ID3-SD: An Algorithm for Learning Characteristic Decision Trees by Controlling the Degree of Generalization

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. In this report, 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 accomplished 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 called ID3-SD 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 gap between the limits (i.e., the degree of generalization) can be controlled. The two methods are then evaluated empirically in three different domains: the classic Iris database, a wine database, and a novel database from a coin-sorting machine. It is concluded that the dynamical properties of the ID3-SD method makes it preferable in many applications. Finally, we argue that this method in fact is general in that it, in principle, can be applied to any empirical learning algorithm, i.e., it is not restricted to decision tree algorithms.

1 Introduction

One often ignored problem for a learning system is how to know when it encounters an instance of an unknown category. In many practical applications it cannot be assumed that every category is represented in the set of training examples[1] 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 instance in medical diagnosis, where the

[1] That is, they are open domains (cf. Hutchinson [8]).

cost of a misclassification often is very high — it is better to remain silent than to give an incorrect diagnosis. Moreover, this is a critical problem in autonomous learning systems, as such systems must be able to decide when to create a new concept (cf. Davidsson [3]).[2]

As been claimed by Smyth and Mellstrom [13], 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 cat-

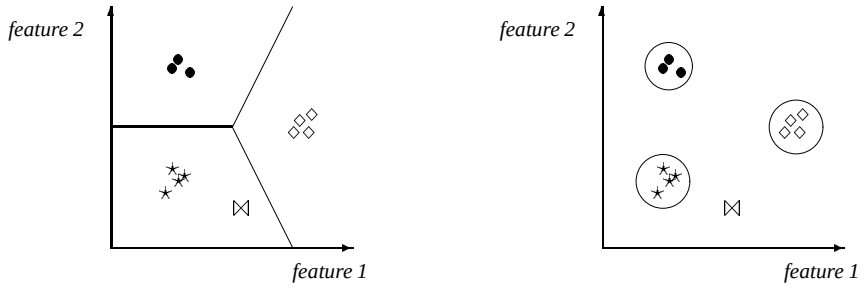


Figure 1: Discriminative versus characteristic category descriptions.

egory 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 [10], as well as, for instance, nearest neighbor algorithms and neural networks learned by back-propagation,[3] 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 [13], ART-MAP [2], 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. [7], CN2 (a relative to the AQ-family) can be modified to learn characteristic descriptions in the form of rule-based *maximum specific descriptions*.

In the next sections two methods of learning characteristic decision trees will be presented. The first method is a straightforward adaption of the idea of maximum specific descriptions to the decision tree domain. The second method, which is an improved version of the method outlined by Davidsson [4], is a novel approach that makes 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 three different domains: the classic Iris classification problem, a wine recognition problem, and the coin classification problem described above. In the last section there is a general discussion of the suggested methods.

[2] In short, if the system believes that the current observation belongs to an (for the agent) unknown category, it seems rational to create a new category based on this instance.

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

2 Method I – Maximum Specific Description

Both methods are based on the idea of augmenting each leaf of the decision tree resulting from the original ID3 algorithm with a subtree. The purpose of these subtrees is to impose further restrictions on the feature values. A lower and an upper limit are computed for every feature. These will serve as tests in the following way: 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. [7]).

3 Method II – Statistical Distribution

While being intuitive and straight-forward, the method described above 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. An ability to do this is 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). This is to avoid under-generalization, a similar trade-off has to be balanced to avoid over-generalization, i.e., when some instances of unknown categories are not rejected. Since it is impossible in the above method to balance these trade-offs, a more dynamic method in which the degree of generalization can be controlled has been developed.

The central idea of this method is to make use of statistical information concerning the distribution of the feature values of the instances in the 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 mentioned trade-offs 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.

3.1 Computing the Limits

It turns out that only very simple statistical methods are needed to compute such limits. Assuming that X is 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[4]) are normally distributed. This assumption seems not too strong for most applications. However, as we typically cannot assume that the actual values of m and σ are known, they have to be estimated. A simple way of doing this is just to compute the mean and the standard deviation of the training instances $(x_1, ..., x_n)$ belonging to the current 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 (and will be shown in Section 5) 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 the lower limit and smaller than the upper limit for $\alpha = 0.01$ is 99%.

