Using Subclasses to Improve Classification Learning

Achim Hoffmann, Rex Kwok, and Paul Compton

School of Computer Science and Engineering,
The University of New South Wales,
UNSW SYDNEY NSW 2052,
Australia
{achim,rkwok,compton}@cse.unsw.edu.au

Abstract. We propose to use systematic simulation studies as opposed to the use of real-world benchmark datasets to better understand the behaviour, strengths and weaknesses of machine learning algorithms. Simulated data sets allow much better control and understanding of the nature of the learning problem than empirical benchmark data sets.
To demonstrate the value of our proposed research methodology, we describe in this paper the results of our studies concerning the problem of learning multiple classes. We derived the following hypothesis: *"Learning classification functions using decision tree learners can be helped by providing additional subclass labels."* To illustrate, for learning a two class problem "car is OK/car needs service" it can be helpful to provide a finer-grained classification in the training data such as "car OK", "faulty brakes", "faulty engine", "faulty lights", etc.
This hypothesis was corroborated using a number of 'real-world' multi-class data sets from the UCI ML repository. Our empirical studies demonstrate the usefulness of the proposed research methodology using artificial data sets as an important methodological complement to using real-world datasets.

1 Introduction

The careful evaluation of the effectiveness of a particular approach in AI is important but often difficult to conduct in practice. Empirical studies have been conducted in a large number of subfields of AI, including theorem proving, constraint satisfaction, vision, machine learning and neural networks. Usually benchmark datasets are used for the comparison of approaches and evaluation studies. Benchmark datasets, such as the UCI machine learning (ML) repository, are often obtained from real data.

This paper demonstrates the use of simulated domains in investigating the strengths and weaknesses of machine learning techniques. A model of a domain is developed that provides a (probabilistic) source of examples and a target classification function which is the target of the learning process. The major insight that has arisen from the work reported in this paper is that the simulated domain approach enables one to investigate how the domain structure affects the performance of machine learning techniques much more readily than using 'real-world' data where the target function is unknown.

In fact, using simulated domains, we found an intriguing behaviour of decision tree learners (C4.5) when dealing with a large number of different classes: With an increasing

L. De Raedt and P. Flach (Eds.): ECML 2001, LNAI 2167, pp. 203–213, 2001.

number of classes that need to be distinguished, the accuracy of C4.5 *increased* for target concepts of comparable complexity.

This observation lead us to hypothesise a new approach to achieving increased accuracy on classification problems with two or only few classes as follows:

If the provided examples can be divided into 'meaningful' subclasses then using the subclass label to train a learner may result in improved accuracy for classifying according to the initial set of two or only a few classes. The critical question here is, what exactly constitutes meaningful subclasses. Ultimately, the success of using subclass labels depends on the distribution of the data of each subclass in the instance space. If it is distributed as in our simulation studies, it will be successful. If it is distributed differently, we don't know how useful it is. In practice, given our current understanding, we have to deal with the question of meaningful subclasses on a rather intuitive basis. Intuitively, meaningful subclasses of a class should be ontologically justifiable. I.e. each subclass should occupy a separate area in the instance space, such that the learner can represent a suitable separating function of low complexity (i.e. few additional nodes in a decision tree). On the other hand, the domain must be sufficiently complex to represent a challenge for the learner when no subclass labels are provided.

The paper is organised as follows: In section 2 we discuss our methodology and rationale for using artificial data sets as a general useful strategy for exploring issues in machine learning that complements the widespread use of 'real' data sets (often from UCI) for experimental studies. The following section presents the details of our simulated domain approach. In section 4 we present our empirical corroboration of our hypothesis using 'real-world data' from the UCI repository. Section 5 contains a discussion of our approach and our new hypothesis.

2 Simulation Studies

In this paper we advocate simulation studies as a complement to both empirical studies using 'real-world' data sets as well as to rigorous mathematical analyses of learning tasks and techniques.

