Available online at www.sciencedirect.com
SCIENCE DIRECT®
Theoretical Computer Science 348 (2005) 70–83
Theoretical Computer Science
www.elsevier.com/locate/tcs

Learning from positive and unlabeled examples☆

François Denis[a,*], Rémi Gilleron[b], Fabien Letouzey[b]

[a] *Équipe BDAA, LIF, UMR 6166 CNRS, Université de Provence, Marseille, France*
[b] *Équipe Grappa, LIFL, UPRESA 8022 CNRS, Université de Lille 1 and Université Charles de Gaulle, Lille 3, France*

Abstract

In many machine learning settings, labeled examples are difficult to collect while unlabeled data are abundant. Also, for some binary classification problems, positive examples which are elements of the target concept are available. Can these additional data be used to improve accuracy of supervised learning algorithms? We investigate in this paper the design of learning algorithms from positive and unlabeled data only. Many machine learning and data mining algorithms, such as decision tree induction algorithms and naive Bayes algorithms, use examples only to evaluate statistical queries (SQ-like algorithms). Kearns designed the statistical query learning model in order to describe these algorithms. Here, we design an algorithm scheme which transforms any SQ-like algorithm into an algorithm based on positive statistical queries (estimate for probabilities over the set of positive instances) and instance statistical queries (estimate for probabilities over the instance space). We prove that any class learnable in the statistical query learning model is learnable from positive statistical queries and instance statistical queries only if a lower bound on the weight of any target concept f can be estimated in polynomial time. Then, we design a decision tree induction algorithm POSC4.5, based on C4.5, that uses only positive and unlabeled examples and we give experimental results for this algorithm. In the case of imbalanced classes in the sense that one of the two classes (say the positive class) is heavily underrepresented compared to the other class, the learning problem remains open. This problem is challenging because it is encountered in many real-world applications.

Keywords: PAC learning; Statistical query model; Semi-supervised learning; Data mining

1. Introduction

The field of data mining (sometimes referred to knowledge discovery in databases) addresses the question of how best to use various sets of data to discover regularities and to improve decisions. The learning step is central in the data mining process. A first generation of supervised machine learning algorithms (e.g. decision tree induction algorithms, neural network learning methods, Bayesian learning methods, logistic regression, . . .) have been demonstrated to be of significant value in a data mining perspective and they are now widely used and available in commercial products. But these machine learning methods are issued from nonparametric statistics and suppose that the input sample is a quite large set of independently and identically distributed (i.i.d.) labeled data described by numeric or symbolic features. But, in a data mining or a text mining perspective, one has to use historical data that have been collected from various

☆ This research was partially supported by: "CPER 2000-2006, Contrat de Planétat, région Nord/Pas-de-Calais: axe TACT, projet TIC"; fonds européens FEDER "TIC, Fouille Intelligente de données, Traitement Intelligent des Connaissances" OBJ 2-phasing out, 2001/3 - 4.1 - n 3".
* Corresponding author. Tel.: +491113605; fax: +491113602.
E-mail addresses: fdenis@cmi.univ-mrs.fr (F. Denis), gilleron@lifl.fr (R. Gilleron), letouzey@lifl.fr (F. Letouzey).

0304-3975/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2005.09.007

origins and moreover, i.i.d. labeled data may be expensive to collect or even unavailable. On the other hand, unlabeled data providing information about the underlying distribution or examples of one particular class (that we shall call the *positive* class) may be easily available. Can this additional information help to learn? Here, we address the issue of designing classification algorithms that are able to utilize data from diverse data sources: labeled data (if available), unlabeled data, and positive data.

Along this line of research, there has recently been significant interest in semi-supervised learning, that is the design of learning algorithms from both labeled and unlabeled data. In the semi-supervised setting, one of the questions is: can unlabeled data be used to improve accuracy of supervised learning algorithms? Intuitively, the answer is positive because unlabeled data must provide some information about the hidden distribution. Nevertheless, it seems that the question is challenging from a theoretical perspective as well as a practical one. A promising line of research is the co-training setting first defined in [3]. Supposing that the features are naturally divided into two disjoint sets, the co-training algorithm builds two classifiers, and each one of these two is used to label unlabeled data for the other. In [3], theoretical results are proved, learning situations for which the assumption is true are described in [14], experimental results may be found in [3,15]. See also [8] for another approach of the co-training setting. Other approaches include using the EM algorithm [16], and using transductive inference [11]. A NIPS'99 workshop and a NIPS'00 competition were also organized on using unlabeled data for supervised learning.

In this paper, we consider binary classification problems. One of the two classes is called the positive class. We are interested in the following questions:

- How can unlabeled data and positive data be used to improve the accuracy of supervised learning algorithms?
- How can learning algorithms from unlabeled data and positive data be designed from previously known supervised learning algorithms?

First, let us justify that the problem is relevant for applications. We argue that, in many practical situations, elements of the target concept may be abundant and cheap to collect. For instance, consider one diagnosis of diseases: in order to obtain an i.i.d. sample of labeled examples, it is necessary to systematically detect the disease on a representative sample of patients and this task may be quite expensive (or impossible). On the other hand, it may be easy to collect the medical files of patients who have the disease. Also, unlabeled data are any pool of patients possibly having the disease.

Second, let us note that many machine learning algorithms as decision tree learning algorithms and Bayesian learning algorithms only use examples to estimate statistics. In other words, many machine learning algorithms may be considered as statistical query (SQ) learning algorithms. Thus, we are interested in general schemes which transform supervised SQ-like learning algorithms into learning algorithms from both unlabeled data and positive data.

In a preliminary paper [6], we have given evidence—with both theoretical and empirical arguments—that positive data and unlabeled data can boost accuracy of SQ-like learning algorithms. It was noted that learning with positive and unlabeled data is possible as soon as the weight of the target concept (i.e. the ratio of positive examples) is known by the learner. An estimate of the weight can be obtained either by an extra-oracle (say for a similar problem) or from a small set of labeled examples. In the present paper, we consider the more general problem where only positive data and unlabeled data are available. We present a general scheme which transforms any SQ-like supervised learning algorithm L into an algorithm PL using only positive data and unlabeled data. We prove that PL is a learning algorithm as soon as the learner is given access to a lower bound on the weight of the target concept. It remains open whether it is possible to design an algorithm from positive data and unlabeled data from any SQ learning algorithm in the general case.