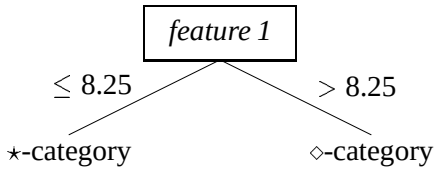
It can be argued that this is a rather crude way of computing the limits. A more elaborate approach would be to compute confidence intervals for the limits (i.e., for m and σ) and use these instead. This was actually the initial idea but it turned out that this only complicates the algorithm and does not increase the classification performance significantly.

4 A Simple Example

In this section a very simple example is presented to illustrate the suggested methods and compare them with the original ID3 algorithm. All instances are described by two numerical features, and the training instances belong to either of two categories: the $\star$-category or the $\diamond$-category. The system is given four training instances of each category.

The feature values of the training instances of the $\star$-category are: (3.0, 1.2), (3.5, 0.9), (4.5, 1.1), (5.0, 0.8) and the $\diamond$-category training instances are: (11.5, 1.8), (12.5, 1.5), (12.5, 1.8), (13.5, 2.1). Figure 2 shows the instances' positions in the feature space.

If these training-instances are given to the ID3 algorithm, the output will be the following decision tree:[5]



This tree represents the decision rule: if *feature 1* $\leq$ 8.25 then the instance belongs to the $\star$-category, else it belongs to the $\diamond$-category. The classification boundary that follows from this rule is illustrated in Figure 2 by a vertical dashed line. If we now apply the decision tree to an instance of another category ($\bowtie$) with the feature values (9.0,0.5), it will be (mis)classified as an instance of the $\diamond$-category.

[4] Systems that learn from examples do often not succeed in creating purely conjunctive descriptions for each category. Instead, they create a descriptions that consists of several disjuncts, where each disjunct corresponds to a subcategory, or cluster of training instances, represented by a conjunctive description (which typically corresponds to only one cluster of instances).

[5] Or a similar one, depending on the cut-point selection strategy. In all examples of this paper the cut-point is chosen by first sorting all values of the training instances belonging to the current node. The cut-point is then defined as the average of two consecutive values of the sorted list if they belong to instances of different classes.

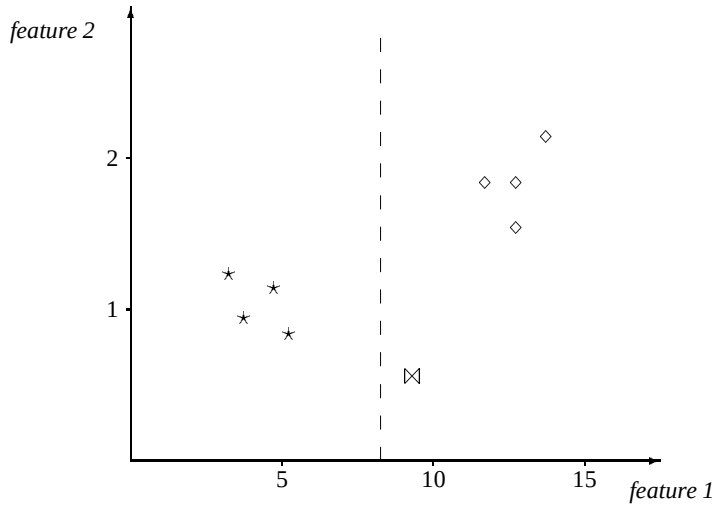


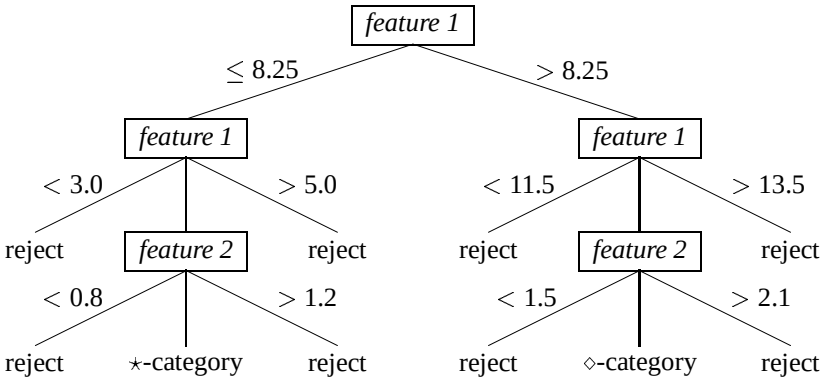
Figure 2: The feature space of the example. The boundary between the two categories ($\star$ and $\diamond$) induced by the ID3 algorithm is represented by the vertical dashed line.