Empirical studies using 'real-world' data sets, taken from the UCI ML repository or 'fresh' data sets from an available application domain, may reveal certain aspects of the domain the data comes from. This is an important objective in itself, although characteristics of domains may be more effectively studied by not compounding the analysis of the data with the study of learning algorithms whose characteristics are insufficiently understood.

Similarly, the study of the learning techniques using 'real-world' data sets is compounded by the poorly understood nature of the domains the data sets come from. Some simple characteristics of 'real-world' data sets have been drawn up, e.g. in the STATLOG project [6] where domains were characterised by the number and types of attributes (whether numerical or non-numerical etc.). It is clear that such a coarse characterisation is only the beginning of understanding peculiarities of domains.

In theoretical/mathematical studies of learning algorithms and learning tasks, it is usually very difficult both, to characterise a given learning algorithm in detail as well

as to characterise a learning task. As a consequence, the results achieved so far in computational learning theory are rather coarse results, where many practitioners tend to question whether these studies have any relevance whatsoever for the practice of machine learning.

Artificial data sets or the simulation of a domain have been used on many occasions in Machine Learning research mainly to provide evidence for the validity of a particular statement; in many cases for showing the superiority of one learning technique over another one. See e.g. [1,7,4]. In [2] a data generator was used which is available from the UCI ML repository.

On other occasions, artificially generated benchmark data sets have been used to gauge the abilities of learning techniques. E.g. the two spiral problem in neural networks [3] is an example of that, where the purpose of such benchmarks is rather questionable, as it is not clear at all that solving such artificial problems is a worthwhile endeavour.

In this paper, we advocate the use of simulated domains and to run implemented learning algorithms on them. By properly setting up the simulation environment for domains and running systematic experiments, results can be achieved in a much easier way than in the traditional mathematical analysis of computational learning theory. The simulation results can have the same rigour as mathematical proofs. The results are in principle reproducible without the recourse to empirical data. I.e. the used data is generated using a data source that can be described in a mathematically rigorous way. Similarly, the learning algorithm can also be described in a mathematically precise way, as the actual program code would warrant. However, to study how a given learning algorithm would perform on a precisely characterised domain (the domain would be described as a probabilistic source of data) is virtually impossible by traditional mathematical approaches as the analysis of the execution of perhaps many million computing steps of a program on perhaps ten thousand or more data points is practically infeasible.

On the other hand, simulation studies can very easily study the behaviour of algorithms on large data sets at least in individual instances (running an algorithm on a particular data set). Mathematical theorems have usually a generality which covers a whole range of learning tasks or training data sets. Similar generality on the basis of simulation experiments can only be achieved by drawing conclusions from experimental results very carefully. However, in many cases it is not really necessary to establish 'absolute' validity of a conclusion drawn from simulation experiments. This is the case, for example, where one moves in uncharted territory and one first wants to obtain a coarse impression of the phenomena around which, if they appear sufficiently important, in turn can be studied in more detail. Such initial studies would just serve the purpose of hypotheses generation. Once interesting hypotheses have been found, further studies can be designed and conducted to establish the validity of a hypothesis.

Gaining such a coarse survey and the generation of interesting hypotheses is comparably simple using simulated domains and requires only very limited resources. Yet it is much more revealing and effective than empirical studies involving 'real-world' data sets.

The missing link to the practice of machine learning, is the relationship between the characteristics of the simulated domains and the characteristics of the real domains. While this remains a critically important relationship to explore, it appears still that

progress can and should be achieved via simulation studies: The simulation studies have the potential to allow us further insight into why a given learning algorithm performs well or poorly upon a particular characteristic of the simulated domain. On the basis of such findings, one can build a conceptual framework of important characteristics or features one need to look for in assessing 'real-world' domains.

Studying 'real-world' domains has at least two objectives that can be distinguished: To develop better learning techniques and to ascertain the suitability of already existing learning techniques.

The choice of a learning technique for a given domain can then be guided either by characteristics of the individual domain, perhaps gauged by preliminary data analysis, or by characteristics that are commonly found among *similar* domains. Of course, in turn it is not clear what it might mean that two domains are similar.

