Learning from Ambiguously Labeled Examples*

Eyke Hüllermeier and Jürgen Beringer

Fakultät für Informatik

Otto-von-Guericke-Universität

Universitätsplatz 2

D-39106 Magdeburg, Germany

fon: ++49-391-67-18842, fax: ++49-391-67-12020

{huellerm,beringer}@iti.cs.uni-magdeburg.de

Abstract

Inducing a classification function from a set of examples in the form of labeled instances is a standard problem in supervised machine learning. In this paper, we are concerned with *ambiguous label classification* (ALC), an extension of this setting in which several candidate labels may be assigned to a single example. By extending three concrete classification methods to the ALC setting (nearest neighbor classification, decision tree learning, and rule induction) and evaluating their performance on benchmark data sets, we show that appropriately designed learning algorithms can successfully exploit the information contained in ambiguously labeled examples. Our results indicate that the fundamental idea of the extended methods, namely to disambiguate the label information by means of the inductive bias underlying (heuristic) machine learning methods, works well in practice.

Keywords: machine learning; classification; missing data; inductive bias; nearest neighbor estimation; decision trees; rule induction.

*Revised and extended version of a conference paper presented at IDA–05, 6th International Symposium on Intelligent Data Analysis, Madrid, Spain, 2005.

1 Introduction

One of the standard problems in (supervised) machine learning is inducing a *classification function* from a set of training data, i.e., a function which assigns objects to classes or categories. In fact, a large repertoire of corresponding classification methods is now available.

Usually, the training data consists of a set of *labeled examples*, i.e., a set of objects (instances) whose correct classification is known. Over the last years, however, several variants of the standard classification setting have been considered. For example, in *multi-label classification* a single object can have several labels (belong to several classes), that is, the labels (classes) are not mutually exclusive [25]. In *semi-supervised learning*, only a part of the objects in the training set is labeled [1]. In *multiple-instance learning*, a positive or negative label is assigned to a so-called *bag* rather than to an object directly [8]. Each bag is a collection of several instances. A bag is labeled negative if all the instances it contains are negative, otherwise it is labeled positive. Given a set of labeled bags, the task is to induce a model that will label unseen bags and instances correctly.

In this paper, we are concerned with another extension of the standard classification setting that has recently been introduced in [16, 19], and that we shall subsequently refer to as *ambiguous label classification* (ALC). In this setting, an example might be labeled in a non-unique way by a *subset* of classes, just like in multi-label classification. In ALC, however, the existence of a (unique) *correct* classification is assumed, and the labels are simply considered as *candidates*.

In [16, 19], the authors rely on probabilistic methods in order to learn a classifier in the ALC setting. The approach presented in this paper can be seen as an alternative strategy which is more in line with standard (heuristic) machine learning methods. Our idea is to exploit the inductive bias underlying these methods in order to disambiguate label information. This idea, as well as the relation between the two approaches, is discussed in more detail in section 3. Before, the problem of ALC is introduced in a more formal way (section 2). In section 4, three concrete methods for ALC are proposed, namely

extensions of nearest neighbor classification, decision tree learning, and rule induction. Experimental results are presented in section 5. In section 6, wie briefly outline two alternative applications of ALC. The paper concludes with a summary of the results in section 7.

2 Ambiguous Label Classification

Let $\mathcal{X}$ denote an instance space, where an instance typically (though not necessarily) corresponds to the attribute–value description x of an object. In this case, $\mathcal{X} = X_1 \times X_2 \times \ldots \times X_\ell$, with $X_\imath$ the domain of the $\imath$-th attribute, and an instance is represented as a vector $x = (x^1 \ldots x^\ell) \in \mathcal{X}$. Moreover, let $\mathcal{L} = \{\lambda_1 \ldots \lambda_m\}$ be a set of labels (classes). Training data shall be given in the form of a set $\mathcal{D}$ of examples $(x_\imath, L_{x_\imath})$, $\imath = 1 \ldots n$, where $x_\imath = (x_\imath^1 \ldots x_\imath^\ell) \in \mathcal{X}$ and $L_{x_\imath} \subseteq \mathcal{L}$ is a set of candidate labels associated with instance $x_\imath$. $L_{x_\imath}$ is assumed to contain the true label $\lambda_{x_\imath}$, and $x_\imath$ is called an *ambiguous* example if $|L_{x_\imath}| > 1$. Note that this includes the special case of a completely unknown label ($L_x = \mathcal{L}$), as considered in semi-supervised learning. Here, however, we usually have the case in mind where $1 \leq |L_x| < |\mathcal{L}|$. For example, in molecular biology the functional category of a protein is often not exactly known, even though some alternatives can definitely be excluded [11].

The learning task is to select, on the basis of $\mathcal{D}$, an optimal model (hypothesis) $h : \mathcal{X} \to \mathcal{L}$ from a hypothesis space $\mathcal{H}$. Such a model assigns a (unique) label $\lambda = h(x)$ to any instance $x \in \mathcal{X}$. Optimality usually refers to *predictive accuracy*, i.e., an optimal model is one that minimizes the expected loss (risk) with respect to a given loss function $\mathcal{L} \times \mathcal{L} \to \mathbb{R}$.

3 Learning from Ambiguous Examples

Needless to say, even ambiguous data may comprise important information. The approach proposed in this paper is based on the observation that the benefit of this information might indeed be especially high if it is considered, not as an isolated piece of knowledge,

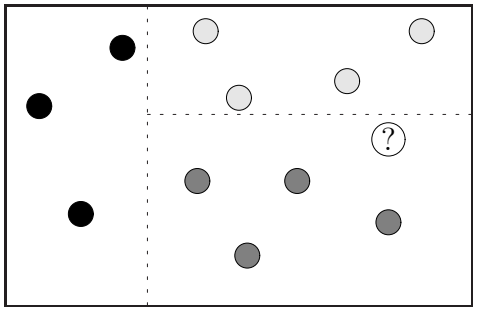

Figure 1: Classification problem with three labels: black (λ_1), grey (λ_2), light (λ_3). The instance with a question mark is either black or grey. Assigning label grey allows one to fit a very simple decision tree (as represented by the axis-parallel decision boundaries). Note that this hypothetical labeling also provides important information on the decision boundary between the grey and light class.

but in conjunction with the other data and the model assumptions underlying the hypothesis space $\mathcal{H}$. To illustrate this important point, consider a simple example in which the true label λ_{x_i} of an instance x_i is known to be either λ_1 or λ_2. Moreover, we seek to fit a classification tree to the data, which basically amounts to assuming that $\mathcal{X}$ can be partitioned by axis-parallel decision boundaries. Now, by setting $\lambda_{x_i} = \lambda_2$ we might find a very simple classification tree for the complete data, while $\lambda_{x_i} = \lambda_1$ requires a comparatively complex model (see Fig.1). Relying on the simplicity heuristic underlying most machine learning methods [9], this finding clearly suggests that $\lambda_{x_i} = \lambda_2$. In other words, looking at the original information $\lambda_{x_i} \in \{\lambda_1, \lambda_2\}$ with a view that is "biased" by the model assumptions, namely with the eyes of a decision tree inducer, the benefit of this information has highly increased.

As can be seen, the inductive bias underlying the learning process can help to *disambiguate* the label information given. This shows that ambiguous label information might indeed be useful and, in particular, suggests that it might be easier for learning methods with a strong inductive bias to exploit such information than for methods with a weak bias. Both these conjectures will be supported by our experimental results in section 5.

As an aside, note that not only the model assumptions might provide evidences for the true class label, but also vice versa: Once the label information has been disambiguated, it gives in turn information about the true model. In Fig. 1, for example, hypothetical

labeling (grey) provides important information on the decision boundary between the grey and light class.

The above example has shown that candidate labels can appear more or less likely against the background of the underlying model assumptions. In fact, the insight that fitting a model to the data might change the likelihood of candidate labels can be formalized more rigorously in a probabilistic context. Assuming a parameterized model M_θ, the goal can roughly be stated as finding the parameter

$$\theta^* \;=\; \arg\max_{\theta} \prod_{\imath=1}^{n} \Pr(\lambda_{x_\imath} \in L_{x_\imath} \,|\, x_\imath, \theta).$$

This approach gives rise to an EM (expectation-maximization) approach in which model adaptation and modification of label information are performed alternately: Given a distribution $D_\imath$ over each label set $L_{x_\imath}$, an optimal parameter θ^* is determined. Using this parameter resp. the associated model M_{θ^*}, the probabilities of the labels $\lambda \in L_{x_\imath}$ are then re-estimated. This process of estimating parameters and adjusting probabilities starts with the uniform distribution and iterates until convergence is eventually achieved [16, 19].