4.1 Method I – Maximum Specific Description

Let us apply the same problem to the method based on the maximum specific description. The lower and the upper limits will then be as follows:

	feature 1	*feature 2*
$\star$-category	3.0 .. 5.0	0.8 .. 1.2
$\diamond$-category	11.5 .. 13.5	1.5 .. 2.1

These limits give rise to the following decision tree (which covers the dotted boxes in Figure 3):



If we now apply the augmented decision tree to the $\bowtie$-category instance, we first use the decision tree as before resulting in a preliminary classification which, still as before, suggests that it belongs to the $\diamond$-category. However, as we proceed further down the tree into the appended subtree, we will eventually encounter a test that brings us to a reject-leaf (i.e., we check whether the new instance is inside the dotted box, and find out that it is not). As a consequence, the instance is rejected and treated as an instance of a novel, or unknown, category.

4.2 Method II – Statistical Distribution

If we apply the statistical method with $\alpha = 0.05$, the lower and upper limits will be as follows:

	feature 1	*feature 2*
$\star$-category	2.2 .. 5.8	0.6 .. 1.4
$\diamond$-category	10.9 .. 14.1	1.3 .. 2.3

These limits will yield a decision tree similar to that of the maximum specific method but with different values on the rejection branches, and will cover the inner dashed boxes in Figure 3. Such a box can be interpreted as meaning that, if the assumptions mentioned above were correct and if the features are independent, 90.2% (0.95×0.95) of the instances of the category are inside the box. Just as with the maximum specific tree this tree will reject the $\bowtie$-category instance. We can also see that the lesser α-

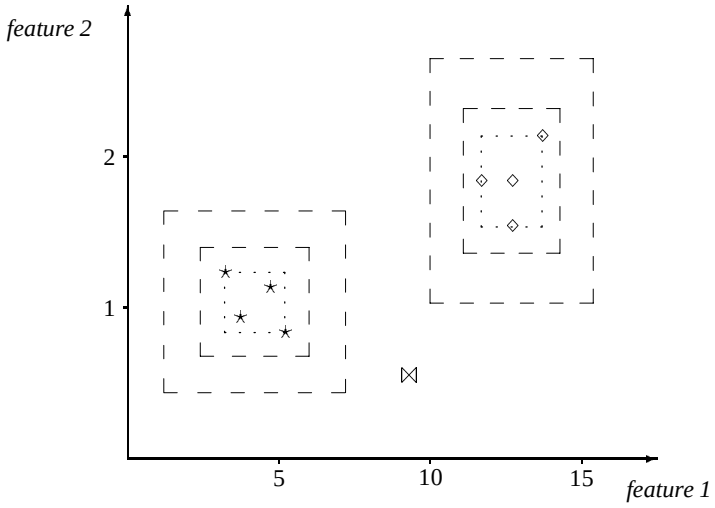


Figure 3: The feature space of the example. The maximum specific description correspond to the dotted boxes. The inner dashed boxes correspond to the description resulting from the method based on statistical distribution with α = 0.05. The outer dashed boxes correspond to α = 0.001.

value that is chosen, the more will the algorithm generalize. The outer dashed boxes correspond to α = 0.001, i.e., the probability is 99.8% that an instance of the category is inside the box.

5 Empirical Evaluation

In this section the two methods are evaluated empirically in three different domains: the classic Iris classification problem, a wine recognition problem, and a novel coin classification problem. 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 exist only 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.