In any case, in order to find useful characteristics of real-world domains, and in order to become clearer about what constitutes similar domains, simulation studies have the potential to lay the conceptual foundation by clarifying the aspects in a domain upon which the success of one or more learning techniques depend. Those aspects which do not significantly influence the performance of a learning technique can safely be ignored when it comes to characterising and assessing real-world domains.

3 Learning Multiple Classes by Decision Trees

Learning multiple classes is an important learning task as recent research [8,5] on extending learning techniques to deal with multiple classes shows. C4.5 or decision tree learners generally are naturally suited to the task. In our simulation studies, we wanted to explore the ability of C4.5 to handle large numbers of classes in more detail.

3.1 Simulations

The target concepts we generate are binary decision trees. At each node an attribute is randomly chosen and a random threshold (a floating point value between 0 and 100) is generated. Leaf nodes are randomly assigned a class. Parameters are used to control the depth of the tree, the number of classes and the number of available attributes. Given a decision tree, examples are generated simply by picking uniformly distributed random points in the instance space and seeing how the tree classifies the point.

The study presented here looks at how the number of classes affects the error rate produced by C4.5. Typical results can be seen in Figure 1. The only variable that is varied is the class number. As the number of classes increases, the error rate initially decreases and then plateaus out. In this case, the error rate with only 2 classes is more than double the error rate when there are 40 classes. This result is surprising for three reasons. Firstly, the complexity of the target concepts are similar, since the number of cuts and divisions to the instance space is determined by the depth of the target concept. Increasing the class number only increases the number of labels with which the divisions can be named. Secondly, guessing becomes harder. With fewer classes, a guess is more likely to produce the correct answer than when there are more classes. Finally, when there are fewer classes, the target concept often receives some pruning. For instance,

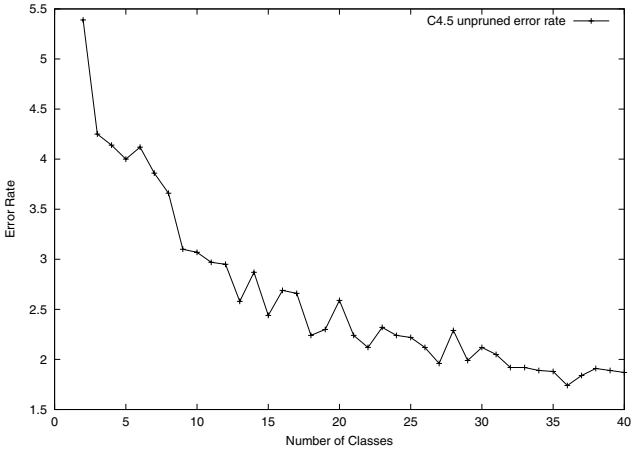

Fig. 1. Varying the number of classes in the simulated domain. Each point is the average over 20 trials.

consider a target concept with two classes. When two adjacent leaf nodes are attributed the same class (there is a 50% chance of this happening), the last attribute test is rendered redundant.

The graph in Figure 1 is quite general. Changes to the depth of the target concept or the number of available attributes has only a slight effect. The main variable affecting the trend is the average number of examples presented to C4.5 for each leaf node in the target concept. As this number increases, the curve shifts downwards.

The reason for the trend may be due to the anecdotally known weakness of C4.5 to handle target concepts where a class is scattered across the instance space. With only two classes, half of the leaf nodes in the target concept will be assigned to one class. With more classes, the probability that the class is represented by a few convex regions in the instance space becomes greater. One hypothesis to account for the degradation in performance is that C4.5 is oversimplifying the target concept by joining close regions with the same class. The 'small' intervening region where the class alternates would then be misclassified.

If this is true, then it might be expected that smaller trees are produced when there are fewer classes. In fact, the opposite is the case. With fewer classes, larger trees are produced. The plot of tree size corresponding to the target concepts in Figure 1 can be seen in Figure 2. Further research will be required to determine the exact cause for these results.