On the one hand, this approach is rather elegant and first empirical evidence has been gathered for its practical effectiveness [16, 19]. On the other hand, the assumption of a parameterized model basically restricts its applicability to statistical classification methods. Moreover, model optimization by means of EM can of course become quite costly from a computational point of view. Our idea of disambiguating label information by implementing a simplicity bias can be seen as an alternative strategy. As heuristic machine learning in general, this approach is of course theoretically not as well-founded as probabilistic methods. Still, heuristic methods have been shown to be often more effective and efficient in practical applications.

Given ambiguous label information, the EM approach estimates a model by means of likelihood maximization. Analogously, our approach looks for a model that is as much as possible in agreement with the label information. However, instead of likelihood maxi-

mization, it employs a heuristic strategy to induce that model. Here, the implicit assumption is that, if two models are both compatible with the data, the simpler one is more likely than the complex one. In the re-estimation step, the EM approach assigns new probabilities to the candidate labels, whereas our approach is more drastic and makes a definite decision in favor of one label. As a consequence, our approach does not require an iteration of the process.

Unfortunately, standard machine learners generally cannot exploit the information provided by ambiguous data, simply because they cannot handle such data. This is one motivation underlying the development of methods for ALC (as will be done in section 4). Note that a straightforward strategy for realizing ALC is a reduction to standard classification: Let the class of *selections*, $\mathcal{F}(\mathcal{D})$, of a set $\mathcal{D}$ of ambiguous data be given by the class of compatible (standard) samples

$$\mathcal{S} = \{(x_1, \alpha_{x_1}), (x_2, \alpha_{x_2}), \ldots, (x_n, \alpha_{x_n})\} \tag{1}$$

such that $\alpha_{x_\imath} \in L_{x_\imath}$ for all $1 \leq \imath \leq n$. In principle, a standard learning method could be applied to all samples $\mathcal{S} \in \mathcal{F}(\mathcal{D})$, and an apparently most favorable model could be selected among the models thus obtained. However, since the number of selections, $|\mathcal{F}(\mathcal{D})| = \prod_{\imath=1}^{n} |L_{x_\imath}|$, will usually be huge, this strategy is of course not practicable.

4 Methods for ALC

In this section, we present three relatively simple extensions of standard learning algorithms to the ALC setting, namely k-nearest neighbor classification, decision tree learning, and rule induction.

4.1 Nearest Neighbor Classification

In k-nearest neighbor (k-NN) classification [7], the label $\lambda_{x_0}^{est}$ assigned to a query x_0 is given by the label that is most frequent among x_0's k nearest neighbors, where nearness

is measured in terms of a similarity or distance function. In weighted k-NN, the neighbors are moreover weighted by their distance [34]:

$$\lambda_{x_0}^{est} \stackrel{\text{df}}{=} \arg\max_{\lambda \in \mathcal{L}} \sum_{\imath=1}^{k} \omega_\imath \, \mathbb{I}(\lambda = \lambda_{x_\imath}), \tag{2}$$

where $x_\imath$ is the $\imath$-th nearest neighbor; $\lambda_{x_\imath}$ and $\omega_\imath$ are, respectively, the label and the weight of $x_\imath$, and $\mathbb{I}(\cdot)$ is the standard $\{\text{true}, \text{false}\} \to \{0, 1\}$ mapping. A simple definition of the weights is $\omega_\imath = 1 - d_\imath \cdot (\sum_{\jmath=1}^{k} d_\jmath)^{-1}$, where the $d_\imath$ are the corresponding distances.

Now, a relatively straightforward generalization of (2) to the ALC setting is to simply replace $\mathbb{I}(\lambda = \lambda_{x_\imath})$ by $\mathbb{I}(\lambda \in L_{x_\imath})$:

$$\lambda_{x_0}^{est} \stackrel{\text{df}}{=} \arg\max_{\lambda \in \mathcal{L}} \sum_{\imath=1}^{k} \omega_\imath \, \mathbb{I}(\lambda \in L_{x_\imath}). \tag{3}$$

Thus, a neighbor $x_\imath$ is allowed not one single vote only, but rather one vote for each acceptable label. If the maximum in (3) is not unique, one among the labels with highest score is simply chosen at random.[1]

4.2 Decision Tree Induction

Another standard learning method, whose extension to the ALC setting might be of interest, is decision tree induction [30, 2]. The basic strategy of decision tree induction is to partition the data in a recursive manner. This strategy can of course be maintained for ALC. Besides, with regard to the stopping condition of the recursive partitioning scheme, note that a further splitting of a (sub)set of examples $\mathcal{D}$ is not necessary if

$$L(\mathcal{D}) \stackrel{\text{df}}{=} \bigcap_{x_\imath \in \mathcal{D}} L_{x_\imath} \neq \emptyset, \tag{4}$$

hence, (4) defines a natural stopping condition. The corresponding node in the decision tree then becomes a leaf, and any label $\lambda \in L(\mathcal{D})$ can be chosen as the prescribed label associated with that node. Finally, pruning a fully grown tree can principally be done

[1] A reasonable alternative is to choose the prevalent class in the complete training set.

in the same way as pruning standard trees. In our implementation, we used the pruning technique that is also implemented in C4.5 [30].

The main modification of the standard algorithm concerns the "goodness of split" measure. In decision tree induction, such measures are used to select an attribute according to which the data is partitioned. Roughly speaking, the idea is to make a set of examples (or, more precisely, the subsets of examples obtained after splitting) as "pure" as possible in terms of the class distribution. A commonly used measure of purity is the entropy of this distribution.

But what is the purity, or rather the decision capability of a set of ALC-examples $\mathcal{D}$? Let $q_\imath$ denote the frequency of the label $\lambda_\imath$ in the set of examples $\mathcal{D}$, i.e., the number of examples $(x_\jmath, L_{x_\jmath})$ such that $\lambda_\imath \in L_{x_\jmath}$. Thus defined, the $q_\imath$ do obviously satisfy the inequality

$$q_1 + \ldots + q_m \geq 1. \tag{5}$$

However, (5) does no longer hold with an equality, as it does in the case where each example has a unique label. Formally, the standard entropy measure

$$\mathrm{H}(\mathcal{D}) = -\sum_{\imath=1}^{m} q_\imath \cdot \log_2(q_\imath) \tag{6}$$

and, hence, the common information gain criterion could still be applied. However, entropy has a meaningful interpretation only for probability distributions. Apart from that, this measure does not appear reasonable in our context: The stopping condition (4) obviously suggests a measure that is monotone increasing in the $q_\imath$, since the larger the $q_\imath$, the higher the chance to satisfy the condition of a non-empty intersection (after few further splits). As opposed to this, the entropy measure does not only reward high values $q_\imath$ (close to 1), but also small values close to 0.

Inequality (5) and the fact that the $q_\imath$ can be considered as *upper bounds* to the true relative frequencies of the labels $\lambda_\imath$ might suggest looking at generalized uncertainty calculi such as, e.g., evidence theory or possibility theory [10]. For example, plausibility

degrees in evidence theory can also be interpreted as upper probability bounds, the sum of which is only lower-bounded by 1. And indeed, entropy-like measures, just like other measures of uncertainty originally established within the framework of probability, have been generalized to alternative frameworks such as evidence theory and possibility theory [23]. These are, however, either measures of non-specificity or measures of conflict, and none of these types of measures is actually useful in our context.

The perhaps simplest monotone measure is

$$\max(q_1 \dots q_m), \tag{7}$$

and this measure is indeed interesting since satisfaction of condition (4) is equivalent to $\max(q_1 \dots q_m) = 1$. A drawback of the measure is of course that it is completely determined by only one of the $q_\imath$. Thus, it completely ignores the remaining values and is hence not very discriminative. For example, it does not distinguish between the distributions $(.1, .1, .1, .7)$ and $(0, 0, .7, .7)$, since the maximum is in both cases .7.

