Learning from Ambiguously Labeled Examples

Eyke Hüllermeier and Jürgen Beringer

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg, Germany
eyke.huellermeier@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 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.

1 Introduction

One of the standard problems in (supervised) machine learning is inducing a *classification function* from a set of training data. The latter usually 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 [14]. 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 [7]. A bag, which is a collection of several instances, is labeled positive iff if contains at least one positive example. 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 [11, 13], 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 [11, 13], 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 finally presented in Section 5.

2 Ambiguous Label Classification

Let $\mathcal{X}$ denote an instance space, where an instance corresponds to the attribute–value description x of an object: $\mathcal{X} = X_1 \times X_2 \times \ldots \times X_\ell$, with $X_\imath$ the domain of the $\imath$-th attribute. Thus, 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 [2].

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

Ambiguous data may comprise important information. In fact, the benefit of this information might be especially high if it is considered, not as an isolated piece of knowledge, 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 $\lambda_{x_\imath}$ of an instance $x_\imath$ 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 $\lambda_{x_\imath} = \lambda_2$ we might find a very simple classification tree for the complete data, while $\lambda_{x_\imath} = \lambda_1$ requires a comparatively complex model (see Fig.1). Relying on the simplicity heuristic underlying most machine learning methods [8], this finding clearly suggests that $\lambda_{x_\imath} = \lambda_2$. Thus, looking at the original information $\lambda_{x_\imath} \in \{\lambda_1, \lambda_2\}$ with a view that is "biased" by the model assumptions, 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 suggests that ambiguous label information might

indeed be useful and, in particular, 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.

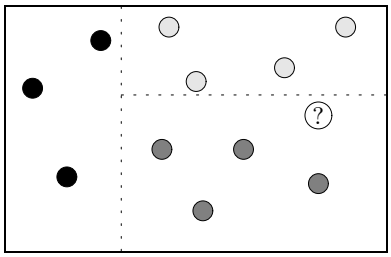

Fig. 1. Classification problem with three labels: black (λ_1), grey (λ_2), light (λ_3). The instance with a question mark is either black or grey. Assigning label grey allows one to fit a very simple decision tree (as represented by the axis-parallel decision boundaries). Note that this hypothetical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

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

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

This approach gives rise to an EM (expectation-maximization) approach in which model adaptation and modification of label information are performed alternately: Starting with a uniform distribution over each label set L_{x_i}, an optimal parameter θ^* is determined. Using this parameter resp. the associated model M_{θ^*}, the probabilities of the labels $\lambda \in L_{x_i}$ are then re-estimated. This process of estimating parameters and adjusting probabilities is iterated until convergence is eventually achieved [11, 13].

On the one hand, this approach is rather elegant and first empirical evidence has been gathered for its practical effectiveness [11, 13]. 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.

Unfortunately, standard classification methods 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 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 [6], the label $\lambda_{x_0}^{est}$ hypothetically 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:

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

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 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 its associated labels. If the maximum in (3) is not unique, one among the labels with highest score is simply chosen at random.[1]

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

4.2 Decision Tree Induction

Another standard learning method whose extension to the ALC setting might be of interest, is decision tree induction [15]. Its basic strategy of partitioning the data in a recursive manner can of course be maintained for ALC. The main modification rather concerns the splitting measure. In fact, standard measures of the (im)purity of a set of examples, such as entropy, cannot be used, since these measures are well-defined only for a probability distribution over the label set.

As an extended measure of (im)purity, we propose the *potential entropy* of a set of examples $\mathcal{D}$, defined by

$$E^*(\mathcal{D}) \stackrel{\text{df}}{=} \min_{\mathcal{S} \in \mathcal{F}(\mathcal{D})} E(\mathcal{S}), \tag{4}$$

where $\mathcal{F}(\mathcal{D})$ is the set of selections (1) and $E(\mathcal{S})$ denotes the standard entropy: $E(\mathcal{S}) \stackrel{\text{df}}{=} -\sum_{\imath=1}^{m} p_\imath \log_2(p_\imath)$, with $p_\imath$ the proportion of elements in $\mathcal{S}$ labeled by $\lambda_\imath$. As can be seen, (4) 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 hopefully constructed in an optimal manner.

Of course, computing the potential entropy comes down to solving a combinatorial optimization problem and becomes intractable for large samples. Therefore, we suggest the following heuristic approximation of (4):

$$E^+(\mathcal{D}) \stackrel{\text{df}}{=} E(\mathcal{S}^*), \tag{5}$$

where the selection $\mathcal{S}^*$ is defined as follows: Let $q_\imath$ be 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 $\lambda_\imath \in L_{x_\jmath}$. 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. We found that the measure (5) yields very good results in practice and compares favorably with alternative extensions of splitting measures [12].

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

Pruning a fully grown tree can principally be done in the same way as pruning standard trees. We implemented the pruning technique that is used in C4.5 [15].

4.3 Rule Induction

An alternative to the *divide-and-conquer* strategy followed by decision tree learners is to induce rules in a more direct way, using a *separate-and-conquer* or *covering* strategy [10]. Concrete implementations of this approach include algorithms such as, e.g., CN2 [4, 3] and Ripper [5].