The theoretical framework is presented in Section 2. Our learning algorithm is defined and proved in Section 3, some consequences about the equivalence of models are also given. It is applied to tree induction and experimental results are given in Section 4.

2. Learning models

2.1. Learning models from labeled data

For each $n \geqslant 1$, X_n denotes an instance space on n attributes. A concept f is a subset of some instance space X_n or equivalently a $\{0, 1\}$-valued function defined on X_n. For each $n \geqslant 1$, let $\mathcal{C}_n \subset 2^{X_n}$ be a set of concepts. Then $\mathcal{C} = \bigcup_{n \geqslant 1} \mathcal{C}_n$ denotes a concept class over $X = \bigcup_{n \geqslant 1} X_n$. The *size* of a concept f is the size of a smallest representation

of f for a given representation scheme. An *example* of a concept f is a pair $\langle x, f(x)\rangle$, which is *positive* if $f(x) = 1$ and *negative* otherwise. Let D be a distribution over the instance space X_n, for a subset A of X_n, we denote by $D(A)$ the probability of the event $[x \in A]$. For a subset A of X_n such that $D(A) \neq 0$, we denote by D_A the induced distribution over A. For instance, for a concept f over X_n such that $D(f) \neq 0$ and for any $x \in X_n$, $D_f(x) = D(x)/D(f)$ when $f(x) = 1$ and $D_f(x) = 0$ otherwise. Let f and g be concepts over the instance space X_n, we denote by $\overline{f}$ the complement of the set f in X_n and by $f\Delta g$ the set $f\Delta g = \{x \in X_n \mid f(x) \neq g(x)\}$.

Let f be a target concept over X in some concept class $\mathcal{C}$. Let D be the hidden distribution defined over X. In the PAC model [18], the learner is given access to an example oracle $EX(f, D)$ which returns an example $\langle x, f(x)\rangle$ drawn randomly according to D at each call. A concept class $\mathcal{C}$ is *PAC learnable* if there exist a learning algorithm L and a polynomial $p(.,.,.,.)$ with the following property: for any n and any $f \in \mathcal{C}_n$, for any distribution D on X_n, and for any $0 < \varepsilon < 1$ and $0 < \delta < 1$, if L is given access to $EX(f, D)$ and to inputs ε and δ, then with probability at least $1 - \delta$, L outputs a hypothesis concept h satisfying $error(h) = D(f\Delta h) \leqslant \varepsilon$ in time bounded by $p(1/\varepsilon, 1/\delta, n, size(f))$. In this paper, we always suppose that the value of $size(f)$ is known by the learner. Recall that if $size(f)$ is not given then the halting criterion of the algorithm is probabilistic [9]. Also, for many concept classes the natural definition of $size(f)$ is already bounded by a polynomial in n.

One criticism of the PAC model is that it is a noise-free model. Therefore, extensions in which the label provided with each random example may be corrupted with random noise were studied. The *classification noise model* (CN model for short) was first defined by Angluin and Laird [1]. A variant of the CN model, namely the *constant-partition classification noise model* (CPCN model for short) has been defined by Decatur [5]. In this model, the labeled example space is partitioned into a constant number of regions, each of which may have a different noise rate. An interesting example is the case where the rate of false-positive examples differs from the rate of false-negative examples. We only define this restricted variant of the CPCN model. The noisy oracle $EX^{\eta_+,\eta_-}(f, D)$ is a procedure which, at each call, draws an element x of X_n according to D and returns (i) $(x, 1)$ with probability $1 - \eta_+$ and $(x, 0)$ with probability η_+ if $x \in f$, (ii) $(x, 0)$ with probability $1 - \eta_-$ and $(x, 1)$ with probability η_- if $x \in \overline{f}$. Let $\mathcal{C}$ be a concept class over X. We say that $\mathcal{C}$ is *CPCN learnable* if there exist a learning algorithm L and a polynomial $p(.,.,.,.,.)$ with the following property: for any n and any $f \in \mathcal{C}_n$, for any distribution D on X_n, and for any $0 \leqslant \eta_+, \eta_- < \frac{1}{2}$ and $0 < \varepsilon, \delta < 1$, if L is given access to $EX^{\eta_+,\eta_-}(f, D)$ and to inputs ε and δ, then with probability at least $1-\delta$, L outputs a hypothesis concept $h \in \mathcal{C}$ satisfying $D(f\Delta h) \leqslant \varepsilon$ in time bounded by $p(1/\varepsilon, 1/\delta, 1/\gamma, size(f), n)$ where $\gamma = \min\{1/2 - \eta_+, 1/2 - \eta_-\}$.

Many machine learning algorithms only use examples in order to estimate probabilities. This is the case for induction tree algorithms such as C4.5 [17] and CART [4]. This is also the case for highly practical Bayesian learning method as the naive Bayes classifier. Kearns defined the *statistical query model* (SQ model for short) in [12]. The SQ model is a specialization of the PAC model in which the learner forms its hypothesis solely on the basis of estimates of probabilities. A *statistical query* over X_n is a mapping $\chi : X_n \times \{0, 1\} \to \{0, 1\}$ associated with a tolerance parameter $0 < \tau \leqslant 1$. In the SQ model the learner is given access to a statistical oracle $STAT(f, D)$ which, at each query (χ, τ), returns an estimate of $D(\{x \mid \chi(\langle x, f(x)\rangle) = 1\})$ within accuracy τ. Let $\mathcal{C}$ be a concept class over X. We say that $\mathcal{C}$ is *SQ learnable* if there exist a learning algorithm L and polynomials $p(.,.,.), q(.,.,.)$ and $r(.,.,.)$ with the following property: for any $f \in \mathcal{C}$, for any distribution D over X, and for any $0 < \varepsilon < 1$, if L is given access to $STAT(f, D)$ and to input ε, then, for every query (χ, τ) made by L, the predicate χ can be evaluated in time $q(1/\varepsilon, n, size(f))$, and $1/\tau$ is bounded by $r(1/\varepsilon, n, size(f))$, L halts in time bounded by $p(1/\varepsilon, n, size(f))$ and L outputs a hypothesis $h \in \mathcal{C}$ satisfying $D(f\Delta h) \leqslant \varepsilon$.