This problem could be avoided by a measure that indeed favors distributions with large values $q_\imath$ but still takes the remaining values into account, at least to some extent. This property is satisfied by so-called *ordered weighted average* (OWA) operators [35] which derive a weighted average of m values *after reordering* them. Thus, given weights $w_1 \dots w_m$, the highest value is always weighted by w_1, the second-highest by w_2, and so on. As a particular OWA operator of that type we used

$$\mathrm{O}_\gamma \stackrel{\mathrm{df}}{=} \frac{\sum_{\imath=1}^{m} \gamma^{m-1}\, r_\imath}{(1-\gamma)^m/(1-\gamma)}, \tag{8}$$

where $0 \leq \gamma < 1$ and $r_\imath$ is the $\imath$–th largest value in the sequence $q_1 \dots q_m$; the maximum operator (7) is obviously a special case of (8), as it is recovered for $\gamma = 0$.

As a further alternative, more directly related to the standard entropy, we considered a

measure of "potential entropy" which is defined by

$$\mathrm{H}^*(\mathcal{D}) \stackrel{\mathrm{df}}{=} \min_{\mathcal{S}\in\mathcal{F}(\mathcal{D})} \mathrm{H}(\mathcal{S}), \tag{9}$$

where $\mathcal{F}(\mathcal{D})$ is the set of selections (1). As can be seen, (9) is the standard entropy obtained for the most favorable instantiation of the ALC-examples $(x_\imath, L_{x_\imath})$. It corresponds to the "true" entropy that would have been derived if this instantiation was compatible with the ultimate decision tree. Taking this optimistic attitude is justified since the tree is indeed constructed in a hopefully optimal manner.

Computing (9) comes down to solving a combinatorial optimization problem and becomes intractable for large samples. Therefore, we devised two heuristic approximations of (9). The first approximation is

$$\mathrm{H}^*_1(\mathcal{D}) \stackrel{\mathrm{df}}{=} \mathrm{H}(\mathcal{S}^*), \tag{10}$$

where the selection $\mathcal{S}^*$ is defined as follows: The labels $\lambda_\imath$ are first put in a (total) "preference" order according to their frequency: $\lambda_\imath$ is preferred to $\lambda_\jmath$ if $q_\imath > q_\jmath$ (ties are broken by coin flipping). Then, the most preferred label $\lambda_\imath \in L_{x_\imath}$ is chosen for each example $x_\imath$. Clearly, the idea underlying this selection is to make the distribution of labels as skewed (non-uniform) as possible, as distributions of this type are favored by the entropy measure.

The second approximation is defined as follows: Without loss of generality, suppose $q_1 \geq q_2 \geq \ldots \geq q_m$. Then,

$$\mathrm{H}^*_2(\mathcal{D}) \stackrel{\mathrm{df}}{=} -\sum_{\imath=1}^{m} q'_\imath \log_2(q'_\imath), \tag{11}$$

where $q'_\imath = q_\imath$ if $q_1 + \ldots + q_\imath \leq 1$ and $= \max(1 - (q_1 + \ldots + q_{\imath-1}), 0)$ otherwise. Note that (11) may underestimate the potential entropy (there might be no selection that agrees with the modified frequencies $q'_\imath$), whereas (10) is provably an upper bound.

To evaluate and compare the performance of the different splitting measures introduced

above, we have conducted a number of controlled experiments using synthetic data. This data was derived on the basis of randomly generated decision trees that serve as reference models. For each experiment, a decision tree of this kind is generated as follows: In a first step, a standard tree is grown in a recursive manner, starting with the root of the tree and flipping a (biased) coin to decide whether the current node becomes an inner node or a leaf. The probability of a node to become an inner node is given by $0.8 - t/10$, where t is the depth of the node. Once a leaf node has been generated, a decision $\lambda \in \{\lambda_1 \dots \lambda_m\}$ is assigned to that node at random. Likewise, each inner node is assigned one among k possible attributes at random. We let each attribute X_j assume the v values $x_j \in \{1 \dots v\}$, for fixed $v \in \mathbb{N}$, so that an inner node does always have v successors.

In a second step, additional labels are assigned to the leaf nodes of the tree. More precisely, at every leaf node, the labels are added independently of each other with a fixed probability p. Finally, it is checked whether the expanded tree thus obtained can be simplified, i.e., whether one of the inner nodes satisfies the stopping condition (4) and should hence become a leaf.

A set of training examples is then generated by choosing 1,000 instances x at random, using a uniform distribution over $\mathcal{X}$. The labels L_x of these examples are derived from the above reference tree. For the training examples, a decision tree is induced using one of the splitting measures discussed above. The performance of a measure is quantified in terms of the ratio between the complexity of the induced tree (number of leaf nodes) and the complexity of the reference tree.[2] The expected performance of each splitting measure is approximated by averaging over 2,000 experiments.

The tables 1–4 show the experimental results for different settings of the parameters k, m, v, p (recall that k = number of attributes, m = number of labels, v = number of values per attribute, p = probability of adding a split; qualitatively similar results were obtained for various other settings, not presented here due to reasons of space). More precisely, each table shows the results for the different splitting measures discussed above and different parameters p. For comparison purpose, and despite of the aforementioned

[2]Note that this ratio can thoroughly assume values < 1, since the training examples might be explained by a model which is simpler than the model that has generated the data.

p	0.00	0.20	0.40	0.60	0.80
H_1^*	**1.01**	**1.05**	**1.18**	**1.27**	**1.14**
H_2^*	1.01	1.24	1.48	1.36	1.15
H	1.01	1.28	1.73	1.90	1.63
O_0	1.62	1.64	1.54	1.46	1.22
$O_{0.2}$	1.40	1.42	1.44	1.40	**1.20**
$O_{0.4}$	1.30	1.33	1.39	**1.38**	1.22
$O_{0.6}$	1.22	**1.26**	**1.38**	1.45	1.29
$O_{0.8}$	**1.17**	1.28	1.51	1.68	1.47

Table 1: Setting $k = 12$, $m = 10$, $v = 3$.

p	0.00	0.20	0.40	0.60	0.80
H_1^*	**0.94**	**0.97**	**0.99**	**0.89**	**0.64**
H_2^*	0.94	1.01	1.05	0.91	0.63
H	0.94	1.24	1.40	1.27	0.93
O_0	1.09	1.10	**1.07**	**0.94**	**0.66**
$O_{0.2}$	1.04	1.09	1.08	0.96	0.66
$O_{0.4}$	1.01	**1.06**	1.08	0.97	0.67
$O_{0.6}$	0.98	1.07	1.11	1.01	0.71
$O_{0.8}$	**0.97**	1.12	1.20	1.11	0.80

Table 2: Setting $k = 12$, $m = 10$, $v = 4$.

semantic problems, we also included results for the standard entropy formula (6) applied to the values q_i.

The results clearly show that the measure H_1^* consistently outperforms the other candidates. Moreover, this measure yields models that are only slightly more or even less complex than the reference models. This suggests that H_1^* is indeed a very good measure, in fact not only in comparison to the alternatives but also in view of "absolute" standards.

p	0.00	0.20	0.40	0.60	0.80
H_1^*	**1.01**	**1.03**	**1.12**	**1.22**	**1.12**
H_2^*	1.01	1.12	1.43	1.32	1.13
H	1.01	1.30	1.78	1.91	1.62
O_0	1.49	1.45	1.45	1.38	1.19
$O_{0.2}$	1.31	1.30	1.35	1.32	**1.16**
$O_{0.4}$	1.20	1.21	1.29	**1.30**	1.17
$O_{0.6}$	1.12	**1.15**	**1.27**	1.35	1.23
$O_{0.8}$	**1.08**	1.18	1.43	1.59	1.43

Table 3: Setting $k = 12$, $m = 15$, $v = 3$.

p	0.00	0.20	0.40	0.60	0.80
H_1^*	**1.01**	**1.04**	**1.18**	**1.26**	**1.15**
H_2^*	1.01	1.19	1.52	1.38	1.17
H	1.01	1.33	1.89	2.03	1.73
O_0	1.71	1.65	1.61	1.51	1.25
$O_{0.2}$	1.46	1.46	1.49	1.42	**1.23**
$O_{0.4}$	1.32	1.35	**1.42**	**1.42**	1.25
$O_{0.6}$	1.24	**1.29**	1.43	1.51	1.32
$O_{0.8}$	**1.16**	1.31	1.60	1.77	1.54

Table 4: Setting $k = 18$, $m = 10$, $v = 3$.

In our ALC-extension of decision tree learning, we will hence employ this measure.

As it was to be expected, the standard entropy H yields good results for p close to 0 but quickly breaks down as p increases. The results of the best OWA-measures are somewhat comparable to those of H_2^*. As a disadvantage, however, note that the performance strongly depends on the parameter γ which has to be adjusted to the parameter p: The larger p, the smaller γ should be chosen.