5.1 The Iris Database

For this series of experiments the Iris database described by Fisher [6] has been used. It contains 3 categories of 50 instances each, where a category refers to a type of Iris plant (Iris Setosa, Iris Versicolor or Iris 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 (2×25) instances were used for training and 75 (3×25) instances for testing. Each experiment was performed with the original 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 Versicolor. Since these categories are linearly separable, the ID3 algorithm have no problem

	Iris Setosa			Iris Versicolor			Iris Virginica		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	100.0	0.0	0.0	100.0	0.0	0.0	0.0	100.0	0.0
ID3-Max	68.8	0.0	31.2	76.4	0.0	23.6	0.0	0.4	99.6
ID3-SD 0.2	42.4	0.0	57.6	47.2	0.0	52.8	0.0	0.0	100.0
ID3-SD 0.1	62.8	0.0	37.2	65.2	0.0	34.8	0.0	1.2	98.8
ID3-SD 0.05	73.2	0.0	26.8	82.0	0.0	18.0	0.0	2.8	97.2
ID3-SD 0.01	84.4	0.0	15.6	95.2	0.0	4.8	0.0	18.8	81.2
desired	100.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0	100.0

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

separating them, but misclassifies (of course) all the instances of Iris Virginica. 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. However, the maximum description algorithm works very well on this simple problem. It handles every category slightly better than the statistical algorithm for $\alpha = 0.1$. It is possible to achieve zero misclassifications if we choose $\alpha = 0.2$, but then we get a rejection rate of over 50% also for the two known categories.

A more difficult task is when the training set consists of instances of Iris Setosa and Iris Virginica. The results are shown in Table 2. The most interesting fact to notice here, is that the ID3-SD algorithm ($\alpha = 0.1$) performs significantly better than the ID3-Max algorithm. It has a slightly higher rejection-rate, but misclassifies over 60% less instances than the ID3-Max algorithm.

Finally, there is the case when the training instances are either Iris Versicolor or Iris Virginica (see Table 3). Here we can see that it is possible to reduce the number of misclassifications also of known categories by using characteristic descriptions (also Table 2 shows this, but to a lesser degree). For instance, the decision tree induced by the ID3 algorithm misclassifies 9.2% of the Iris Virginica instances, whereas both ID3-max and ID3-stat ($\alpha = 0.2$ and 0.1) induce trees that do not misclassify any of these instances. The main reason for this is probably 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.

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

	Iris Setosa			Iris Versicolor			Iris Virginica		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	0.0	100.0	0.0	91.6	8.4	0.0	90.8	9.2	0.0
ID3-max	0.0	0.0	100.0	66.0	2.0	32.0	65.2	0.0	34.8
ID3-stat 0.2	0.0	0.0	100.0	40.0	1.6	58.4	38.0	0.0	62.0
ID3-stat 0.1	0.0	0.0	100.0	60.0	3.2	36.8	64.4	0.0	35.6
ID3-stat 0.05	0.0	0.0	100.0	74.0	4.8	21.2	74.4	1.6	24.0
ID3-stat 0.01	0.0	0.0	100.0	84.4	4.8	10.8	82.8	4.8	12.4
desired	0.0	0.0	100.0	100.0	0.0	0.0	100.0	0.0	0.0

Table 3: Results from training set containing instances of Iris Versicolor and Iris Virginica (averages in percentages over 10 runs).

5.2 Wine Recognition

This data set comes from the Institute of Pharmaceutical and Food Analysis and Technologies in Genoa, Italy and is available at the UCI Repository of Machine Learning databases.[6] It contains results of chemical analyses of wines grown in the same region of Italy, but are fermented using three different kinds of yeast. In the analyses the quantities of 13 different constituents were measured. The data set consists of 59 instances of wine of type 1, 71 of type 2, and 48 of type 3. This is a more difficult problem than the above since we here are dealing with many, potentially irrelevant, features.

In each experiment 50 (2×25) instances were used for training and 60 (3×20) instances for testing. Each experiment was performed with the original ID3 algorithm, ID3-Max, and ID3-SD for the α-values: 0.1, 0.05, 0.01, 0.001 and 0.0001. Table 4, Table 5, and Table 6 shows the classification results of these experiments.