We slightly modify the statistical oracle $STAT(f, D)$. Let f be the target concept and let us consider a statistical query χ made by a statistical query learning algorithm L. The statistical oracle $STAT(f, D)$ returns an estimate $\widehat{D}_\chi$ of $D_\chi = D(\{x \mid \chi(\langle x, f(x)\rangle) = 1\})$ within some given accuracy. We may write:

$$\begin{aligned} D_\chi &= D(\{x \mid \chi(\langle x, 1\rangle) = 1 \wedge f(x) = 1\}) + D(\{x \mid \chi(\langle x, 0\rangle) = 1 \wedge f(x) = 0\}) \\ &= D(\{x \mid \chi(\langle x, 1\rangle) = 1\} \cap f) + D(\{x \mid \chi(\langle x, 0\rangle) = 1\} \cap \overline{f}) \\ &= D(B \cap f) + D(C \cap \overline{f}), \end{aligned}$$

where the sets B and C are defined by

$$B = \{x \mid \chi(\langle x, 1\rangle) = 1\} \quad \text{and} \quad C = \{x \mid \chi(\langle x, 0\rangle) = 1\}.$$

Therefore, we consider a statistical oracle which, at each query (A, τ), returns estimates for probabilities $D(f \cap A)$ and $D(\overline{f} \cap A)$ within accuracy τ, where f is the target concept, $\overline{f}$ its complement and A any subset—for which membership is decidable in polynomial time—of the instance space. It should be clear for the reader that this technical modification does not change the SQ learnable classes.

It is clear that access to the example oracle $EX(f, D)$ being given, it is easy to simulate the statistical oracle $STAT(f, D)$ by drawing a sufficiently large set of labeled examples. Moreover, there is a general scheme which transforms any SQ learning algorithm into a PAC learning algorithm. It is also proved in [12] that the class of parity functions is learnable in the PAC model but cannot be learned from statistical queries.

It has been shown by Kearns that any class learnable from statistical query is also learnable in the presence of classification noise [12]. Following the results by Kearns, it has been proved by Decatur [5] that any class learnable from statistical queries is also learnable in the presence of CPCN. The proof uses the *hypothesis testing property*: a hypothesis with small error can be selected from a set of hypotheses by selecting the one with the fewest errors on a set of CPCN corrupted examples. If we confuse, in the notations, the name of the model and the set of learnable classes, we can write the following inclusions:

$$SQ \subseteq CPCN \subseteq CN \subseteq PAC, \tag{1}$$

$$SQ \subset PAC. \tag{2}$$

To our knowledge, the equivalences between the models CN and SQ or between the models CN and PAC remain open despite recent insights [2,10].

2.2. Learning models from positive and unlabeled data

The *learning model from positive examples* (POSEX for short) was first defined in [7]. The model differs from the PAC model in the following way: the learner gets information about the target function and the hidden distribution from two oracles, namely a *positive example oracle* $POS(f, D)$ and an *instance oracle* $INST(D)$ instead of an example oracle $EX(f, D)$. At each request by the learner, the instance oracle $INST(D)$ returns an element of the instance space X, i.e. an unlabeled example, according to the hidden distribution D. At each request by the learner, the positive example oracle $POS(f, D)$ returns a positive example according to the hidden distribution D_f. We have the following result:

Proposition 1 (*Denis [7]*). *Any class learnable in the CPCN model is learnable in the POSEX model.*

Proof. The proof is simple and as it may help to understand the proof of the main algorithm of the present paper, we sketch it below.

Let $\mathcal{C}$ be a CPCN learnable concept class, let L be a learning algorithm for $\mathcal{C}$ in the CPCN model, let f be the target concept, let D be a distribution over the instance space and let us suppose that $D(f) \neq 0$. We must show how L can be used to learn from the oracles $POS(f, D)$ and $INST(D)$.

Run L. At each call of the noisy oracle:

- with probability $\frac{2}{3}$, call $POS(f, D)$ and keep the positive label,
- with probability $\frac{1}{3}$, call $INST(D)$ and label the example as negative.

It can easily be shown that this is strictly equivalent calling the noisy oracle $EX^{\eta_+,\eta_-}(f, D')$ where:

$$D'(x) = \begin{cases} \dfrac{D(x)}{3} & \text{if } f(x) = 0, \\ \dfrac{D(x) + 2D_f(x)}{3} & \text{if } f(x) = 1, \end{cases}$$

$$\eta_+ = \frac{D(f)}{2 + D(f)},$$

$$\eta_- = 0.$$

Note that $\eta_+ \leqslant \frac{1}{3} < \frac{1}{2}$. And as for any subset A of the instance space, we have $D(A) \leqslant 3D'(A)$, it is sufficient to run the algorithm L with input accuracy $\varepsilon/3$ and input confidence δ to output with confidence greater than $1-\delta$ a hypothesis whose error rate is less than ε. □

The *learning model from positive queries* (POSQ for short) was also defined in [7]. In the POSQ model, there are a *positive statistical oracle PSTAT*(f, D) which provides estimates for probabilities $D_f(A)$ for any subset A of the instance space within a given tolerance and an *instance statistical oracle ISTAT*(D) which provides estimates for probabilities $D(A)$ for any subset A of the instance space within a given tolerance. The definition of a *POSQ learnable* class is similar to the definition of a SQ learnable class: the oracle *STAT*(f, D) is replaced by the two oracles *PSTAT*(f, D) and *ISTAT*(D). The POSQ model is weaker than the SQ model as there is no direct way to obtain an estimate of the weight $D(f)$ of the target concept. However, if we can get such an estimate, both models become equivalent. Indeed, statistical queries can be computed from instance queries and positive statistical queries as soon as the weight of the target concept is known because of the following equations:

$$\begin{aligned} \hat{D}(f \cap A) &= \hat{D}_f(A) \times \hat{D}(f), \\ \hat{D}(\overline{f} \cap A) &= \hat{D}(A) - \hat{D}(f \cap A). \end{aligned} \tag{3}$$