4.3 Rule Induction

Since each path from the root of a decision tree to a leaf node defines a single rule, a decision tree can be considered as a set of rules having a particular structure. These rules are induced by following a *divide-and-conquer* strategy. An alternative approach is to learn rules in a more direct way, using a *separate-and-conquer* or *covering* strategy [14]. Separate-and-conquer rule learning constitutes another important class of machine learning algorithms. Concrete implementations include the AQ-family [26, 27], CN2 [5, 4], Ripper [6] and Foil [29].

In order to learn a concept, i.e., to separate positive from negative examples, covering algorithms learn one rule after another. Each rule *covers* a subset of (positive) examples, namely those that satisfy the condition part of the rule (usually a conjunction of *selectors* of the form $attribute = value$). The covered examples are then removed from the training set. This process is iterated until no positive examples remain. Covering algorithms can be extended to the m-class case ($m > 2$) in several ways. For example, following a

one-versus-rest strategy, CN2 learns rules for each class in turn, starting with the least frequent one. Since in the m-class case the order in which rules have been induced is important,[3] the rules thus obtained have to be treated as a *decision list*.

A key component of all covering algorithms is a "find-best-rule" procedure for finding a good or even optimal rule that partly covers the current training data. Starting with a maximally general rule, CN2 follows a top-down approach in which the the candidate rules are successively specialized (e.g. by adding conditions). The search procedure is implemented as a beam search, guided by the Laplace-estimate as a heuristic evaluation:

$$L(r) \stackrel{\text{df}}{=} \frac{p+1}{n+p+2}, \tag{12}$$

where r is the rule to be evaluated, p is the number of positive examples covered by r, and n the number of negative examples. As a stopping criterion, CN2 employs a statistical significance test (likelihood ratio) that decides whether or not the distribution of positive and negative examples covered by the rule is significantly different from the overall distribution in the complete training set.

In order to adapt CN2 to the ALC setting, we have made the following modifications: Motivated by our previous findings for generalizations of the entropy measure, we have turned the Laplace-estimate into a "potential" Laplace-estimate: Considering label $\lambda_\jmath$ as the positive class, $p = p_\jmath$ is given by the number of all examples $x_\imath$ covered by the rule r and such that $\lambda_\jmath \in L_{x_\imath}$. This way, (12) can be derived for each label, and the maximal value is adopted as an evaluation of the rule:

$$L(r) = \max_{1 \leq \jmath \leq m} \frac{p_\jmath + 1}{|r| + 2},$$

where $|r|$ is the number of examples covered by the rule. The consequent of r is then given by the label $\lambda_\jmath$ for which the maximum is attained.

As noted before, CN2 learns classes in succession, starting with the smallest (least frequent) one. As opposed to this, we learn rules without specifying a class in advance.

[3] Conflicts are possible due to rules with different consequences and overlapping condition parts.

Rather, the most suitable class is chosen adaptively, depending on the condition part of a rule. In fact, the label predicted by a rule can even change during the search process. This modification is in agreement with our goal of disambiguating by implementing a simplicity bias. Moreover, the focusing on one particular label is less useful in the ALC setting. In fact, in the presence of ambiguously labeled examples, it may easily happen that a rule r is dominated by a class λ_j while all of its direct specializations are dominated by other classes.

5 Experimental Results

The main purpose of our experimental study was to provide evidence for the conjecture that a suitable ALC-method is able to exploit the information contained in ambiguously labeled examples. More specifically, the goal is to show that using an ALC-method is usually better than the obvious alternative, namely to ignore ambiguous data and learn with a standard algorithm from the remaining (exactly labeled) examples. We hence used the latter approach as a baseline method.

Note that this conjecture is by far not trivial. In fact, whether or not ambiguous data can be useful will strongly depend on the performance of the ALC-method. If this method is not able to exploit the information contained in that data, ambiguous examples might be misleading rather than helpful. In this connection, recall our supposition that the weaker the inductive bias of a learning method, the more likely that method might be misled by ambiguous examples.

5.1 Experimental Setup

We have worked with "contaminated" versions of standard benchmark data sets (in which each instance is assigned a unique label), which allowed us to conduct experiments in a controlled way. In order to contaminate a given data set, we have devised two different strategies:

- **Random model:** For each example in the training set, a biased coin is flipped in order to decide whether or not this example will be contaminated; the probability of contamination is p. In case an example x_i is contaminated, the set L_{x_i} of candidate labels is initialized with the original label λ_{x_i}, and all remaining labels $\lambda \in \mathcal{L} \setminus \{\lambda_{x_i}\}$ are added with probability q, independently of each other. Thus, the contamination procedure is parameterized by the probabilities p and q, where p corresponds to the expected fraction of ambiguous examples in a data set. Moreover, q reflects the "average benefit" of a contaminated example x_i: The smaller q is, the smaller the (average) number of candidate labels becomes and, hence, the more informative such an example will be. In fact, note that the expected cardinality of L_{x_i}, in the case of contamination, is given by $1 + (m-1)q$.

- **Bayes model:** The random model implicitly assumes that candidate labels occur independently of each other. In practice, this idealized assumption will rarely be valid. For example, the probability that a label is added will usually depend on the true label. In order to take this type of dependency into account, our second approach to contamination works as follows: First, a Naive Bayes classifier is trained using the original data, and a probabilistic prediction is derived for each input x_i. Let $\Pr(\lambda \,|\, x_i)$ denote the probability of label λ as predicted by the classifier. Whether or not an example is contaminated is decided by flipping a biased coin as before. In the case of contamination, the true label λ_{x_i} is again retained. Moreover, the other $m-1$ labels $\lambda \in \mathcal{L} \setminus \{\lambda_{x_i}\}$ are arranged in an increasing order according to their probability $\Pr(\lambda \,|\, x_i)$. The k-th label in this order, $\lambda^{(k)}$, is then added with probability $(2 \cdot k \cdot q)/m$. Thus, the expected cardinality of L_{x_i} is again $1 + (m-1)q$, but the probabilities of the individual labels are now biased in favor of the labels found to be likely by the Bayes classifier.

Intuitively, the Bayes model should come along with a decrease in performance for the ALC approach (while the baseline method is of course not affected). Roughly speaking, since the Bayes model contaminates the data in a more "systematic" way, one might expect that disambiguating the data becomes more difficult. To illustrate, consider the

extreme example where the true class is always λ_1, i.e., all instances do have label λ_1. Now, if additional labels are added in a purely random way, most learning methods will still be able to recognize the default rule $\top \Rightarrow \lambda_1$ as a model. When always adding label λ_2, however, a distinction between the correct model and the one that always predicts λ_2 becomes impossible.

The experimental results have been obtained in the following way: In a single experiment, the data is randomly divided into a training set and a test set of the same size. The training set is contaminated as outlined above. From the contaminated data, a model is induced using an ALC-extension of a classification method (kNN, decision trees, rule induction). Moreover, using the classification method in its standard form, a model is learned from the reduced training set that consists of the non-contaminated examples. Then, the classification accuracy of the two models is determined by classifying the instances in the test set. The *expected* classification accuracy of a method – for the underlying data set and fixed parameters p, q – is approximated by averaging over 1,000 such experiments. This was done for a number of benchmark data sets from the UCI repository.

For decision tree learning and rule induction, all numeric attributes have been discretized in advance using hierarchical entropy-based discretization [12].[4] For the NN classifier, we used the simple Euclidean distance measure, and we didn't include feature selection or feature weighting (even though it is well-known that irrelevant features can badly deteriorate and, on the other hand, that feature weighting can greatly improve performance [32]). In fact, we didn't try to optimize the performance of the three learning methods themselves, because this was not the goal of the experiments. Rather, the purpose of the study was to *compare* – under equal conditions – ALC learning with the baseline method. Below, we present results for the five UCI data sets[5] in table 5 and a few combinations of (p, q)-parameters; qualitatively quite similar results have been obtained for other data sets.

[4]Discretization was done in advance for the complete data set, not separately for each training set.

[5]The output attribute in the Housing data is actually a numeric variable (price of a house). We discretized this attribute using an equi-width partition of size 10.

	name	instances	attributes	classes
1	dermatology	385	34	6
2	ecoli	336	7	8
3	housing	506	13	10
4	glass	214	9	6
5	zoo	101	16	7

Table 5: Data sets used in the experiments.