	Wine 1			Wine 2			Wine 3		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	93.0	7.0	0.0	90.0	10.0	0.0	0.0	100.0	0.0
ID3-Max	31.0	0.0	69.0	27.5	0.0	72.5	0.0	1.0	99.0
ID3-SD 0.1	19.0	0.0	81.0	22.5	0.0	77.5	0.0	0.0	100.0
ID3-SD 0.05	42.0	0.0	58.0	37.5	0.0	62.5	0.0	0.0	100.0
ID3-SD 0.01	67.5	3.0	29.5	63.5	0.0	36.5	0.0	1.5	98.5
ID3-SD 0.001	77.5	6.0	16.5	75.5	1.5	23.0	0.0	8.0	92.0
ID3-SD 0.0001	85.5	7.0	7.5	80.0	2.5	17.5	0.0	12.0	88.0
desired	100.0	0.0	0.0	100.0	0.0	0.0	0.0	0.0	100.0

Table 4: Result from training set containing wine of type 1 and 2 (averages in percentages over 10 runs).

We can see that for the right α-value (between 0.05 and 0.001 depending on the constraints of the application) the ID3-SD algorithm performs significantly better than the ID3-Max algorithm. ID3-Max has greater problems than ID3-SD when the number of features grows since each additional feature decreases the probability that every feature value of an instance is between the limits. In other words, the number of training instances needed by ID3-Max increases when the number of features increases.

5.3 Coin Classification

This corresponds to the problem of learning the decision mechanism in coin sorting machines described in the introduction.[7] A preliminary study of this problem is described in a Master's thesis by Mårtensson [12]. He tested a neural network approach, a statistical method based on a Bayesian classifier, and a decision tree induction method (ID3-Max) on the problem. Although these methods had approximately the same classification accuracy (on known types of coins), he concluded that ID3-Max was the most appropriate method for this application, mainly because it learns explicit rules and is fast both in the learning and in the classification phase.

[6] Retrievable by anonymous ftp from `ftp.ics.uci.edu` in directory `/pub/machine-learning-databases`.

[7] This work has in part been carried out in collaboration with Scan Coin AB (Malmö, Sweden).

	Wine 1			Wine 2			Wine 3		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	99.5	0.5	0.0	0.0	100.0	0.0	99.0	1.0	0.0
ID3-Max	33.0	0.0	67.0	0.0	0.0	100.0	35.0	0.0	65.0
ID3-SD 0.1	20.0	0.0	80.0	0.0	0.0	100.0	26.5	0.0	73.5
ID3-SD 0.05	46.0	0.0	54.0	0.0	0.5	99.5	44.0	0.0	56.0
ID3-SD 0.01	74.0	0.0	26.0	0.0	5.0	95.0	78.5	0.0	21.5
ID3-SD 0.001	86.5	0.0	13.5	0.0	40.5	59.5	91.0	0.0	9.0
ID3-SD 0.0001	94.5	0.0	6.5	0.0	59.0	31.0	95.0	0.0	5.0
desired	100.0	0.0	0.0	0.0	0.0	100.0	100.0	0.0	0.0

Table 5: Result from training set containing wine of type 1 and 3 (averages in percentages over 10 runs).

	Wine 1			Wine 2			Wine 3		
	correct	miss	reject	correct	miss	reject	correct	miss	reject
ID3	0.0	100.0	0.0	91.5	8.5	0.0	84.5	15.5	0.0
ID3-Max	0.0	2.5	97.5	31.0	0.0	69.0	32.0	0.0	68.0
ID3-SD 0.1	0.0	0.0	100.0	25.5	0.0	74.5	24.0	0.0	66.0
ID3-SD 0.05	0.0	0.0	100.0	43.0	0.0	57.0	40.5	0.0	59.5
ID3-SD 0.01	0.0	10.0	90.0	65.5	0.5	34.0	64.5	2.0	33.5
ID3-SD 0.001	0.0	22.0	78.0	77.0	3.0	20.0	73.0	6.0	21.0
ID3-SD 0.0001	0.0	29.5	70.5	83.0	4.0	13.0	77.0	10.5	12.5
desired	0.0	0.0	100.0	100.0	0.0	0.0	100.0	0.0	0.0

Table 6: Result from training set containing wine of type 2 and 3 (averages in percentages over 10 runs).