4 Results with UCI Datasets

The simulation results in the previous section clearly show that C4.5 performs better when there are more class labels. This is in spite of the fact that the complexity of the target concept remains the same. This hypothesis can be corroborated by using datasets from the UCI repository. Since we cannot increase the number of classes for a single

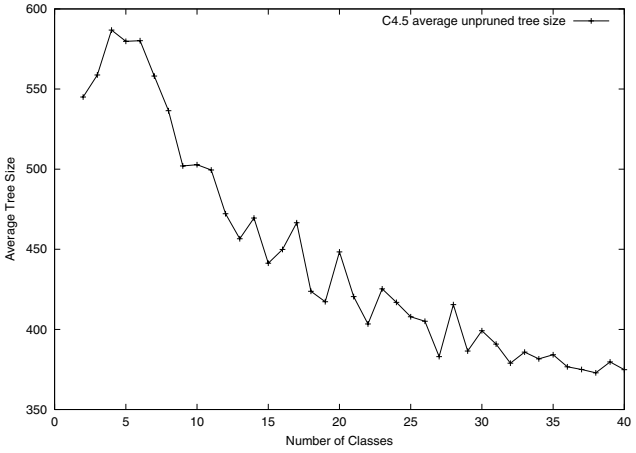

Fig. 2. Varying class number and the resulting C4.5 tree sizes. The tree size decreases with an increasing number of classes.

dataset (except arbitrarily), we study the contrapositive to our hypothesis. By grouping or conflating classes in a dataset, we would expect the performance of C4.5 to degrade.

In conflating several classes for a dataset, the learning task becomes simpler. To compensate for this, *confusion errors* between equivalent classes need to be eliminated. For example, suppose that a dataset X contains two classes, A and B. Also, suppose that X is altered to Y by setting A and B to the same class. To compare the performance of C4.5 on X and Y, it is necessary to eliminate the errors where A type examples are misclassified as B (and *vice versa*) from the result of C4.5 on X.

A summary of the UCI results can be seen in Figure 3. For each dataset, trials were carried out by grouping classes in a certain manner. Each grouping was then given equal weight in determining the average. To generate some variation in the amount of training data, both 2-fold and 10-fold cross-validation trials were carried out. With the 10-fold cross-validation trials, only two datasets seem to confirm our hypothesis to any degree. However, the 2-fold cross-validation trials gave a far better confirmation of our hypothesis. This agrees with our simulation results which show that the trend towards improved performance with increasing class numbers is more accentuated with fewer training examples.

The exact groupings and the performance of C4.5 for each particular grouping can be seen in the following figures. For each data set we show the average error rates of the pruned trees of 50 2-fold cross validation results and the average error rates from 50 10-fold cross validations. The first line for each dataset shows the performance of C4.5 without altering the data. The last column lists the classes present in the dataset.

The average improvements using subclasses as listed in figure 3 cannot be taken too seriously, as averaging results from a collection of sometimes more or less arbitrarily chosen class groupings is not a mathematically meaningful operation. However, it should provide a first glance.

The car evaluation dataset (see Figure 4) best corroborates our hypothesis. An example in the dataset describes the price and technical features of a car and is assigned

Dataset	Average Improvement in% (2-fold trials)	Average Improvement in % (10-fold trials)
car evaluation	37.9	19.95
dermatology	7.4	1.3
glass identification	9.6	0.3
nursery	1.9	0.7
handwritten digits	4.1	0.8
image segmentation	11.4	11.9

Fig. 3. Summary of UCI trials.

Error in % (2fold cv)		Error in % (10-fold cv)		Groupings
normal	subclasses	normal	subclasses	
17.46	17.46	8.06	8.06	{unacc(u)}, {acc(a)}, {good(g)}, {vgood(v)}
8.06	8.565	5.03	4.60	{a, g, v}, {u}
18.90	17.03	8.53	6.50	{u, v}, {a}, {g}
26.34	11.18	4.91	3.65	{u, a}, {v}, {g}

Fig. 4. The **car evaluation** data set (6 atts, 4 classes): The groupings were done on the basis of forming larger groups of similar cars according to their rated quality (in the class label).

one of four classes. The distribution of the examples is heavily weighted towards two classes. There is also an intuitive ordering to the classes, ranging from unacceptable to very good. In all but one trial, grouping classes results in a significantly higher error rate than the adjusted error rate.