So, any class learnable in the SQ model is learnable in the POSQ model as soon as the learner is given access to the weight of the target concept or can compute it from the positive statistical oracle and the instance statistical oracle. This is formalized in the following result:

Proposition 2 (*Denis [7]*). *Let $\mathcal{C}$ be a concept class such that the weight of any target concept can be estimated in polynomial time within any given tolerance. If $\mathcal{C}$ is SQ learnable then $\mathcal{C}$ is POSQ learnable.*

We can summarize all the results with the following inclusions:

$$POSQ \subseteq SQ \subseteq CPCN \subseteq POSEX \subseteq PAC, \tag{4}$$

$$CPCN \subseteq CN \subseteq PAC, \tag{5}$$

$$SQ \subset POSEX. \tag{6}$$

The inequality between SQ and POSEX holds because the class of parity functions is POSEX learnable but not SQ learnable. Equivalences between POSQ and SQ and between POSEX and PAC remain open.

3. Learning algorithms from positive and unlabeled examples

We have already noticed that in practical data mining and text mining situations, statistical query-like algorithms, such as C4.5 or naive Bayes, are widely used. It is straightforward to see how a statistical query can be evaluated from labeled data. In a similar way, positive and instance statistical queries can easily be evaluated from positive and unlabeled data. So, in order to adapt classical learning algorithms to positive and unlabeled examples, we can show how SQ learning algorithms can be modified into POSQ learning algorithms.

In [6], we have studied the case where the weight of the target concept is either given by an oracle or evaluated from a small set of labeled examples. In this case, Eqs. (3) and Proposition 2 show how the transformation of the SQ algorithm can be achieved. We now consider the more general problem where no information on the weight of the target concept is given to the learner.

3.1. A generic learning algorithm from positive statistical queries and instance statistical queries

In this section, we provide a general scheme which transforms any SQ-like algorithm into a POSQ-like algorithm.

Let us consider a concept class $\mathcal{C}$ learnable in the SQ model by a learning algorithm L, and let γ be a positive real number. Let us recall that we suppose that *size*(f) is known by the learner. Also note that for most concept classes $\mathcal{C}$ learnable from statistical queries, the size of every target concept $f \in \mathcal{C}_n$ is bounded by a polynomial in n. We design

POSQ learning algorithm PL
parameters: SQ learning algorithm L, $\gamma \in (0,1)$; let p (respectively r) be the polynomial which bounds the running time of L (respectively the inverse of the tolerance needed for queries)
input: ϵ
Construction of a hypothesis set
Set ϵ' to $\frac{1}{2} \times \frac{\gamma}{2-\gamma} \times \epsilon$, τ_{min} to $\frac{1}{r(1/\epsilon, n, size(f))}$, N to $\lceil \frac{2}{\tau_{min}} \rceil$, α to $\frac{1}{2N}$
for $i = 1$ **to** N
the current estimate of $D(f)$ is $\hat{p}_i = (2i-1)\alpha$
run L with accuracy ϵ' using oracles $PSTAT(f, D)$, $ISTAT(D)$
within tolerance $\frac{\tau_{min}}{4}$ and use Equations 3 ;
if the running time exceeds $p(1/\epsilon', n, size(f))$
then let h_i be a default hypothesis
else let h_i be the output of L
Hypothesis testing algorithm
for $i = 1$ **to** N
call $PSTAT$ with input $\overline{h_i}$ within tolerance $\frac{\epsilon}{12}$
call $ISTAT$ with input h_i within tolerance $\frac{\epsilon}{12}$
set $\hat{e}(h_i)$ to $2\hat{D}_f(\overline{h_i}) + \hat{D}(h_i)$
output: $h = \operatorname{argmin}_{h_i} \hat{e}(h_i)$

Fig. 1. Learning algorithm from positive and unlabeled queries.

a POSQ learning algorithm *PL* based on the algorithm *L* which learns any target concept f in $\mathcal{C}$ such that $D(f) \geqslant \gamma$. A consequence of this result is that whenever a *lower bound* on the weight of the target concept is known a priori, every SQ learnable class is POSQ learnable. First, we give some comments on the algorithm *PL* which is described in Fig. 1 and second, we prove its correctness in Section 3.2.

The algorithm *PL* is based on a SQ learning algorithm *L* and is given access to a lower bound γ on the weight of the target concept. *PL* is composed of two stages: in the first stage, a set of hypotheses is constructed; in the second stage, a hypothesis is selected in the hypothesis set.

In the first stage, the algorithm *PL* iterates over larger guesses for $D(f)$. At each guess, the SQ learning algorithm is called. But only positive and instance queries are available, thus when *L* makes a SQ, Eqs. (3) are used with the current estimate $\hat{p}_i$ of $D(f)$ together with the estimates returned by the oracles $PSTAT(f, D)$ and $ISTAT(D)$: at each statistical query (A, τ), return $\hat{D}(f \cap A) = \hat{D}_f(A) \times \hat{p}_i$ and $\hat{D}(\overline{f} \cap A) = \hat{D}(A) - \hat{D}(f \cap A)$ where $\hat{D}_f(A)$ is the estimate given by the positive statistical oracle $PSTAT(f, D)$ with set A within tolerance $\tau_{\min}/4$ and where $\hat{D}(A)$ is the estimate given by the instance statistical oracle with set A within tolerance $\tau_{\min}/4$. Note that the simulation of $STAT(f, D)$ may produce erroneous results when the estimate $\hat{p}_i$ of $D(f)$ is poor. In this case, the behavior of the algorithm *L* is not known. Thus, we bound the running time of *L* and output a default hypothesis.

In the second stage, the algorithm *PL* selects the hypothesis h which minimizes the quantity $\hat{e}(h)$. Minimizing $\hat{e}(h)$ is equivalent to minimizing an estimate of the error rate according to the noisy oracle defined in the proof of Proposition 1: with probability $\frac{2}{3}$ draw a positive example and label it as positive and with probability $\frac{1}{3}$ draw an unlabeled example and label it as negative. Indeed, if an unlabeled example is drawn, the probability of error is equal to $D(h)$. And if a positive example is drawn, the probability of error is equal to $D_f(\overline{h})$. That is, the error rate using the noisy oracle is $(2D_f(\overline{h}) + D(h))/3$.