In our 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 company'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 ($2\times7\times50$) instances for testing. This scenario is quite similar to the actual situation where you in the training phase expose the system only to the coins of one country, but in the classification phase also confront it with coins of other countries. Each experiment was performed with the original ID3 algorithm, the maximum specific tree algorithm (ID3-Max), and the algorithm based on statistical distribution (ID3-SD) for the α-values: 0.1, 0.05, 0.01, 0.001, and 0.0001.

Table 7 shows the classification results when training on the Canadian coin database. We can see that

	Canadian Coins			Foreign Coins		
	correct	miss	reject	correct	miss	reject
ID3	99.7	0.3	0.0	0.0	100.0	0.0
ID3-Max	83.7	0.0	16.3	0.0	0.0	100.0
ID3-SD 0.1	62.1	0.0	37.9	0.0	0.0	100.0
ID3-SD 0.05	77.5	0.0	22.5	0.0	0.0	100.0
ID3-SD 0.01	92.2	0.0	7.8	0.0	0.0	100.0
ID3-SD 0.001	97.9	0.0	2.1	0.0	0.0	100.0
ID3-SD 0.0001	98.9	0.0	1.1	0.0	0.0	100.0
desired	100.0	0.0	0.0	0.0	0.0	100.0

Table 7: Result from training set containing Canadian coins (averages in percentages over 10 runs).

all foreign coins (i.e., Hong Kong coins) are rejected, except of course for the ID3 algorithm. Neither were there any problems with misclassifications. However, 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 coins and very few misclassifications (less 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.

In Table 8 the results when training on the Hong Kong coin database are shown. As indicated by the percentages of misclassifications of known types of coins, this is a more difficult problem. Although there are two α-values (0.001 and 0.0001) that meet the requirements, they are very close to the acceptable number of rejects (0.001) and misclassifications (0.0001) respectively.

6 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

	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 0.1	60.0	0.0	40.0	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 8: Result from training set containing Hong Kong coins (averages in percentages over 10 runs).

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.

6.1 Some Potential Limitations

The main limitation of the SD-method seems to be that it is only applicable to numerical attributes. The maximum specific 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 specific 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. However, the degree of generalization can, of course, only be controlled for numeric features.

As we observed in section 5, some instances that would have been correctly classified by the decision tree are rejected by the augmented tree (i.e., if any of its feature values is outside their interval). This is related to the trade-off between the number of rejections and misclassifications that can be controlled by selecting a proper α-value. Development of methods to automatically determine the appropriate degree of generalization belongs, however, to future research.

Moreover, the original ID3-algorithm is quite good at handling the problem of irrelevant features

(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 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 most values of this feature will be inside this interval, the feature will still be irrelevant for the classification.

Another potential problem for the ID3-SD algorithm 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 (which are fundamental for computing the limits) will not be sufficiently good. However, preliminary experiments in the coin classification domain indicates that the classification performance decreases slowly when the training examples get fewer. As can be seen in figure 4, 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. [7]).

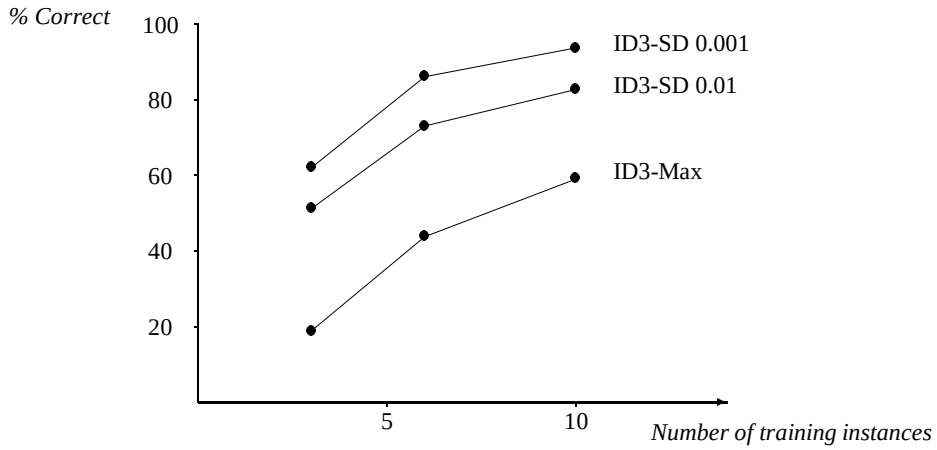


Figure 4: The percentage of correctly classified instances of known categories (Canadian coins) as a function of the number of instances of each category in small training sets (averages over 10 runs). The remaining instances were rejected.