Error in % (2fold cv)		Error in % (10-fold cv)		Groupings
normal	subclasses	normal	subclasses	
7.09	7.09	6.21	6.21	{psoriasis(p)}, {seboreic dermatitis(sd)}, {lichen planus(lp)}, {pityriasis rosea(pr)}, {cronic dermatitis(cd)}, {pityriasis rubra pilaris(prp)}
7.08	7.08	7.07	6.21	{pr, prp}, {p}, {sd}, {lp}, {cd}
6.37	7.08	5.80	6.21	{pr, prp}, {sd, cd}, {p}, {lp}
8.74	6.57	5.29	5.30	{p, lp}, {sd}, {cd}, {pr}, {prp}
7.25	6.81	5.57	5.67	{sd, lp}, {p}, {cd}, {pr}, {prp}

Fig. 5. The **dermatology** data set (34 atts, 6 classes): As we have no knowledge of the various conditions being classified here, the groupings have to be considered random groupings, which are presumably not meaningful.

The dermatology, glass identification, and nursery datasets (Figure 5,6,7) all show only a slight tendency for C4.5 to perform worse when classes are grouped together.

However, there are two significant results in the 2-fold cross-validation trials. In one dermatology trial and one glass identification trial the adjusted error rate was exceeded by some 30%. It can be argued that the class groupings used for these particular trials are counter-intuitive. Consultation with domain experts would be required to determine whether this is the case.

Some of the dermatology results may point to an interesting finding. The groupings in the first two dermatology trials resulted in virtually no change in the adjusted error. This means that C4.5 is not confused between the classes in those groupings. However, when the dataset is changed to reflect these groupings, two contrasting results are obtained. In one trial, an absolute increase in error (in the order of 10%) is observed. In the other, a decrease is error (in the order of 6%) is observed.

Error in % (2fold cv)		**Error in % (10-fold cv)**		**Groupings**
normal	subclasses	normal	subclasses	
36.98	36.98	32.73	32.73	{building_windows_float_processed(bwf)}, {building_windows_non_float_proc'd(bwnf)}, {vehicle_windows(vw)}, {containers(c)}, {tableware(t)},{headlamps(h)}
21.72	20.78	19.21	19.62	{bwf, bwnf}, {vw}, {c}, {t}, {h}
38.86	36.74	32.64	32.55	{t, vw}, {bwf}, {bwnf}, {c}, {h}
34.81	34.44	31.70	31.60	{vw, c, t, h}, {bwf}, {bwnf}
42.92	33.84	30.78	29.96	{bwf, h}, {bwnf, t}, {c}, {vw}

Fig. 6. The **glass identification** data set (10 atts, 6 classes): As we have no knowledge of the various kinds of glass being classified here, the groupings have to be considered random groupings, which are presumably not meaningful.

Error in % (2fold cv)		**Error in % (10-fold cv)**		**Groupings**
normal	subclasses	normal	subclasses	
6.80	6.80	3.51	3.51	{not_recom(nr)}, {recommend(r)}, {very_recom(vr)}, {priority(p)}, {spec_prior(sp)}
6.94	6.80	3.54	3.51	{nr, sp}, {r}, {vr}, {p}
6.47	6.78	3.48	3.50	{nr, r, vr}, {p}, {sp}
5.34	4.94	2.35	2.31	{nr, r}, {vr, p}, {sp}

Fig. 7. The **nursery** data set (8 atts, 5 classes): The groupings were done on the basis of forming larger groups according to their ratings (in the class label).