Minimizing $\hat{e}(h)$ can also be seen as: choosing a hypothesis h approximately consistent with positive data—when minimizing the first term of the sum $2\hat{D}_f(\overline{h_i})$—while avoiding over-generalization—when minimizing the second term $\hat{D}(h_i)$.

Note that as the statistical oracles $PSTAT(f, D)$ and $ISTAT(D)$ can be simulated by using positive and unlabeled examples. Consequently, the previous scheme allows to transform any SQ-like learning algorithm into an algorithm using positive and unlabeled examples only.

3.2. Proof of the algorithm

Lemma 3. *There exists* $i \in \{1, \ldots, N\}$ *such that* $error(h_i) \leqslant \varepsilon'$.

Proof. There exists i such that $D(f) \in [\hat{p}_i - \alpha, \hat{p}_i + \alpha]$ since, by definition of $\hat{p}_i$, $\bigcup_i [\hat{p}_i - \alpha, \hat{p}_i + \alpha] = [0, 1]$. For that value, $\hat{p}_i$ is an estimate of $D(f)$ within tolerance $\tau_{\min}/4$ since $\alpha \leqslant \tau_{\min}/4$. For all queries made by L, the oracles *PSTAT* and *ISTAT* are called with tolerance $\tau_{\min}/4$ and Eqs. (3) are used. It is easy to prove that estimates for algorithm L are made within tolerance $\tau_{\min}$. Consequently, by hypothesis on L, L outputs some h_i such that $error(h_i) \leqslant \varepsilon'$. □

Lemma 4. *Let* f *be the target concept, let* g *be some hypothesis and let* $\beta \geqslant 2D(f)$. *We have*

$$error(g) \leqslant e_\beta(g) - D(f) \leqslant error(g)\left(\frac{\beta - D(f)}{D(f)}\right),$$

where $error(g) = D(f \Delta g)$ *is the (classical) error and* $e_\beta(g) = \beta D_f(\overline{g}) + D(g)$.

Proof. We have

$$\begin{aligned} error(g) &= D(f \cap \overline{g}) + D(g \cap \overline{f}) \\ &= D(g) - D(f) + 2D(f \cap \overline{g}) \\ &= e_\beta(g) - D(f) + 2D(f \cap \overline{g}) - \beta D_f(\overline{g}) \\ &= e_\beta(g) - D(f) + D(f \cap \overline{g})\left(2 - \frac{\beta}{D(f)}\right) \end{aligned}$$

$$e_\beta(g) - D(f) = error(g) + D(f \cap \overline{g})\frac{\beta - 2D(f)}{D(f)}.$$

As $\beta \geqslant 2D(f)$ and $D(f \cap \overline{g}) \leqslant error(g)$, we have

$$error(g) \leqslant e_\beta(g) - D(f) \leqslant error(g)\left[1 + \frac{\beta - 2D(f)}{D(f)}\right] = error(g)\,\frac{\beta - D(f)}{D(f)}. \qquad \square$$

Note that the learner is not given access to an upper bound on $D(f)$. The previous lemma holds if $\beta \geqslant 2D(f)$, thus we set β to 2 and we simply denote $e_2(h)$ by $e(h)$. That is we have: $e(g) = 2D_f(\overline{g}) + D(g)$ and the reader should note that in the hypothesis testing stage of the algorithm *PL* we use an estimate $\hat{e}(h)$ of $e(h)$ where h is a hypothesis in the hypothesis set.

Lemma 5. *Let* h *and* h' *be two hypotheses such that* $error(h) \leqslant \frac{1}{2} \times \gamma/(2-\gamma) \times \varepsilon$ *and* $error(h') > \varepsilon$, *then* $e(h') - e(h) > \varepsilon/2$.

Proof. Using the previous lemma – with $\beta = 2$ –, we have

$$e(h') - e(h) \geqslant error(h') - error(h)\left(\frac{2 - D(f)}{D(f)}\right).$$

As the function $r(x) = (2-x)/x$ is decreasing and $D(f) \geqslant \gamma$, we have

$$e(h') - e(h) \geqslant error(h') - error(h)\left(\frac{2-\gamma}{\gamma}\right).$$

By hypothesis on h and h',

$$error(h) < \frac{1}{2}\left(\frac{\gamma}{2-\gamma}\right) error(h'),$$

so

$$e(h') - e(h) > \frac{error(h')}{2} > \varepsilon/2. \qquad \square$$

Proposition 6. *The output hypothesis satisfies* $error(h) \leqslant \varepsilon$ *and the running time is polynomial in* $1/\varepsilon$, n, l *and* $1/\gamma$.

Proof. All estimates $\hat{e}(h_i)$ of $e(h_i)$ are done within tolerance $\varepsilon/4$ and Lemmas 3 and 5 ensure that the output hypothesis satisfies $error(h) \leqslant \varepsilon$. $\square$

The number of hypotheses is N which is linear in $1/\tau_{\min}$. We have supposed for sake of clarity in the definition of the algorithm that $\tau_{\min}$ was fixed and known by the learner. Actually, $\tau_{\min}$ is polynomial in the input accuracy of L, therefore $\tau_{\min}$ is polynomial in ε' that is also polynomial in ε and γ. It is easy to verify that all queries are made within a tolerance polynomial in ε and γ.

3.3. Equivalence of the SQ and POSQ models

Whether or not any SQ algorithm can be transformed into a POSQ algorithm remains an open question. It has been proved in [7] that this transformation is possible when the weight of the target concept can be estimated from the oracles $PSTAT(f, D)$ and $ISTAT(D)$ in polynomial time. In this paper, we improve this result by showing that any SQ algorithm can be transformed into a POSQ algorithm when a lower bound on the weight of the target concept is given to the learner. However, the running time of the algorithm is polynomial in the inverse of this lower bound.