5.2 NN Classification

Regarding NN classification, it is quite obvious that (3) will perform poorly in the case $k = 1$. In fact, the 1-NN rule simply amounts to guessing one among the labels suggested by the nearest neighbor, so the probability of a correct classification will be low if this neighbor is contaminated. The expected classification rate of 1-NN can be calculated exactly for our random model: If the original classification rate (for the non-contaminated data) is c, then the rate after contamination becomes[6]

$$c - pc\left(1 - \frac{1-(1-q)^m}{mq}\right) + \frac{p(1-c)}{m-1}\left(1 - \frac{1}{mq} + (1-q)^m\right)$$

Note that this rate is linear decreasing in p. This result was confirmed by our experiments. For example, Fig. 2 shows results for the GLASS data. The classification accuracy for $q = 0.3$ is plotted as a function of the parameter p.[7] As can be seen, in the case $k = 1$ the standard classifier is obviously superior to the ALC-version. In fact, the accuracy of the standard classifier remains surprisingly high, a visible deterioration only occurs when p becomes large.

This changes, however, when increasing the neighborhood size k. In fact, the standard version becomes inferior to the ALC-version for $k = 5$ already. Intuitively, this can be explained by the fact that the larger the number of neighbors, the smaller the probability that the top-label suggested by these neighbors through majority vote will be changed by contamination. Roughly speaking, relying on a nearby contaminated neighbor is better

[6] We omit the derivation which is rather lengthy.

[7] In order to obtain a smooth curve, we carried out 50,000 instead of only 1,000 experiments. Since the approximations are almost exact, we plotted the averages without standard deviations.

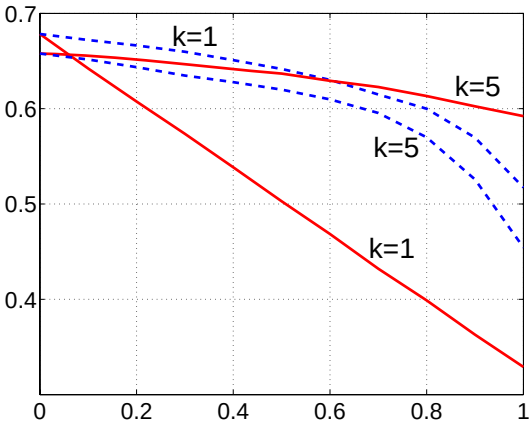

Figure 2: Classification accuracy (y-axis) for the Glass data as a function of the "contamination" p (x-axis): NN classification with $k = 1$ and $k = 5$, using the standard (dashed, blue lines) and ALC-version (solid, red lines).

than looking at a non-contaminated example that is faraway (because the close ones have been removed).

The results for k-NN classification with $k = 5$ are summarized in table 6, where (r) stands for the random model and (b) for the Bayes model. As can be seen, the ALC version is generally superior to the standard 5-NN classifier. Exceptions (marked with a *) only occur in cases where both p and q are large, that is, where the data is strongly contaminated. We obtained similar results for $k = 7, 9, 11$.

The results do not convincingly confirm the supposition that the ALC version will perform better for the random model than for the Bayes model. Even though it is true that the results for the former are better than for the latter in most cases, the corresponding difference in performance is only slight and much smaller than expected. In general, it can be said that the contamination model does hardly influence the performance of the classifier most of the time. In fact, there is only one noticeable exception: For the Zoo data, the performance for the random model is much better than for the Bayes model in the case of highly contaminated data.

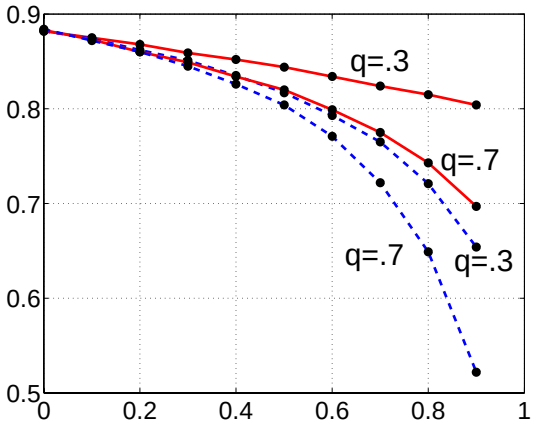

Figure 3: Classification accuracy (y-axis) for the Zoo data as a function of the "contamination parameter " p (x-axis): Standard decision tree induction (dashed, blue lines) and ALC-version for the random model (solid, red lines).

5.3 Decision Tree Induction

For decision tree induction, the ALC-version consistently outperforms the standard version. In fact, as the results in table 7 show, the difference between the ALC-version and the standard version is even larger than for NN classification (see also Fig. 3). This confirms our conjecture that the stronger the inductive bias underlying the generalizer, the more successful ALC will be.

Again, the results also show that a systematic contamination of the data, using the Bayesian instead of the random model, does hardly affect the performance of ALC classification. It is true that the classification performance deteriorates on average, but again only slightly and not in every case.

An interesting exception to the above findings is the Housing data (indeed, not only for decision tree learning but also for NN classification, cf. table 6). First, for this data the standard version is down the line better than the ALC-version. Second, the ALC-version is visibly better in the case of the Bayesian model than in the case of the random model. The most plausible explanation for this is the fact that for the Housing data the classes are price categories and hence do have a natural *order*. That is, we actually face a problem of *ordinal classification* rather than standard classification.[8] Moreover, the Bayesian model

[8]Consequently, ordinal classification methods [17, 24, 28, 13, 3] should be applied, and the results for this data set should not be overrated in our context.

tends to add classes that are, in the sense of this ordering, neighbored to the true price category, thereby distorting the original class information but slightly. Compared to this, ambiguous information will be much more conflicting in the case of the random model.

With regard to complexity, let us note that both approaches induce decision trees of approximately the same size. On average, the ALC-tree has three leaves more than the standard tree, probably due to the fact that it learns on larger training sets.

5.4 Rule Induction

The experimental results for rule induction are summarized in table 8. Again, the ALC version outperforms the standard version most of the time, especially for large p-values. Since the results are rather similar to those for decision tree learning, we refrain from a more detailed discussion.

5.5 Summary of Experimental Results

In summary, the experiments show that our ALC extensions of standard learning methods can successfully deal with ambiguous label information. In fact, except for some rare cases, these extensions yield better results (in terms of classification performance) than the baseline method (which ignores ambiguous examples and applies standard learning methods).

Fig. 4 provides a graphical illustration of the gain in classification accuracy for the ALC approach in comparison with the baseline method.[9] This statistic is interesting for at least two reasons. Firstly, it suggests that the gain is a monotone increasing function of the parameter p (probability of contamination). With regard to the parameter q, however, the dependency appears to be non-monotone: The gain first increases but then decreases for large enough q-values. These findings are intuitively plausible. In fact, since q represents a kind of "expected benefit" of an ambiguous example, the utility of such an example is likely to become negative for large q-values. Consequently, it might then be better to

[9] We have omitted the Housing data due to the aforementioned reasons.

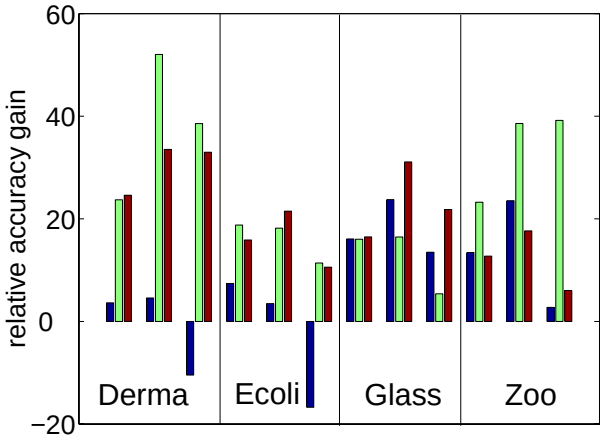

Figure 4: Gain in classification accuracy (percentage) for the ALC approach (random model) in comparison with the baseline method. A block of three bars shows the gain for NN classification, decision tree learning, and rule learning (from left to right). The first, second, and third block correspond, respectively, to the q-values .3, .5, and .7. The p-value is always .9.

simply ignore such examples, at least if enough other examples are available.

Secondly, the performance gain for decision tree learning seems to be slightly higher than the one for rule induction, at least on average, and considerably higher than the gain for NN classification. This ranking is in perfect agreement with our conjecture that the stronger the inductive bias of a learning method, the more useful ALC will be.

6 Alternative Applications

In this section, we briefly outline two alternative applications for which ALC methods might potentially be useful. However, we only give a sketch of the basic ideas, as an in-depth investigation is beyond the scope of this paper.