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. 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-all 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, 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 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:

$$L(r) \stackrel{\text{df}}{=} (p+1)(n+p+2)^{-1}, \tag{6}$$

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: Similarly to the generalization of the entropy measure, we have turned the Laplace-estimate into a "potential" Laplace-estimate: Considering label λ_j as the positive class, $p = p_j$ is given by the number of all examples x_i covered by the rule r and such that $\lambda_j \in L_{x_i}$. This way, (6) can be derived for each label, and the maximal value is adopted as an evaluation of the rule:

$$L(r) \;=\; \max_{1 \leq j \leq m} (p_j + 1)(|r| + 2)^{-1},$$

where $|r|$ is the number of examples covered by the rule. The consequent of r is then given by the label λ_j 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. Rather, the most suitable class is chosen 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 to disambiguate 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 exploiting ambiguous data for model induction by using a suitable ALC-method is usually better than the obvious alternative, namely to ignore such data and learn with a standard algorithm from the remaining (exactly labeled) examples. We 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 other 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 assumes that labels are added 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, $\lambda^{(k)}$, 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 because, roughly speaking, disambiguating the data might become more difficult in the case of a "systematic" contamination.

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 experiments.

For decision tree learning and rule induction, all numeric attributes have been discretized in advance using hierarchical entropy-based discretization [9]. 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.

5.2 Results

Due to reasons of space, results are presented for only five data sets from the UCI repository: (1) dermatology (385 instances, 34 attributes, 6 classes), (2) ecoli (336, 7, 8), (3) housing (506,13,10), (4) glass (214, 9, 6), (5) zoo (101, 16, 7) and a few combinations of (p, q)-parameters (more results can be found in an extended version, available as a technical report from the authors).

The results for k-NN classification with $k = 5$ are summarized in Table 1, 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. Roughly speaking, the superiority of the ALC version shows that relying on a nearby ambiguous neighbor is usually better than looking at an exact example that is faraway (because the close, ambiguous ones have been removed). 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

Table 1. Results for 5-NN classification (classification rate and standard deviation).

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 2. Results for decision tree induction.

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)

the random model is much better than for the Bayes model in the case of highly contaminated data.

For decision tree induction, the ALC-version consistently outperforms the standard version. As the results in Table 2 show, the gain in performance is even higher than for NN classification. Again, the results show that a systematic contamination of the data, using the Bayes instead of the random model, does hardly affect the performance of ALC. 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 (not only for decision tree learning but also for NN classification). 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. A 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. (Consequently, ordinal classification methods should be applied, and the results for this data set should not be overrated in our context.) Moreover, the Bayesian model 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.

Since the experimental results for rule induction are rather similar to those for decision tree learning, they are omitted here for reasons of space.

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 than the baseline method (which ignores ambiguous examples and applies standard learning methods). A closer examination reveals two interesting points: Firstly, it seems that the gain in classification accuracy (of ALC compared with the baseline method) 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. Intuitively, these findings can be explained as follows: 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 simply ignore such examples, at least if enough other data is 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 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, we argued that looking at the label information with a "biased view" may remove the ambiguity of that information to some extent. This idea gives 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. In any case, our approach can be seen as a simple yet effective alternative that complements the probabilistic approaches proposed in [11, 13] in a reasonable way.

References

1. KP. Bennet and A. Demiriz. Semi-supervised support vector machines. In *Advances in Neural Information Processing 11*, pages 368–374. MIT Press, 1999.
2. J. Cavarelli et al. The structure of Staphylococcus aureus epidermolytic toxin A, an atypic serine protease, at 1.7 A resolution. *Structure* 5(6):813–24, 1997.
3. P Clark and R Boswell. Rule induction with CN2: Some recent improvements. In *Proc. 5th Europ. Working Session of Learning*, pages 151–163, Porto, 1991.
4. P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.
5. W.W. Cohen. Fast effective rule induction. *Proc. 12th ICML*, pages 115–123, Tahoe City, CA, 1995.
6. B.V. Dasarathy, editor. *Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques*. IEEE Computer Society Press, Los Alamitos, California, 1991.
7. T.G. Dietterich, R.H. Lathrop, and T. Lozano-Perez. Solving the multiple-instance problem with axis-parallel rectangles. *Art. Intell. Journal*, 89, 1997.
8. P. Domingos. The role of Occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, 3:409–425, 1999.
9. U. Fayyad and KB. Irani. Multi-interval discretization of continuos attributes as preprocessing for classification learning. Proc. *IJCAI–93*, pages 1022–1027, 1993.
10. J Fürnkranz. Separate-and-conquer rule learning. *AI Review*, 13(1):3–54, 1999.
11. Y Grandvalet. Logistic regression for partial labels. *IPMU–02*, pages 1935–1941, Annecy, France, 2002.
12. E. Hüllermeier and J. Beringer. Learning decision rules from positive and negative preferences. *IPMU-04*, Perugia, Italy, 2004.
13. R Jin and Z Ghahramani. Learning with multiple labels. *NIPS-02*, Vancouver, Canada, 2002.
14. A. McCallum. Multi-label text classification with a mixture model trained by EM. In *AAAI–99 Workshop on Text Learning*, 1999.
15. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.