Let us consider a concept class $\mathcal{C}$ which is SQ learnable. We say that $\mathcal{C}$ satisfies the property *Lowerbound* if there exists an algorithm W which, for any f in $\mathcal{C}$, for any distribution D on X, with input ε and given access to $PSTAT(f, D)$ and $ISTAT(D)$

$$\text{outputs} \begin{cases} yes & \text{if } D(f) < \dfrac{\varepsilon}{2}, \\ no & \text{if } D(f) > \varepsilon, \\ ? & \text{if } \dfrac{\varepsilon}{2} \leqslant D(f) \leqslant \varepsilon \end{cases}$$

in time polynomial in $1/\varepsilon$. Then we have the following result:

Proposition 7. *Any SQ learnable class which satisfies Lowerbound is POSQ learnable.*

Proof. Consider the following algorithm:
input: ε
  if W outputs yes
    output function 0
  else
    run the POSQ learning algorithm with parameter $\gamma = \varepsilon/2$ and input ε

It is easy to prove that this algorithm is a learning algorithm from positive and instance statistical queries using Proposition 6 and the definition of the property *Lowerbound*. $\square$

Note that proving the property *Lowerbound* for every SQ learnable concept class would imply the equality between SQ and POSQ.

4. Decision tree learning algorithms from positive and unlabeled examples

Induction tree algorithms are widely used for data mining purposes. These algorithms are "SQ like" since they only use examples in order to estimate probabilities. In the first part of this section, we recall the notions of *entropy* and *information gain* on which C4.5 is based. In the second part, we introduce C4.5POSUNL, a learning algorithm based

on C4.5 first defined in [6], where the statistical queries required by C4.5 are estimated with the help of Eqs. (3), an estimate of the weight of the target concept being given as input. In the third part of this section, we present POSC4.5 an induction tree learning algorithm from positive data and unlabeled data only. In the last part of this section, we give experimental results for POSC4.5 both on artificial problems and on two benchmarks chosen from the *UCI Machine Learning Database*.

4.1. Top down decision tree algorithms

Most algorithms for tree induction use a top-down, greedy search through the space of decision trees. The *splitting criterion* used by C4.5 [17] is based on a statistical property, called *information gain*, itself based on a measure from information theory, called *entropy*. We only consider binary problems. Given a sample S of some target concept, the entropy of S is

$$Entropy(S) = -p_0 \log_2 p_0 - p_1 \log_2 p_1, \tag{7}$$

where p_i is the proportion of examples in S belonging to the class i. The information gain is the expected reduction in entropy by partitioning the sample according to an attribute test t. It is defined as

$$Gain(S, t) = Entropy(S) - \sum_{v \in Values(t)} \frac{N_v}{N} Entropy(S_v), \tag{8}$$

where $Values(t)$ is the set of every possible value for the attribute test t, N_v is the cardinality of the set S_v of examples in S for which t has value v and N is the cardinality of S.

As the information gain criterion has a strong bias in favor of tests with many outcomes, the criterion used in C4.5 is the *Gain ratio* defined by

$$GainRatio(S, t) = \frac{Gain(S, t)}{SplitInfo(S, t)},$$

where

$$SplitInfo(S, t) = -\sum_{v \in Values(t)} \frac{N_v}{N} \log \frac{N_v}{N}.$$

Let D be the hidden distribution defined over the set of instances. Let n be the current node, let D_n be the filtered distribution, that is the hidden distribution D restricted to instances reaching the node n. Let S be the set of training examples associated with the current node n and let p_1 be the proportion of positive examples in S: p_1 is an estimate of $D_n(f)$ and p_0 is an estimate of $D_n(\overline{f})$.

4.2. C4.5POSUNL: a top-down induction tree algorithm from positive and unlabeled examples with the help of an estimate of the weight of the target concept

Roughly speaking, C4.5POSUNL is a version of C4.5 in which the statistical queries are estimated from positive examples and unlabeled examples by using Eqs. (3), an estimate of the weight of the target concept being given. The differences between C4.5POSUNL and C4.5 are the following:

- C4.5POSUNL takes as input:
  - a set *POS* of positive examples,
  - together with a set *UNL* of unlabeled examples,
  - together with an estimate $\hat{D}(f)$ of $D(f)$ which is the weight of the target concept.
- For the current node, entropy and gain are calculated using Eqs. (7) and (8) where, based on Eqs. (3), the ratios p_0 and p_1 are given by the equations:

$$p_1 = \inf\left\{\frac{|POS^n|}{|POS|} \times \hat{D}(f) \times \frac{|UNL|}{|UNL^n|}, 1\right\},$$
$$p_0 = 1 - p_1, \tag{9}$$

POSC4.5
input: a set POS of positive examples and a set UNL of unlabeled examples
Split POS and UNL with ratios $2/3$, $1/3$ into POS_L, POS_T, UNL_L and UNL_T
Construction of a hypothesis set
for $i = 1$ **to** 9
  the current estimate of $D(f)$ is set to $\hat{D}(f) = \frac{i}{10}$
  run C4.5POSUNL with input POS_L, UNL_L and $\hat{D}(f) = \frac{i}{10}$, and output h_i
Selecting the best estimate of $D(f)$
for $i = 1$ **to** 9
  set $\hat{e}(h_i)$ to $2\frac{|\{x \in POS_T | h_i(x)=0\}|}{|POS_T|} + \frac{|\{x \in UNL_T | h_i(x)=1\}|}{|UNL_T|}$
set j to $\underset{i}{\operatorname{argmin}}\ \hat{e}(h_i)$
Construction of the final hypothesis
run C4.5POSUNL with input POS, UNL and $\hat{D}(f) = \frac{j}{10}$, and **output** h

Fig. 2. **POSC4.5:** induction tree algorithm from positive and unlabeled examples.

where POS^n is the set of positive examples associated with the node n and UNL^n is the set of unlabeled examples associated with the node n.

- When the gain ratio is used instead of the information gain, split information *SplitInfo* is calculated from unlabeled examples.
- The majority class is chosen as 0 or 1 according to the values of p_0 and p_1 calculated with Eqs. (9).
- Halting criteria during the top-down tree generation are evaluated from unlabeled data.
- When pruning trees, classification errors are estimated with the help of ratios p_0 and p_1 from (9).