Handwritten digits (Figure 8) only present a moderate trend. This is perhaps a little surprising because the dataset shares some important characteristics of the target concepts used in the simulation studies. It contains a reasonable number (10) of classes and errors result from a number of digits being confused with other digits. However, there

Error in % (2fold cv)		Error in % (10-fold cv)		Groupings
normal	subclasses	normal	subclasses	
11.51	11.51	9.52	9.52	{0}, {1}, {2}, {3}, {4}, {5}, {6}, {7}, {8}, {9}
8.48	7.25	7.11	5.96	{0, 1, 2, 3, 4}, {5, 6, 7, 8, 9}
5.54	5.51	4.53	4.62	{0, 2, 4, 6, 8}, {1, 3, 5, 7, 9}
7.57	7.12	5.80	5.97	{0, 1, 2, 3, 4, 7}, {5, 6, 8, 9}
5.72	6.10	4.58	5.13	{0, 1, 4, 6, 8, 9}, {2, 3, 5, 7}
9.36	9.11	7.62	7.51	{3, 7, 9}, {2, 8}, {5, 6}, {0}, {1}, {4}, {8}

Fig. 8. The **handwritten digits** data set (64 atts, 10 classes): We listed various grouping of the 10 digits. The numbers 5, 6, 8, 9, appear to have a rather similar shape for the human eye. This group is reflected in the first and second grouping where we have the groups {0, 1, 2, 3, 4} vs. {5, 6 ,7 8, 9} on the one hand and the same grouping with the 7 shifted to the lower digits on the other hand. In the first case grouping resulted in a significant improvement over the normal training. When the 7 was shifted to the lower digits, the absolute error rate dropped further. However, C4.5 managed to perform significantly better than with the first grouping, resulting in no improvement over the normal training procedure. The result suggests that the instance space may be less fractured for the classes {0, 1, 2, 3, 4, 7} vs. {5, 6, 8, 9}, resulting in a better performance of C4.5 and a lower relative advantage using subclass labels.

are two stand out results. In one, the digits are divided into two groups, those between 0 and 4 and those between 5 and 9. For this trial, the error rate is approximated 18% greater than the adjusted rate. In another trial where the digits are separated according to whether they are prime or not, the error rate is 6–11% lower than the adjusted value.

Error in % (2fold cv)		Error in % (10-fold cv)		Groupings
normal	subclasses	normal	subclasses	
6.21	6.21	3.38	3.38	{grass(g)}, {foliage(f)}, {window(w)}, {sky(s)}, {cement(c)}, {path(p)}
6.60	5.56	3.97	3.30	{g, f}, {w, s}, {c, p}
3.75	3.60	2.09	2.02	{g, f, s}, {w, c, p}

Fig. 9. The **image segmentation** data set (19 atts, 7 classes): The first grouping was done on the basis of similarity to the human eye. The second grouping is about natural and artificial surfaces, which appears to be a more arbitrary grouping. The results show a marked improvement for the intuitive grouping while the improvement for the second grouping is rather marginal.

In Figure 9 the results from image data are shown. For image domains intuitive similarities to the human eye are rather easy to determine. However, what appears similar to the human eye is not necessarily a good grouping for the learner, depending on the extracted features of the image used as the image representation to the learner.

Besides these observations of the behaviour of the error rates, we made another noteworthy observation concerning the tree sizes: The sizes of the learned trees using subclasses was about the same size as the trees trained without subclasses. No clear trend had been observed, although occasionally marked differences occurred. The results with

the UCI data sets did not show the same clear trend we observed on our simulated domains. In the simulated domains we had a clear trend towards smaller trees when more classes had to be covered by a single tree. We do not yet understand why the same trends were not reproduced for the learning trials using the UCI data.

This suggests that our simulated domains are not accurately reflecting the domain structure of the UCI data. At the same time we are vindicated with our approach of using simulation studies to enhance the understanding of learning processes, as the results with the UCI data were rather mixed.

The mixed results are due to significant differences in the various domains of the UCI data, which are poorly understood at this stage. To enhance our understanding, further studies on simulated domains in comparison to 'real-world' data sets are necessary in order to understand which domain features are responsible for what effects. This is not limited to studying the effect of subclasses but includes also many other important aspects of learning, such as overfitting, noise handling, model selection, or the effectiveness of committee classifiers.