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%[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.

6.2 Noisy Data and The Generality of the SD-approach

The ID3-SD algorithm is better at handling noisy data than the ID3-Max algorithm 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. (See Figure 5) A method, not yet evaluated, for

[8] We are here still assuming that the assumptions made in Section 3.1 are correct

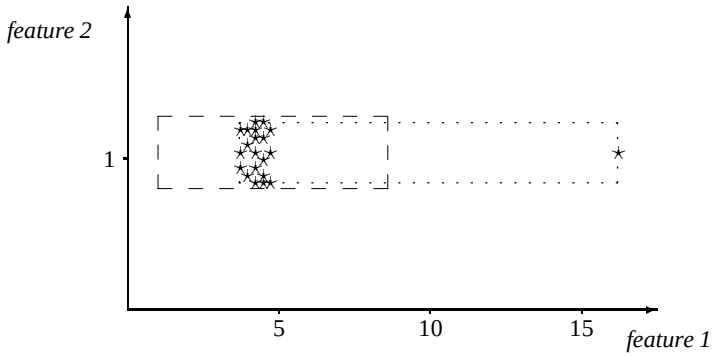


Figure 5: A category with twenty good and one noisy instance. The maximum specific description correspond to the dotted box. The dashed box correspond to the description resulting from the ID3-SD algorithm with α = 0.1.

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 the traditional ID3 algorithm 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 a pruning method (cf. Mingers [9]) to cut off the undesired branches, or use any other noise tolerant algorithm (e.g., C4.5 [11]) for inducing decision trees, 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 (supervised or unsupervised) provided that all attributes can be ordered (using the hybrid approach).[9] However, if the instances of a category corresponds to more than one cluster in the feature space (cf. disjunctive concepts), the method will probably 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. This is also the answer to the question: Why bother building a decision tree in the first place? Could we not just compute the lower and the upper limits for every category and test unknown instances against these? This is, in fact, an adequate method when dealing with purely conjunctive concepts (if we don't care about classification efficiency). However, when dealing with disjunctive concepts, for the reason mentioned above, this would not be a good approach. In this case, we must have an algorithm that is able to find suitable disjuncts of the concept (which, in fact, is an unsupervised learning problem), a task that ID3-like algorithms normally are quite good at.[10]

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 clas-

[9] In incremental methods, however, the training instances must be saved.

[10] However, Van de Merckt [5] has suggested that for numerical attributes a similarity-based selection measure is more appropriate for finding the correct disjuncts than the original entropy-based measure that has been used in the empirical evaluations presented here.

sified 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 ID-3 algorithm (i.e., it can be regarded a case study). Moreover, this is the main reason why we have not compared the ID3-SD algorithm with other kinds of algorithms that learn characteristic descriptions.

Acknowledgements

I wish to thank Arne Andersson, Eric Astor, Christian Balkenius and Måns Holgersson for helpful comments and suggestions.

References

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

[2] G.A. Carpenter, S. Grossberg, and J.H. Reynolds. ARTMAP: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, 4, 1991.

[3] P. Davidsson. Concepts and autonomous agents. LU–CS–TR: 94–124, Dept. of Computer Science, Lund University, Sweden, 1994.

[4] P. Davidsson. A note on learning characteristic decision trees. In *Annual Workshop of the Swedish AI Society (SAIS'94)*, pages 67–71, 1994.

[5] T. Van de Merckt. Decision trees in numerical attribute spaces. In *IJCAI-93*, pages 1016–1021, 1993.

[6] R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7, part II:179–188, 1936.

[7] 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.

[8] A. Hutchinson. *Algorithmic Learning*. Clarendon Press, 1994.

[9] J. Mingers. An empirical comparision of pruning methods for decision tree induction. *Machine Learning*, 4(2):227–243, 1989.

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

[11] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[12] E. Mårtensson. Improved coin classification, (Master's thesis). LU–CS–EX: 94–2, Dept. of Computer Science, Lund University, Sweden, 1994. (In Swedish).

[13] 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.