Learning Characteristic Decision Trees

Paul Davidsson
Department of Computer Science, Lund University
Box 118, S–221 00 Lund, Sweden
E-mail: Paul.Davidsson@dna.lth.se

Abstract

Decision trees constructed by ID3-like algorithms suffer from an inability of detecting instances of categories not present in the set of training examples, i.e., they are discriminative representations. Instead, such instances are assigned to one of the classes actually present in the training set, resulting in undesired misclassifications. Two methods of reducing this problem by learning characteristic representations are presented. The central idea behind both methods is to augment each leaf of the decision tree with a subtree containing additional information concerning each feature's values in that leaf. This is done by computing two limits (lower and upper) for every feature from the training instances belonging to the leaf. A subtree is then constructed from these limits that tests every feature; if the value is below the lower limit or above the upper limit for some feature, the instance will be rejected, i.e., regarded as belonging to a novel class. This subtree is then appended to the leaf. The first method presented corresponds to creating a maximum specific description, whereas the second is a novel method that makes use of the information about the statistical distribution of the feature values that can be extracted from the training examples. An important property of the novel method is that the degree of generalization can be controlled. The methods are evaluated empirically in two different domains, the Iris classification problem and a novel coin classification problem. It is concluded that the dynamical properties of the second method makes it preferable in most applications. Finally, we argue that this method is very general in that it, in principle, can be applied to any empirical learning algorithm.

1 Introduction

One of the often ignored problems for a learning system is to know when it encounters an instance of an unknown category. In theoretical contexts this problem is often regarded as being of minor importance, mainly because it is assumed that the problem domains under study are closed (i.e., all relevant information is known in advanced). However, in many practical applications it cannot be assumed that every category is represented in the set of training examples (i.e., they are open domains) and sometimes the cost of a misclassification is too high. What is needed in such situations is the ability to reject instances of categories that the system has not been trained on. For example, consider the decision mechanism in a coin-sorting machine of the kind often used in bank offices. Its task is to sort and count a limited number of different coins (for instance, a particular country's), and to reject all other coins. Supposing that this decision mechanism is to be learned, it is for practical reasons impossible to train the learning system on every possible kind of coin, genuine or faked. Rather, it is desired that the system should be trained only on the kinds of coins it is supposed to accept. Another example are decision support systems, for

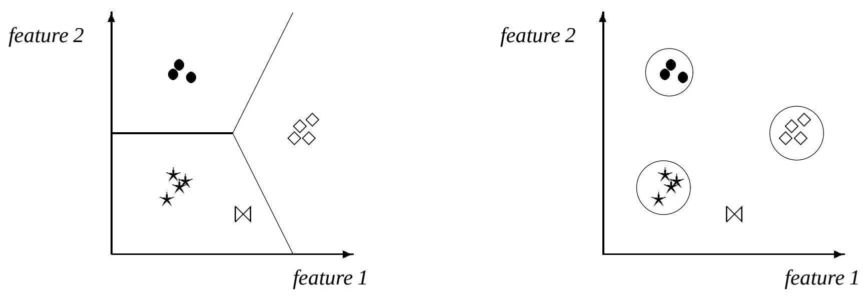

Figure 1: Discriminative versus characteristic category descriptions.

instance in medical diagnosis, where the cost of a misclassification often is very high — it is better to remain silent than to give an incorrect diagnosis.

As been pointed out by Smyth and Mellstrom [5], the only way of solving this problem is to learn *characteristic* category descriptions that try to capture the similarities between the members of the category. This, in contrast to learning *discriminative* descriptions that can be seen as representations of the boundaries between categories. The difference between these kinds of descriptions is illustrated in Figure 1. It shows some instances of three known categories ($\star$, $\bullet$, and $\diamond$), and examples of possible category boundaries of the concepts learned by a system using discriminative descriptions (to the left) and by a system using characteristic descriptions (to the right). In this case, a member of an unknown category ($\bowtie$) will be categorized wrongly by a system using discriminative descriptions, whereas it will be regarded as a member of a novel category by a system using characteristic descriptions. In other words, whereas systems that learn discriminative descriptions tend to overgeneralize, the degree of generalization can be controlled, or at least limited, by systems learning characteristic descriptions.