6.1 Decision Making

In ALC, the labels L_x associated with an instance x are considered as alternatives including the correct classification. However, instead of considering L_x as an *ambiguous* labeling, one can also imagine an application scenario in which it characterizes a set of *admissible* alternatives. In that case, a distinguished correct label does simply not exist.

Instead, each $\lambda \in L_x$ is allowed, which means that the risk of a model $h : \mathcal{X} \to \mathcal{L}$ now becomes

$$\mathcal{R}(h) \stackrel{\text{df}}{=} \int_{\mathcal{X}} (1 - \mathbb{I}(h(x) \in L_x))\, d\,\mu, \tag{13}$$

where μ is a probability measure over $\mathcal{X}$.

In connection with the admissibility interpretation, an obvious concern is to exploit the additional freedom of having several instead of only one admissible label per instance in order to reduce the complexity of a model. Thus, just like in ALC, the problem can roughly be stated as inducing an as simple as possible model that is (to a large extent) consistent with the data. Note, however, that looking for a simple model has a completely different reason in ALC, where the idea is that a simpler model is also more likely to select the correct labels among the candidates. Nevertheless, the same learning algorithms could in principle be applied for both types of application, despite of the differences in the underlying semantic interpretations.

The admissibility semantics is natural in the case where labels represent *decisions* or actions rather than classes. More specifically, suppose that the instances $x \in \mathcal{X}$ encode decision problems, that is, situations in which an agent must choose one among a set $\mathcal{L}$ of potential actions. In this scenario, there will usually not exist one unique "correct" action for each decision problem x. Particularly, a "satisficing" decision maker in the sense of H. Simon [31] may be able to divide $\mathcal{L}$ into actions that are *acceptable* for a problem x and those that are *non-acceptable*. Here, it is assumed that different actions are more or less useful for a problem and that an action λ is acceptable for a problem x if the *utility* of applying λ to solve x exceeds a certain threshold. Thus, knowing that a set of actions L_x is acceptable for a problem x gives rise to an example (x, L_x). Examples of this type represent the agent's *experience*. By generalizing beyond this experience, the agent may try to induce a theory in the form of a decision model, which prescribes acceptable actions for all situations $x \in \mathcal{X}$. This kind of experience-based decision making is closely related to the idea of case-based decision theory [15].

The interesting point in connection with this application concerns the relation between the agent's "ambition" and the complexity of the decision model: Usually, the more ambitious the agent is (i.e., the larger the utility threshold), the more complex the decision model will be. To illustrate, consider the problem of choosing the dose of a drug for different patients. A simple decision model that prescribes different doses for males, females, and children might lead to acceptable results (the utility of a decision depends on the patient's state of health after the treatment), whereas optimizing the results might require a more complex model that differentiates more precisely between patients, e.g., by taking other attributes in addition to sex and age into account. Of course, a medical doctor might thoroughly prefer the simple model to a more complex (and maybe more expensive) one.

More generally, if an ALC method is available for inducing decision models, the utility threshold might be used as a parameter for controlling the tradeoff between the complexity of a model and its performance (decision quality): The smaller the threshold, the larger the admissible sets L_x, and the less complex the decision model [18].

6.2 Model Pruning

ALC might also be useful as a kind of pruning technique. The idea is as follows: Pruning a model such as, e.g., a decision tree or a rule-based classifier, roughly means simplifying that model at the cost of mis-classifying some of the examples in the training set. That is, the label assigned to these examples by the pruned model is different from the prescribed label. Now, if the final (mis-)labeling of an instance would have been allowed from the start, the pruned model might have been inferred directly by means of an ALC method. In other words, tolerating for some mis-classifications can be interpreted as a kind of pruning, at least if the learning method is able to utilize the additional flexibility in order to induce a simpler model. More specifically, it concerns a *pre*-pruning strategy, comparable to the use of evaluation measures in rule induction that allow a rule to cover some negative examples.

But how can one anticipate the mis-classifications? Or, stated differently, which labels should be added, i.e., which mis-classifications should be tolerated? Intuitively, the critical

examples that often lead to complex models are (i) noisy instances and outliers and (ii) instances located near a decision boundary. Since the label of such examples is usually different from the label of closely neighbored instances, we suggest the following approach (that requires a distance measure over $\mathcal{X}$): For an instance x_i, define L_{x_i} by the originally prescribed label λ_{x_i} plus the majority label among x_i's k nearest neighbors. In other words, the strategy is to be tolerant with examples whose k-nearest neighbor (k-NN) classification is not completely certain.

To illustrate the potential effectiveness of the idea, consider the following extreme example: Suppose that the true model can be represented by a simple decision tree. Unfortunately, the data $\mathcal{D}$ contains one instance x_i that is erroneously labeled with λ_1 instead of λ_2. Since decision tree induction is rather sensitive toward noisy data, the tree induced from $\mathcal{D}$ might be very complicated, and even post-pruning that tree might not recover the true model. Now, by including λ_2 as an alternative classification, as suggested by x_i's neighborhood, the true model might have been obtained directly (e.g., by means of the ALC-extension of decision tree induction as introduced in section 4).

ALC-pruning as outlined above is related to other pruning techniques. For example, Wilson [33] proposed a kind of *instance-based pruning* for the nearest neighbor method itself. From the original training set he completely removes all instances that do not agree with the majority of their k nearest neighbors. Likewise, the following procedure is proposed by John [20] for learning *robust* decision trees: A decision tree is first grown on the complete training data and then pruned by converting some selected inner nodes of the tree to leaf nodes (and, as usual, assigning majority classes as labels). After pruning, some of the examples will of course be mis-classified. Now, it is argued that pruning a node of a decision tree effectively removes all instances that were not in the majority class of the subset of examples used to build that node's subtree. In other words, these instances are "locally uninformative" or perhaps even harmful. Then, however, one might wonder why they should still have an influence on the global structure of the tree (as they do have even after pruning). Consequently, it is suggested to remove the mis-classified instances from the training set and then to retrain on the remaining data. This procedure

is iterated until pruning does not change the rebuilt decision tree.

Compared to the above strategies, our approach is less drastic: Critical or potentially misleading examples are not completely ignored. Rather, alternative classifications, suggested by the local neighborhood of an example, are tolerated so as to put a learning algorithm into the position to ignore a misleading labeling. In fact, ALC-pruning is also a kind of instance-based pruning technique, but one that does not prune by removing potentially harmful examples but rather by correcting possibly misleading label information.

Note that the inductive bias of the overall learning algorithm, including ALC-pruning as a preprocessing step, will be a combination of the k-NN bias and the bias of the actual learning method applied afterwards. Consequently, the actual learning method is to some extent biased by k-NN. However, since k-NN (with small k) has a rather weak bias, this effect is indeed limited.

7 Concluding Remarks

In order to successfully learn a classification function in the ALC setting, where examples can be labeled in an ambiguous way, we proposed several extensions of standard machine learning methods. The idea is to exploit the inductive bias underlying these (heuristic) methods in order to disambiguate the label information. In fact, our speculation was that looking at the label information with a "biased view" may remove the ambiguity of that information to some extent. This speculation gives furthermore rise to the conjecture that ALC learning methods with a strong (and of course approximately correct) bias can exploit the information provided by ambiguous examples better than methods with a weak bias. This conjecture has been supported empirically by experiments that have been carried out for three concrete learning techniques, namely ALC extensions of nearest neighbor classification, decision tree learning, and rule induction. The experiments also showed that applying our ALC methods to the complete data will usually yield better results than learning with a standard method from the subset of exactly labeled examples, at least if the expected benefit of the ambiguous examples is not too low.