4.3. POSC4.5: a top-down induction tree algorithm from positive and unlabeled examples only

The learning algorithm POSC4.5 is given in Fig. 2. It is based on the theoretical result proved in Section 3. We intend to use the algorithm scheme *PL* to transform C4.5. But as C4.5POSUNL can already be viewed as a variant of C4.5 which uses positive and unlabeled examples together with an estimate of the target weight, we have directly incorporated C4.5POSUNL in the *PL* algorithm.

Another difference between *PL* and POSC4.5 is that the lower bound γ is not given as input to POSC4.5. Instead, it is implicitly supposed that the weight of the target concept is not too small.

The algorithm takes as input a set *POS* of examples of the target class together with a set *UNL* of unlabeled examples. The algorithm splits the set *POS* (respectively *UNL*) into two sets POS_L and POS_T (respectively UNL_L and UNL_T) using the usual values $\frac{2}{3}$ and $\frac{1}{3}$.

The sets POS_L and UNL_L are used for the construction of the hypothesis set. More precisely, these sets are used to simulate the positive statistical oracle and the instance statistical oracle. In this stage, we run nine times C4.5POSUNL with input POS_L, UNL_L and an estimate $\hat{D}(f)$ of $D(f)$ taking the successive values $0.1, \ldots, 0.9$.

In the second stage of POSC4.5, i.e. the hypothesis testing algorithm, the sets POS_T and UNL_T are used to simulate the positive statistical oracle and the instance statistical oracle. In our implementation, we select in POSC4.5 the best estimate $\hat{D}(f)$ of $D(f)$ according to the minimal estimate $\hat{e}(h)$ of $e(h)$ instead of selecting the best hypothesis like in *PL*.

The output of POSC4.5 is the output of C4.5POSUNL with input *POS*, *UNL* together with the best estimate $\hat{D}(f)$ of $D(f)$.

4.4. Experiments with decision lists

A *decision list* over $x_1, \ldots, x_n$ is an ordered sequence $L = (m_1, b_1), \ldots, (m_p, b_p)$ of terms, in which each m_j is a monomial over $x_1, \ldots, x_n$, and each $b_j \in \{0, 1\}$. The last monomial is always $m_p = 1$. For any input $a \in \{0, 1\}^n$, the

value $L(a)$ is defined as b_j, where j is the smallest index satisfying $m_j(a) = 1$. We only consider 1-decision list where each monomial is a variable x_i or its negation $\overline{x_i}$. We set p to 11 and n to 20. The choice of a target decision list f, the choice of the weight $D(f)$ and the choice of the distribution D are done as follows:

- a target decision list f is chosen randomly;
- for any $a \in \{0,1\}^n$, a weight w_a is chosen randomly in $[0,1)$;
- a normalization procedure is applied to the two sets of weights $\{w_a \mid f(a) = 1\}$ and $\{w_a \mid f(a) = 0\}$. Thus, we get two distributions D_1 on f and D_2 on $\overline{f}$;
- a weight $D(f)$ for the target concept is chosen using a procedure that depends on the experiment;
- D is defined by: $\forall a \in \{0,1\}^n$, $D(a) = D(f) \times D_1(a) + (1 - D(f)) \times D_2(a)$.

In the experiments, we compare C4.5POSUNL and POSC4.5. The algorithm C4.5POSUNL takes as input a set *POS* of positive examples, a set *UNL* of unlabeled examples and an estimate $\hat{D}(f)$ of $D(f)$. The experimental results for C4.5POSUNL depend on the accuracy of the estimate $\hat{D}(f)$ of $D(f)$. Thus, we consider two cases:

- the exact value of $D(f)$ is given as input of the learning algorithm. In the following and in the figures, we denote by C4.5POSUNL($D(f)$) this variant of C4.5POSUNL;
- the estimate $\hat{D}(f)$ is set to the ratio of positive examples in a (small) set *LAB* of labeled examples given as input. We denote by C4.5POSUNL(*LAB*) this variant of C4.5POSUNL. The set *LAB* is only used for the calculation of $\hat{D}(f)$.

In the experimental results and in the plots, the error rates and target weights are expressed in percent. The size of a set is its cardinality.

Experiment 1. In order to obtain experimental results on the relative value of examples, we let the number of positive examples vary and we compare POSC4.5, C4.5POSUNL(*LAB*) and C4.5POSUNL($D(f)$). We set $D(f)$ to 0.5, the size of *POS* is equal to the size of *UNL* and ranges from 50 to 1000 by step 50, the size of *LAB* is fixed to 25. For a given size of *POS*, we iterate 100 times the experiment: a target f is drawn, a distribution D is chosen, sets *LAB*, *POS* and *UNL* are drawn randomly, we run the three algorithms and calculate the error rate of the output hypothesis on a large test set of 10 000 examples. We average the error rates over the 100 experiments. The results are given in Fig. A.1.

The learning algorithm POSC4.5 performs as well as C4.5POSUNL($D(f)$) where the exact value of $D(f)$ is given to the learner. Thus for this artificial problem, the results of POSC4.5 which is based on a hypothesis testing algorithm are convincing. The reader should also note that the two algorithms POSC4.5 and C4.5POSUNL($D(f)$) outperform C4.5POSUNL(*LAB*) which uses a rough estimate of $D(f)$ (solely based on 25 labeled examples). In this first set of experiments, the weight of the target concept is set equal to 0.5. An equal ratio between positive and negative examples is the most favorable to POSC4.5. Therefore, in a second set of experiments, we consider different values for $D(f)$.

Experiment 2. The weight $D(f)$ of the target concept ranges from 0 to 1 by step 0.05. The size of *POS* is equal to the size of *UNL* and is set to 1000. The size of *LAB* is fixed to 25. For a given value of $D(f)$, we average the error rates over 100 experiments. The results are given in Fig. A.2.

The results are similar: POSC4.5 performs as well as C4.5POSUNL($D(f)$); POSC4.5 and C4.5POSUNL($D(f)$) outperform C4.5POSUNL(*LAB*). For this set of experiments, POSC4.5 is robust to the value of the weight of the target concept. Note that the plots for $D(f) = 0.05$ and 0.95 are not significant because POSC4.5 makes its guesses from 0.1 to 0.9.