5 Discussion and Conclusion

While using artificial data sets is not new in machine learning research, it has not been widely used as a tool to obtain a better understanding of the domain characteristics that determine the relative success or lack of success of a given technique. We believe that a host of important insights into characteristics of ML techniques and domains can be obtained using the presented methodology more extensively. The purpose of such simulations is insight into the nature of learning algorithms and reasons why they behave as they do. Using artificial domains it is much easier to understand the characteristics of learning algorithms as the domains are well-controlled and can easily be modified to validate emerging hypotheses.

We demonstrated our point by presenting results showing an intriguing behaviour of C4.5 to improve its classification error when more classes have to be distinguished. We explained the behaviour, that is at first sight counterintuitive, with the fact that more classes may make it easier for C4.5 to find an appropriate split point for an attribute. This is due to the fact that with few classes a single split on an attribute is likely to split also those training examples apart which belong to the same class (but lie in different pockets in the instance space). In contrast to that when C4.5 deals with many classes, it is more likely that C4.5 will find a split on an attribute which does not split apart the examples belonging to the same class. In this sense, the subclasses can be viewed as additional information provided to the learner for guiding the learning process.

In other words, it is easier for C4.5 to learn target functions that assign to each class one or a few convex areas in the object space, as opposed to rather complex shaped areas for only a few classes.

Guessing the correct class becomes drastically more difficult with an increased number of classes. In this light, it is still surprising that the additional information provided to C4.5 using subclasses more than outweighs the increased difficulty of 'guessing'.

Our observation led us to the hypothesis that it may be advantageous for learning tasks for only few classes, to provide additional subclass labels to C4.5, as C4.5 may

use the additional information to find the proper splits. Obviously, this is only going to work, where the subclass labels are assigned in a meaningful way (ideally selecting a single convex sub-area of the object space). Under what circumstances subclass labels would be meaningful in the above sense is not easy to decide at this stage.

However, our empirical studies with the UCI data suggest that using subclass labels can indeed be used as a general heuristic which may well lead to better results for many practical data sets. Further studies are needed to better understand the observed effects. Our results so far show that the effect of using subclasses can significantly differ from application to application.

Another intriguing avenue for further research seems to be the exploration of how other learners react to additional subclass labels. In particular, for some learners, such as Naive Bayes, which let the 'different classes compete' against each other to make a decision on a classification, additional subclasses may have significantly more pronounced effects than for decision tree learners. For decision tree learners the recognition phase is not at all affected by the finer grained classifications, as each subclass is uniquely mapped to one of the grouped classes.

The fact that the empirical UCI data sets produced mixed results confirms our methodological approach to a deeper understanding of learning by using artificial data sets. Using the real-world data sets leads often to mixed results and it is very difficult to understand the reason of the varying results without further studies that involve different types of domain models in a controlled way.

References

1. T. Dietterich. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 7(10):1895–1924, 1998.
2. J. H. Gennari, P. Langley, and D. Fisher. Model of incremental concept formation. *Artificial Intelligence*, 40:11–61, 1989.
3. K. Lang and M. Witbrock. Learning to tell 2-spirals apart. In *Connectionist Models Summer School*, 1988.
4. D. D. Margineantu and T. G. Dietterich. Bootstrap methods for the cost-sensitive evaluation of classifiers. In *Proceedings of the* 17^{th} *International Conference on Machine Learning*, pages 582–590. Kaufmann, 2000.
5. C. Mesterharm. A multi-class linear learning algorithm related to Winnow. In *Neural Information Processing Systems (NIPS-12)*, pages 519–525. MIT Press, 2000.
6. D. Michie and D. Spiegelhalter. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.
7. T. Scheffer. Predicting the generalization performance of cross validatory model selection criteria. In *Proceedings of the* 17^{th} *International Conference on Machine Learning*. Kaufmann, 2000.
8. J. Weston and C. Watkins. Multi-class support vector machines. Technical Report CSD–TR–98–04, Royal Holloway, University of London, 1998.