data	method	q	$p = .1$	$p = .5$	$p = .9$
derma	ALC (r)	.3	.959 (.013)	.955 (.014)	.943 (.017)
	ALC (b)	.3	.959 (.013)	.955 (.015)	.940 (.018)
	standard	.3	.958 (.014)	.948 (.018)	.910 (.039)
	ALC (r)	.5	.958 (.014)	.949 (.015)	.890 (.028)
	ALB (b)	.5	.958 (.014)	.946 (.016)	.874 (.031)
	standard	.5	.957 (.014)	.945 (.019)	.851 (.067)
	ALC (r)	.7	.959 (.014)	.938 (.019)	.746 (.050)
	ALC (b)	.7	.958 (.014)	.936 (.018)	.745 (.046)
	standard	.7	.958 (.014)	.945 (.020)*	.833 (.072)*
ecoli	ALC (r)	.3	.845 (.025)	.832 (.025)	.798 (.024)
	ALC (b)	.3	.846 (.025)	.830 (.028)	.798 (.029)
	standard	.3	.845 (.026)	.827 (.028)	.743 (.059)
	ALC (r)	.5	.844 (.027)	.815 (.023)	.715 (.039)
	ALC (b)	.5	.843 (.022)	.814 (.028)	.709 (.045)
	standard	.5	.843 (.027)	.815 (.030)	.691 (.099)
	ALC (r)	.7	.844 (.022)	.801 (.029)	.582 (.050)
	ALC (b)	.7	.841 (.024)	.802 (.032)	.593 (.052)
	standard	.7	.842 (.024)	.820 (.034)*	.699 (.087)*
glass	ALC (r)	.3	.634 (.041)	.620 (.043)	.592 (.045)
	ALC (b)	.3	.638 (.040)	.622 (.041)	.592 (.044)
	standard	.3	.630 (.041)	.604 (.048)	.510 (.070)
	ALC (r)	.5	.636 (.042)	.611 (.042)	.542 (.052)
	ALC (b)	.5	.635 (.042)	.607 (.043)	.529 (.051)
	standard	.5	.633 (.042)	.599 (.045)	.438 (.077)
	ALC (r)	.7	.633 (.042)	.602 (.045)	.463 (.061)
	ALC (b)	.7	.631 (.042)	.604 (.045)	.453 (.060)
	standard	.7	.631 (.041)	.595 (.051)	.408 (.077)
housing	ALC (r)	.3	.488 (.027)	.461 (.029)	.423 (.017)
	ALC (b)	.3	.476 (.026)	.457 (.029)	.412 (.032)
	standard	.3	.486 (.028)	.455 (.030)	.403 (.032)
	ALC (r)	.5	.476 (.028)	.431 (.030)	.320 (.034)
	ALC (b)	.5	.477 (.027)	.445 (.031)	.367 (.032)
	standard	.5	.474 (.028)	.444 (.030)	.362 (.046)*
	ALC (r)	.7	.486 (.027)	.440 (.033)	.271 (.035)
	ALC (b)	.7	.478 (.029)	.443 (.030)	.324 (.032)
	standard	.7	.484 (.027)	.454 (.031)*	.369 (.048)*
zoo	ALC (r)	.3	.926 (.038)	.912 (.041)	.887 (.054)
	ALC (b)	.3	.925 (.038)	.911 (.042)	.886 (.055)
	standard	.3	.925 (.039)	.896 (.053)	.782 (.104)
	ALC (r)	.5	.924 (.037)	.901 (.048)	.824 (.072)
	ALC (b)	.5	.925 (.039)	.895 (.048)	.777 (.091)
	standard	.5	.923 (.038)	.889 (.059)	.667 (.155)
	ALC (r)	.7	.922 (.038)	.885 (.058)	.673 (.110)
	ALC (b)	.7	.924 (.039)	.881 (.060)	.609 (.111)
	standard	.7	.921 (.038)	.884 (.061)	.655 (.162)

Table 6: Results for 5-NN classification (classification rate and standard deviation).

data	method	q	$p = .1$	$p = .5$	$p = .9$
derma	ALC (r)	.3	.860 (.046)	.841 (.051)	.809 (.069)
	ALC (b)	.3	.861 (.047)	.839 (.055)	.802 (.069)
	standard	.3	.858 (.049)	.814 (.063)	.654 (.129)
	ALC (r)	.5	.858 (.047)	.818 (.057)	.742 (.098)
	ALB (b)	.5	.854 (.047)	.816 (.058)	.736 (.082)
	standard	.5	.858 (.049)	.807 (.069)	.488 (.155)
	ALC (r)	.7	.855 (.048)	.802 (.059)	.618 (.125)
	ALC (b)	.7	.854 (.048)	.801 (.063)	.659 (.092)
	standard	.7	.855 (.048)	.799 (.075)	.446 (.154)
ecoli	ALC (r)	.3	.705 (.039)	.676 (.041)	.645 (.043)
	ALC (b)	.3	.707 (.038)	.682 (.038)	.663 (.039)
	standard	.3	.704 (.043)	.655 (.058)	.543 (.115)
	ALC (r)	.5	.703 (.039)	.658 (.042)	.611 (.051)
	ALC (b)	.5	.703 (.038)	.669 (.039)	.639 (.045)
	standard	.5	.700 (.041)	.646 (.059)	.517 (.139)
	ALC (r)	.7	.700 (.039)	.648 (.043)	.567 (.065)
	ALC (b)	.7	.701 (.039)	.661 (.040)	.635 (.051)
	standard	.7	.699 (.041)	.643 (.064)	.509 (.149)
housing	ALC (r)	.3	.348 (.038)	.321 (.043)	.282 (.045)
	ALC (b)	.3	.348 (.038)	.333 (.042)	.311 (.044)
	standard	.3	.353 (.039)*	.334 (.051)*	.313 (.088)*
	ALC (r)	.5	.346 (.038)	.308 (.043)	.246 (.047)
	ALC (b)	.5	.348 (.038)	.331 (.044)	.294 (.043)
	standard	.5	.353 (.039)*	.336 (.051)*	.306 (.099)*
	ALC (r)	.7	.348 (.038)	.301 (.046)	.261 (.062)
	ALC (b)	.7	.352 (.036)	.321 (.044)	.286 (.078)
	standard	.7	.350 (.042)*	.337 (.052)*	.302 (.104)*
glass	ALC (r)	.3	.557 (.059)	.533 (.065)	.507 (.072)
	ALC (b)	.3	.559 (.059)	.534 (.069)	.496 (.080)
	standard	.3	.553 (.063)	.514 (.086)	.437 (.120)
	ALC (r)	.5	.556 (.055)	.525 (.066)	.460 (.085)
	ALC (b)	.5	.555 (.054)	.513 (.078)	.434 (.082)
	standard	.5	.551 (.064)	.497 (.091)	.395 (.152)
	ALC (r)	.7	.554 (.056)	.507 (.075)	.410 (.092)
	ALC (b)	.7	.557 (.057)	.504 (.079)	.382 (.065)
	standard	.7	.554 (.064)	.493 (.093)	.389 (.172)
zoo	ALC (r)	.3	.876 (.057)	.841 (.063)	.806 (.066)
	ALC (b)	.3	.876 (.058)	.843 (.060)	.807 (.063)
	standard	.3	.876 (.059)	.814 (.084)	.654 (.171)
	ALC (r)	.5	.877 (.057)	.827 (.067)	.765 (.079)
	ALC (b)	.5	.876 (.057)	.830 (.063)	.753 (.103)
	standard	.5	.873 (.060)	.811 (.091)	.552 (.245)
	ALC (r)	.7	.873 (.055)	.820 (.069)	.696 (.102)
	ALC (b)	.7	.874 (.054)	.825 (.066)	.553 (.206)
	standard	.7	.873 (.061)	.807 (.092)	.500 (.272)

Table 7: Results for decision tree induction.