Decision trees constructed by ID3-like algorithms [4], as well as, for instance, nearest neighbor algorithms and neural networks learned by back-propagation,[1] suffer from this inability of detecting examples of categories not present in the training set. Of course, there exist methods for learning characteristic descriptions from examples (with numerical features), for instance, the algorithm presented by Smyth and Mellstrom [5], some neural nets (e.g., ART-MAP), and certain kinds of instance-based methods. The problem with these is that they do not learn explicit rules, which is desired in many practical applications such as the coin classification task outlined earlier. However, as has been shown by Holte el al. [3], the CN2 algorithm can be modified to learn characteristic descriptions in the form of rule-based *maximum specific descriptions*.

In the next section two methods of learning characteristic decision trees will be presented. The first method is a straightforward adaption of the idea of maximum specific descriptions as described by Holte et al. to the decision tree domain. The second method is a novel approach that make use of the information about the statistical distribution of the feature values that can be extracted from the training examples. These methods are then evaluated empirically in two different domains, the classic Iris classification problem and the coin classification problem described above. In the last section there is a general discussion of the suggested methods.

[1] At least, in the original versions of these algorithms.

2 The Methods

Both methods are based on the idea of augmenting each leaf of the tree resulting from a decision tree algorithm with a subtree. The purpose of these subtrees is to impose further restrictions on the feature values. A lower and a upper limit are computed for every feature. These will serve as tests: if the feature value of the instance to be classified is below the lower limit or above the upper limit for one or more of the features, the instance will be rejected, i.e., regarded as belonging to a novel class, otherwise it will be classified according to the original decision tree. Thus, when a new instance is to be classified, the decision tree is first applied as usual, and then, when a leaf would have been reached, every feature of the instance is checked to see if it belongs to the interval defined by the lower and the upper limit. If all features of the new instance are inside their interval the classification is still valid, otherwise the instance will be rejected.

In the first method we compute the minimum and maximum feature value from the training instances of the leaf and let these be the lower and upper limits respectively. This approach will yield a maximum specific description (cf. the modification of CN2 by Holte et al. [3]).

While being intuitive and straight-forward, this method is also rather static in the sense that there is no way of controlling the values of the limits, i.e., the degree of generalization. Such an ability would be desirable, for example, when some instances that would have been correctly classified by the original decision tree are rejected by the augmented tree (which happens if any of its feature values is on the wrong side of a limit). Actually, there is a trade-off between the number of failures of this kind and the number of misclassified instances. How it should be balanced is, of course, dependent of the application (i.e., the costs of misclassification and rejection). Since it is impossible in the above method to balance this trade-off, a more dynamic method in which it can be controlled has been developed.

The central idea of this novel method is to make use of statistical information concerning the distribution of the feature values of the instances in a leaf. For every feature we compute the lower and the upper limits so that the probability that a particular feature value (of an instance belonging to this leaf) belongs to the interval between these limits is $1 - \alpha$. In this way we can control the degree of generalization and, consequently, the above mentioned trade-off by choosing an appropriate α-value. The lesser the α-value is, the more misclassified and less rejected instances. Thus, if it is important not to misclassify instances and a high number of rejected (not classified) instances are acceptable, a high α-value should be selected.

It turns out that only very simple statistical methods are needed to compute such limits. Assuming that X is a normally distributed stochastic variable, we have that:

$$P(m - \lambda_{\frac{\alpha}{2}}\sigma < x < m + \lambda_{\frac{\alpha}{2}}\sigma) = 1 - \alpha$$

where m is the mean, σ is the standard deviation, and λ is a critical value depending on α (for instance $\lambda_{0.025} = 1.960$). Thus, we have, for instance, that the probability of an observation being larger than $m - 1.96\sigma$ and smaller than $m + 1.96\sigma$ is 95%.

In order to follow this line of argument we have to assume that the feature values of each category (or each leaf if it is a disjunctive concept) are normally distributed. This assumption seems not too strong for most applications. However, as we cannot assume that the actual values of m and σ are known, they have to be estimated. A simple way of doing this is to compute the mean and the standard deviation of the training instances $(x_1, ..., x_n)$ belonging to the leaf:

$$m^* = \frac{\Sigma x_i}{n}, \quad \sigma^* = \sqrt{\frac{\Sigma(x_i - \overline{x})^2}{n-1}}$$

To get a nice interpretation of the interval between the upper and lower limit, we have to assume that these estimates are equal to the actual values of m and σ. This is, of course, too optimistic, but it seems reasonable to believe (as will be shown in Section 3) that the method is of practical value also without this interpretation. Anyway, the intended statistical interpretation suggests that the probability of a feature of an instance of a category being larger than lower limit and smaller than upper limit for $\alpha = 0.01$ is 99%. (It can be argued that this is a rather crude way of computing the limits. A more elaborate approach would be to compute confidence intervals for the limits and use these instead. This was actually the initial idea but it turned out that this only complicates the algorithm and does not increase the classification performance significantly.)

