1) tiek izveidota apgrieztu koku virkne, izmantojot kritisko vērtību palielināšanos;
2) izvēlas labāko apgriezto koku, līdzīgi kā L.Breimana piedāvātajā algoritmā.

Apgriešana, izmantojot kļūdas samazināšanos

Arī šo algoritmu piedāvājis J.R.Kvinlans. Tas darbojas sekojoši: testa dati tiek pārbaudīti neapgrieztā kokā, nosakot katras klases parādīšanos katrā koka mezglā. Katram neterminālam mezglam aprēķina kļūdu skaitu, ja apkškoku

saglabā, un kļūdu skaitu, ja zaru nogrieztu. Starpība starp šīm kļūdām parāda, cik lielā mērā uzlabojas koks, ja apakškoku nogriež. No visiem mezgliem apgriešanai izvēlas to, kurā šī starpība ir vislielākā.

Lēmumu koku apgriešana viennozīmīgi uzskatāma par to konstruēšanas procesa loģisku turpinājumu. Koku apgriešana paaugstina to precizitāti, kā arī samazina izmērus, kas konkrētās situācijās var ietaupīt gan laiku, gan naudu.

Taču, pārbaudot praktiski uz mācību piemēra trīs no piecām aplūkotajām koku apgriešanas metodēm, tika secināts, ka šāda veida lēmumu koku apgriešana vairāk ir kvantitatīva darbība. Tā it kā izslēdz gadījuma datu/faktu ietekmi uz lēmuma pieņemšanu. Taču vai tas atbilst konkrēta lietotāja vajadzībām? Un vai vienmēr, visās jomās mēs varam atļauties tādā veidā ierobežot zināšanu atspoguļošanu? Iespējams, dažās situācijās šo aspektu ievērošana ir svarīgāka nekā kvantitatīvu uzlabojumu veikšana.

Bibliography

1. Mingers J. An Empirical Comparison of Pruning Methods for Decision Tree Induction. Machine Learning, 4, 1989, P. 227-243.
2. Mansour Y. Pessimistic decision tree pruning based on tree size. Proc. 14th International Conference on Machine Learning (1997); http://citeseer.nj.nec.com/mansour97pessimistic.html
3. Ignizio J.P. An Introduction to Expert Systems: The Development and Implementation of Rule Based Expert Systems. – McGraw – Hill, Inc., 1991, 402 p.
4. Esposito F., Malerba D., Semeraro G. A Comparative Analysis of Methods for Pruning Decision Trees. IEEE Transactions on Pattern Analysis and Machine Intelligence, VOL. 19, NO. 5, 1997, P. 476-491