data set	method	q	$p = .1$	$p = .5$	$p = .9$
Dermatology	ALC (r)	.3	.836 (.042)	.812 (.047)	.770 (.058)
	ALC (b)	.3	.836 (.042)	.815 (.048)	.775 (.056)
	standard	.3	.832 (.043)	.765 (.064)	.618 (.093)
	ALC (r)	.5	.832 (.042)	.784 (.057)	.685 (.071)
	ALB (b)	.5	.834 (.043)	.786 (.055)	.702 (.064)
	standard	.5	.832 (.044)	.747 (.065)	.513 (.128)
	ALC (r)	.7	.825 (.044)	.748 (.063)	.633 (.084)
	ALC (b)	.7	.829 (.041)	.763 (.059)	.645 (.070)
	standard	.7	.825 (.046)	.736 (.068)	.476 (.143)
Ecoli	ALC (r)	.3	.747 (.033)	.741 (.037)	.730 (.047)
	ALC (b)	.3	.746 (.033)	.739 (.043)	.715 (.054)
	standard	.3	.745 (.034)	.723 (.056)	.630 (.113)
	ALC (r)	.5	.745 (.034)	.735 (.044)	.701 (.064)
	ALC (b)	.5	.744 (.034)	.725 (.052)	.636 (.077)
	standard	.5	.745 (.035)	.723 (.061)	.577 (.138)
	ALC (r)	.7	.746 (.033)	.726 (.053)	.638 (.072)
	ALC (b)	.7	.748 (.032)	.709 (.060)	.504 (.140)
	standard	.7	.746 (.033)	.718 (.064)	.577 (.144)
Glass	ALC (r)	.3	.655 (.048)	.632 (.049)	.615 (.055)
	ALC (b)	.3	.657 (.049)	.642 (.052)	.612 (.060)
	standard	.3	.655 (.050)	.617 (.067)	.528 (.116)
	ALC (r)	.5	.659 (.047)	.621 (.054)	.565 (.062)
	ALC (b)	.5	.659 (.048)	.617 (.059)	.465 (.109)
	standard	.5	.655 (.049)	.607 (.075)	.431 (.157)
	ALC (r)	.7	.656 (.049)	.606 (.055)	.497 (.084)
	ALC (b)	.7	.657 (.048)	.601 (.066)	.220 (.101)
	standard	.7	.658 (.050)*	.605 (.076)	.408 (.176)
Housing	ALC (r)	.3	.446 (.032)	.424 (.033)	.397 (.039)
	ALC (b)	.3	.446 (.032)	.423 (.036)	.389 (.042)
	standard	.3	.447 (.034)*	.411 (.046)	.328 (.091)
	ALC (r)	.5	.445 (.031)	.411 (.034)	.349 (.053)
	ALC (b)	.5	.445 (.030)	.402 (.039)	.281 (.071)
	standard	.5	.443 (.034)	.407 (.049)	.302 (.102)
	ALC (r)	.7	.446 (.032)	.395 (.041)	.300 (.060)
	ALC (b)	.7	.444 (.031)	.384 (.045)	.125 (.049)
	standard	.7	.448 (.034)*	.409 (.049)	.313 (.105)*
Zoo	ALC (r)	.3	.898 (.053)	.862 (.068)	.815 (.079)
	ALC (b)	.3	.897 (.054)	.862 (.070)	.804 (.091)
	standard	.3	.896 (.055)	.847 (.087)	.723 (.168)
	ALC (r)	.5	.897 (.054)	.838 (.070)	.720 (.098)
	ALC (b)	.5	.899 (.052)	.837 (.081)	.618 (.153)
	standard	.5	.901 (.053)*	.843 (.085)*	.612 (.247)
	ALC (r)	.7	.889 (.052)	.812 (.088)	.598 (.125)
	ALC (b)	.7	.891 (.055)	.807 (.092)	.300 (.191)
	standard	.7	.893 (.054)*	.835 (.089)*	.564 (.272)

Table 8: Results for rule induction.

In summary, our idea to "disambiguate via biased induction" can be seen as a simple yet effective alternative to the probabilistic approaches proposed in [16, 19]. Indeed, let us again emphasize that these two approaches, just like machine learning and statistical methods in general, are not competing but *complementary*, in the sense that they apply to different types of learning methods: The expectation-maximization approach assumes a parameterized (statistical) model and is not immediately applicable to machine learning methods as those considered in this paper. For example, it is not applicable to k-NN classification, as there are no parameters to estimate.[10] Likewise, we are not aware of suitable statistical approaches to rule induction. In the case of decision trees, a statistical model amenable to ML estimation has indeed been proposed in [21]. But even if a classifier can be trained in both ways, heuristically and via maximum likelihood, the former will definitely be much more efficient than the latter. For example, the decision tree learned in [22] for 12 binary input variables already involved more than 1,000 parameters, and convergence was not achieved before approximately 100 training epochs.

Despite the fact that our approach has its own right to exist, independently of the statistical alternative, exploring the relation between the two approaches is of course interesting and worth further investigation. Another interesting topic of future work concerns a detailed elaboration of the alternative applications of ALC as outlined in section 6.

References

[1] KP. Bennet and A. Demiriz. Semi-supervised support vector machines. In MS. Kearns, SA. Solla, and DA. Cohn, editors, *Advances in Neural Information Processing 11*, pages 368–374. MIT Press, 1999.

[2] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

[10] This is true at least for the simple k-NN classifier. When approaching k-NN by estimating class conditional density functions, parameters may of course come into play. But even then, it is not obvious how to apply ML estimation, since estimations are derived locally, and no global model is constructed.

[3] K. Cao-Van and B. De Baets. Growing decision trees in an ordinal setting. *International Journal of Intelligent Systems*. To appear.

[4] P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proc. of the 5th European Working Session of Learning*, pages 151–163, Porto, Portugal, 1991.

[5] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

[6] W.W. Cohen. Fast effective rule induction. In *Proc. 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, 1995. Morgan Kaufmann.

[7] B.V. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1991.

[8] T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence Journal*, 89, 1997.

[9] P. Domingos. The role of Occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3:409–425, 1999.

[10] D. Dubois and H. Prade. *Possibility Theory*. Plenum Press, 1988.

[11] J. Cavarelli et al. The structure of Staphylococcus aureus epidermolytic toxin A, an atypic serine protease, at 1.7 A resolution. *Structure*.

[12] U. Fayyad and KB. Irani. Multi-interval discretization of continuos attributes as preprocessing for classification learning. In *Proceedings of the 13th international Joint Conference on Artificial Intelligence*, pages 1022–1027. Morgan Kaufmann, 1993.

[13] E. Frank and M. Hall. A simple approach to ordinal classification. In *Proc.* ECML–2001*, 12th European Conference on Machine Learning*, pages 145–156, Freiburg, Germany, 2001.

[14] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999.

[15] I. Gilboa and D. Schmeidler. Case-based decision: An extended abstract. In H. Prade, editor, *Proceedings* ECAI-98*, 13th European Conference on Artificial Intelligence*, pages 706–710, Brighton, UK, 1998.

[16] Y. Grandvalet. Logistic regression for partial labels. In *IPMU–02, Int. Conf. Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 1935–1941, Annecy, France, 2002.

[17] R. Herbrich, T. Graepel, and K. Obermayer. Regression models for ordinal data: A machine learning approach. Technical Report TR 99-3, Department of Computer Science, Technical University of Berlin, Berlin, Germany, 1999.

[18] E. Hüllermeier. Experience-based decision making. IEEE *Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 35(5):641–653, 2005.

[19] R. Jin and Z. Ghahramani. Learning with multiple labels. In *16th Annual Conference on Neural Information Processing Systems*, Vancouver, Canada, 2002.

[20] GH. John. Robust decision trees: Removing outliers from databases. In UM. Fayyad and R. Uthurusamy, editors, *Int. Conf. on Knowledge Discovery and Data Mining*, pages 174–179, Menlo Park, CA, 1995. AAAI Press.

[21] M. Jordan. A statistical approach to decision tree modeling. In *Proc. ICML–94, International Conference on Machine Learning*, pages 363–370, 1994.

[22] MI. Jordan and RA. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(181).

[23] G.J. Klir and M.J. Wierman. *Uncertainty-Based Information.* Physica-Verlag, Heidelberg, 1998.

[24] S. Kramer, G. Widmer, B. Pfahringer, and M. De Groeve. Prediction of ordinal classes using regression trees. *Fundamenat Informaticae*, 34:1–15, 2000.

[25] A. McCallum. Multi-label text classification with a mixture model trained by EM. In *AAAI–99 Workshop on Text Learning*, 1999.

[26] RS. Michalski. On the quasi-minimal solution of the covernig problem. In *Proc. 5th International Symposium on Information Processing, Vol. A3*, pages 125–128, Bled, Yugoslavia, 1969.

[27] RS. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The multi-purpose incremental learning system AQ15 and its testing application on three medical domains. In *Proc. 5th National Conference on Artificial Intelligence*, pages 1041–1045, Philadelphia, PA, 1986.

[28] R. Potharst and J.C. Bioch. Decision trees for ordinal classification. *Intelligent Data Analysis*, 4:97–112, 2000.

[29] JR. Quinlan. Learning logical definitions from relations. *Machine Learning*, 5:239–266, 1990.

[30] J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

[31] H.A. Simon. *Models of Man*. John Wiley and Sons, New York, 1957.

[32] D. Wettschereck, D.W. Aha, and T. Mohri. A review and empirical comparison of feature weighting methods for a class of lazy learning algorithms. *AI Review*, 11:273–314, 1997.

[33] D.L. Wilson. Asymptotic properties of nearest neighbor rules using edited data. IEEE *Transactions on Systems, Man, and Cybernetics*, SMC-2(3):408–421, 1972.

[34] D.R. Wilson. *Advances in Instance-Based Learning Algorithms*. PhD thesis, Department of Computer Science, Brigham Young University, 1997.

[35] R.R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE *Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, 1988.