3 Empirical Evaluation

As we here are interested in the behaviour of the algorithms when confronted with unknown categories, not all of the categories present in the data sets were used in the training phase. This approach may at first sight seem somewhat strange as we actually know that there are, for instance, three categories of Irises in the data set. But, how can we be sure that there only exist three categories? It might exist some not yet discovered species of Iris. In fact, we believe that in most real world applications it is not reasonable to assume that all relevant categories are known and can be given to the learning system in the training phase.

In the experiments described below N–1 classes were used for training and N classes for testing. Due to shortage of space, however, only the results of one choice of categories in the training set will be presented here. The results of the remaining combinations, together with results from experiments with other data sets, can be found in [2]. Moreover, in these experiments we have used a basic ID3 algorithm [4] for computing the initial decision tree.

3.1 The Iris Database

The classic Iris database contains 3 categories of 50 instances each, where a category refers to a type of Iris plant (Setosa, Versicolor or Virginica). All of the 4 attributes (sepal length, sepal width, petal length, and petal width) are numerical. In each experiment the data set was randomly divided in half, with one set used for training and the other for testing. Thus, 50 (2$\times$25) instances were used for training and 75 (3$\times$25) for testing. Each experiment was performed with the basic ID3 algorithm, the maximum specific tree algorithm (ID3-Max), and the algorithm based on statistical distribution (ID3-SD) for the α-values: 0.2, 0.1, 0.05, and 0.01.

Table 1 shows the classification results when the algorithms were trained on instances of Iris Setosa and Iris Virginica (which is the most difficult case). ID3 misclassifies, of course, all the instances of Iris Versicolor, but more interesting is that ID3-SD (for α = 0.1) performs significantly better than the ID3-Max algorithm. It has a slightly higher rejection-rate, but misclassifies over 60% less instances than ID3-Max. We can also see that by varying the α-value it is possible to control the trade-off between the number of rejected and misclassified instances. It is possible to achieve almost zero misclassifications if we choose α = 0.2, but then we get a rejection rate of over 50% also for the two known categories. In fact, also the number of misclassifications

	Iris Setosa			Iris Versicolor			Iris Virginica		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	98.8	1.2	0.0	0.0	100.0	0.0	99.2	0.8	0.0
ID3-Max	68.8	0.0	31.2	0.0	14.8	85.2	74.0	0.0	26.0
ID3-SD 0.2	42.4	0.0	57.6	0.0	1.2	98.8	49.6	0.0	50.4
ID3-SD 0.1	62.4	0.0	37.6	0.0	5.6	94.4	70.4	0.0	29.6
ID3-SD 0.05	74.0	0.0	26.0	0.0	17.6	82.4	80.0	0.0	20.0
ID3-SD 0.01	84.0	0.0	16.0	0.0	47.2	52.8	91.2	0.0	8.8
desired	100.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0

Table 1: Results from training set containing instances of Iris Setosa and Iris Virginica (averages in percentages over 10 runs).

of known categories is reduced by the algorithms learning characteristic descriptions. The decision trees induced by ID3 misclassifies 1.2% of the Iris Setosa and 0.8% of the Iris Virginica instances, whereas both ID3-Max and ID3-SD induce trees that do not misclassify any of these instances. The main reason for this seems to be that the characteristic decision trees check all features so that they do not take unreasonable values, whereas the discriminative trees only check one or two of the features. Finally, it is interesting and somewhat surprising how difficult this classification problem is, in particular since it in its original formulation (where instances of all three categories are given to the learner) is regarded as almost trivial.

3.2 Coin Classification

This task corresponds to the problem of learning the decision mechanism in coin sorting machines described in the introduction. In the experiments two databases were used, one describing Canadian coins contains 7 categories (1, 5, 10, 25, 50 cent, 1 and 2 dollar), and one describing Hong Kong coins that also contains 7 categories (5, 10, 20, 50 cent, 1, 2, and 5 dollar). All of the 5 attributes (diameter, thickness, conductivity1, conductivity2, and permeability) are numerical. The Canada and Hong Kong databases were chosen because when using the manufacturer's current method for creating the rules of the decision mechanism (which is manual to a large extent), these coins have been causing problems. In each experiment 140 (7×20) instances were randomly chosen for training and 700 ($2 \times 7 \times 50$) for testing. Each experiment was performed with the following α-values: 0.05, 0.01, 0.001, and 0.0001.

Table 2 shows the classification results when training on the Hong Kong coin database (the most difficult case). To begin with, we can see that all foreign coins (i.e., the Canadian coins) are rejected, except of course for the ID3 algorithm. However, there were some problems with misclassifications. In this particular application there are some demands that must be met by the learning system before it can be used in reality, namely, less than 5% rejects of known types of coins and very few misclassifications (not more than 0.5%). In our experiment, these requirements are met only by the ID3-SD algorithm with α = 0.001 and 0.0001, which illustrates the advantage of being able to control the degree of generalization.

	Hong Kong Coins			Foreign Coins		
	correct	miss	reject	correct	miss	reject
ID3	98.3	1.7	0.0	0.0	100.0	0.0
ID3-Max	79.7	0.0	20.3	0.0	0.0	100.0
ID3-SD 0.05	74.8	0.0	25.2	0.0	0.0	100.0
ID3-SD 0.01	88.9	0.0	11.1	0.0	0.0	100.0
ID3-SD 0.001	95.1	0.3	4.6	0.0	0.0	100.0
ID3-SD 0.0001	96.3	0.5	3.2	0.0	0.0	100.0
desired	100.0	0.0	0.0	0.0	0.0	100.0

Table 2: Results from training set containing Hong Kong coins (averages in percentages over 10 runs).

4 Discussion

The rationale behind the methods presented in this paper was to combine the obvious advantages of characteristic representations with the classification efficiency, explicitness, and simplicity of decision trees. Of the two methods presented, the maximum specific description method (ID3-Max) seems to work well in some domains, but often the method based on statistical distribution (ID3-SD) gives significantly better results. The main reasons for this seem to be that it is more robust than the former and that it is possible to control the degree of generalization, which leads to another advantage of the statistical approach, namely, that the trade-off between the number of rejections and misclassifications can be balanced in accordance to the constraints of the application. In some applications the cost of a misclassification is very high and rejections are desirable in uncertain cases, whereas in others the number of rejected instances are to be kept low and a small number of misclassifications are accepted.

The expansion of the ID3 algorithm to ID3-SD was carried out using simple statistical methods. If n is the number of training instances and m is the number of features, the algorithmic complexity of the computations associated with the limits is linear in the product of these (i.e., $O(nm)$) in the learning phase (which can be neglected when compared to the cost of computing the original decision tree), and linear in m (i.e., $O(m)$) in the classification phase.

The main limitation of the SD-method seems to be that it is only applicable to numerical attributes. The maximum description method, on the other hand, requires only that the features can be ordered. Thus, one way of making the former method more general is to combine it with the latter method to form a hybrid approach that is able to handle all kinds of ordered features. We would then use the statistical method for numerical attributes and the maximum description method for the rest of the attributes. Moreover, nominal attributes could be handled by accepting those values present among the instances of the leaf and reject those that are not. In this way we get a method that learns characteristic descriptions using all kinds of attributes.

The original ID3-algorithm is quite good at handling the problem of irrelevant features (i.e., only features that are useful for discriminating between the categories in the training set are selected). But since the suggested methods compute upper and lower limits for every feature and

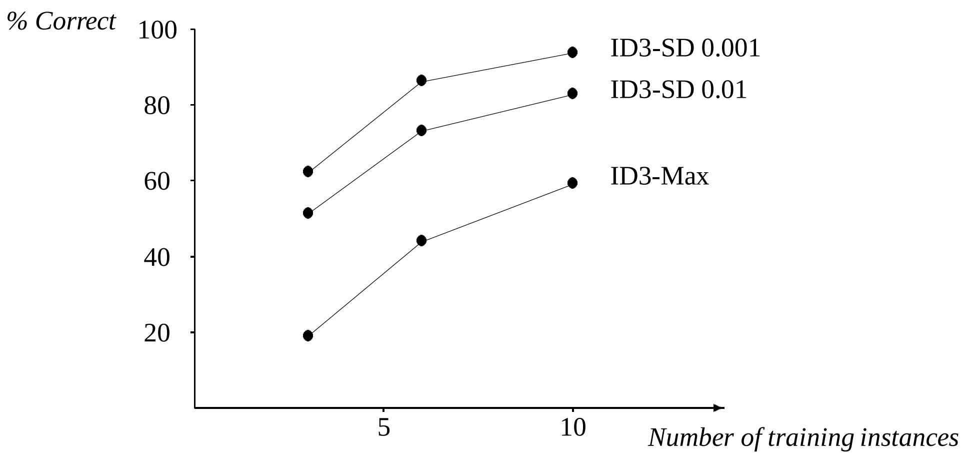

Figure 2: The percentage of correctly classified instances of known categories as a function of the number of instances of each category in small training sets (averages over 10 runs). The remaining instances were rejected.

use these in the classification phase, also the irrelevant features will be subject for consideration. However, this potential problem will typically disappear when using the statistically based method for the following reason. An irrelevant feature is often defined as a feature which value is randomly selected according to a uniform distribution on the feature's value range. (cf. Aha [1]) That is, the feature values have a large standard deviation, which will lead to a large gap between the lower and the upper limit. Thus, as almost all values of this feature will be inside this interval, the feature will still be irrelevant for the classification. Some experimental support for the claim that ID3-SD handles irrelevant features well is presented in [2].

Another potential problem for ID3-SD is the problem of few training instances. One would think that when the number of training examples of a category decreases there is a risk that the estimates of the mean value and the standard deviation will not be sufficiently good. However, preliminary experiments in the coin classification domain indicates that the classification performance decreases only slowly when the training examples get fewer. As can be seen in Figure 2, it handles the problem of few training instances better than the maximum specific description which, in fact, has been suggested as a solution to the related problem of small disjuncts (cf. Holte et al. [3]).

Finally, another problem arises when the number of features is large. If we choose α = 0.01 and have 20 features, the probability that every feature value of an instance is between the the lower and the upper limits is just 81.8% resulting in too many undesired rejected instances. However, a simple solution to this problem is to determine the α-value out of a desired total probability, P_{tot} (we have that $(1-\alpha)^n = P_{tot}$). For example, if there are 20 features and we want a total probability of 95%, we should choose α = 0.0025.

4.1 Noisy Data and The Generality of the SD-approach

ID3-SD is better at handling noisy data than ID3-Max in the sense that an extreme feature value for one (or a few) instance(s) will not influence the positions of the limits of that feature in ID3-

SD as much as it will in ID3-Max. A method, not yet evaluated, for further reducing the problem of noisy instances, would be to use the limits to remove the instances of the leaf that have feature values that are (considerably) lower than the lower or (considerably) higher than the higher limit, and then recalculate the limits. However, in this paper we have used ID3 as a basis, an algorithm that is not very good at handling noisy data in the first place. In fact, there is a trivial solution to the problem with noisy data: Use any noise tolerant algorithm for inducing decision trees (e.g., C4.5) and then compute the subtrees as before for the remaining leaves.

Thus, the statistically based approach for creating characteristic descriptions is a general method in the sense that we can take the output from any decision tree induction algorithm, compute a subtree for every leaf, and append them to their leaf. In fact, the approach can, in principle, be applied to any empirical learning method. However, if the instances of a category corresponds to more than one cluster in the feature space (cf. disjunctive concepts), the method will work better for algorithms that explicitly separates the clusters, i.e., where it is possible to find out which cluster a particular instance belongs to. If this is the case, the limits can be computed separately for each cluster. Otherwise, we must compute only one lower and upper limit for the whole category, which probably will result in a too large gap between the lower and the upper limit.

The procedure for augmenting an arbitrary empirical learning algorithm X is as follows: train X as usual, then compute the limits for every category (i.e., cluster) in the training set as described earlier. When a new instance is to be classified, first apply X's classification mechanism in the same way as usual, then check that all features values of the new instance are larger than the lower limit and smaller than the upper limit. Thus, it is not necessary to represent the limits in the form of decision trees, the main point is that there should be a method for comparing the feature values of the instance to be classified with the limits. Future work will evaluate how different empirical learning methods can be improved in this way. In this perspective, we have in this paper only described an application of the general method to the ID3 algorithm. Moreover, this is the main reason why we have not compared ID3-SD to other kinds of algorithms that learn characteristic descriptions.

References

[1] D.W. Aha. Generalizing from case studies: A case study. In *Ninth International Workshop on Machine Learning*, pages 1–10. Morgan Kaufmann, 1992.

[2] P. Davidsson. ID3-SD: An algorithm for learning characteristic decision trees by controlling the degree of generalization. Technical Report LU–CS–TR: 95–145, Dept. of Computer Science, Lund University, Lund, Sweden, 1995.

[3] R.C. Holte, L.E. Acker, and B.W. Porter. Concept learning and the problem of small disjuncts. In *IJCAI-89*, pages 813–818, 1989.

[4] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[5] P. Smyth and J. Mellstrom. Detecting novel classes with applications to fault diagnosis. In *Ninth International Workshop on Machine Learning*, pages 416–425. Morgan Kaufmann, 1992.