4.5. Experiments with UCI problems

We consider two data sets from the *UCI Machine Learning Database* [13]: `kr-vs-kp` and `adult`. The majority class is chosen as positive. In the experiments, we compare C4.5POSUNL and POSC4.5 with C4.5.

Experiment 3. In order to obtain experimental results for the relative value of examples, we compare C4.5 and C4.5POSUNL(LAB). For `kr-vs-kp`, the size of *POS* and the size of *UNL* are set equal to 600; the error rate is estimated on a hold-out test set of 1000 labeled examples. For `adult`, the size of *POS* and the size of *UNL* are set equal to 10 000; the error rate is estimated on a hold-out test set of 15 000 labeled examples. We let the number of labeled examples vary, and compare the error rate of C4.5 and C4.5POSUNL(*LAB*). For a given size of *LAB*, we iterate 100 times the following: all sets are selected randomly, we compute the error rate for C4.5 with input *LAB* and the error rate for C4.5POSUNL with input *POS*, *UNL* and an estimate of the weight of the target concept which is the ratio of positive examples in the set *LAB*. The reader should note that C4.5POSUNL(*LAB*) only uses labeled examples to compute a rough estimate of the weight of the target concept. Then, we average the error rates over the 1000 experiments.The results can be seen in Fig. A.3.

For the two datasets, C4.5POSUNL(*LAB*) outperforms C4.5 when the number of labeled examples is small until a limit which is about 100 for `kr-vs-kp`—recall that there are 600 positive examples and 600 unlabeled examples—and about 500 for `adult`—recall that there are 10 000 positive examples and 10 000 unlabeled examples. One could also note that, when the estimate of the weight of the target concept is precise enough, the error rate for is C4.5POSUNL constant. Also note that C4.5POSUNL trees are consistently larger than C4.5 ones because of the pruning procedure in C4.5POSUNL which is not optimized.

Experiment 4. In this second set of experiments, we fix the size of *LAB* and we let the number of positive and unlabeled examples vary, and compare the error rate of C4.5POSUNL(*LAB*), C4.5POSUNL($D(f)$) and OSC4.5. The results can be seen in Fig. A.4. For `kr-vs-kp`, the plots are similar, the least good results are obtained by POSC4.5. This seems natural because it uses less information. Surprisingly, POSC4.5 obtains the best results for the data set `adult`.

5. Conclusion

We have given evidence in the present paper that the weight of the target concept is a key parameter for learning from positive data and unlabeled data. In the co-training framework [3], it seems that the weight of the target concept is implicitly known by the learner. The ratio of positive examples in the labeled training sample is set to the weight of the target concept and this ratio is preserved throughout the learning process. It is unclear whether the results depend on this implicit hypothesis.

In this paper, we have shown that knowledge of a lower bound of the target weight is sufficient when learning from positive and unlabeled data. Nevertheless the equivalence between SQ and POSQ remains open. In the semi-supervised setting as in our setting of learning from positive and unlabeled examples, it should be interesting to investigate the relative value of examples (labeled examples vs positive examples vs unlabeled examples). Also it should be clear that more experimental results are needed. We are currently applying the results of the present paper to real-world text mining problems using the naive Bayes algorithm.

Lastly, it is now a challenging problem to find algorithms from positive data and unlabeled data when the weight of the target concept is quite small because many applications fall in this case. For imbalanced classes the classifier's performance cannot be expressed in terms of the accuracy: if only 1% examples are positive the default hypothesis achieves an accuracy of 99%. Thus, another criterion of success for the learning algorithm should be used, say for example the geometric mean of accuracies observed separately on positive examples, and on negative examples. We also plan to investigate this problem, but it is known to be difficult even when learning from labeled data.

Appendix A. Experimental results

The experimental results are summarized in Figs A.1–A.4.

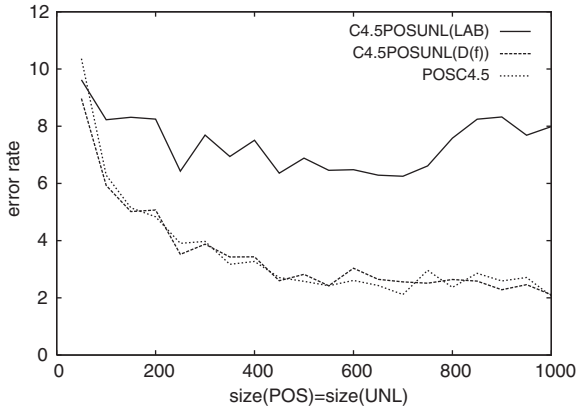

Fig. A.1. We consider decision lists where $D(f) = 0.5$. We compare C4.5POSUNL(*LAB*) where the estimate of $D(f)$ is done on a small random set of 25 labeled examples, C4.5POSUNL($D(f)$) where the exact value of $D(f)$ is given as input and POSC4.5. The three algorithms take as input a set *POS* and a set *UNL* where $size(POS) = size(UNL)$ ranges from 50 to 1000 by step 50.

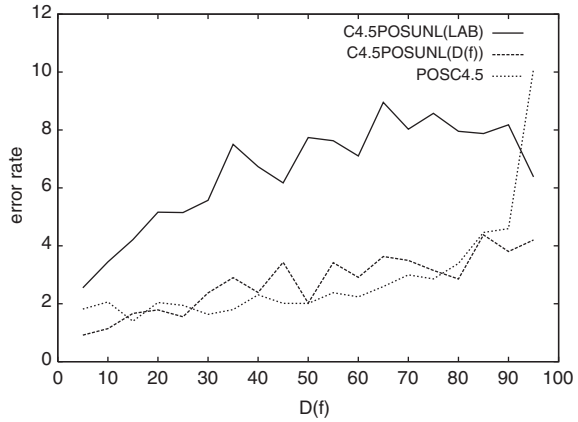

Fig. A.2. We consider decision lists where $D(f)$ ranges from 0 to 1 by step 0.05. We compare C4.5POSUNL(*LAB*) where the estimate of $D(f)$ is done on a small random set of 25 labeled examples, C4.5POSUNL($D(f)$) where the exact value of $D(f)$ is given as input and POSC4.5. The three algorithms take as input a set *POS* and a set *UNL* where $size(POS) = size(UNL) = 1000$.

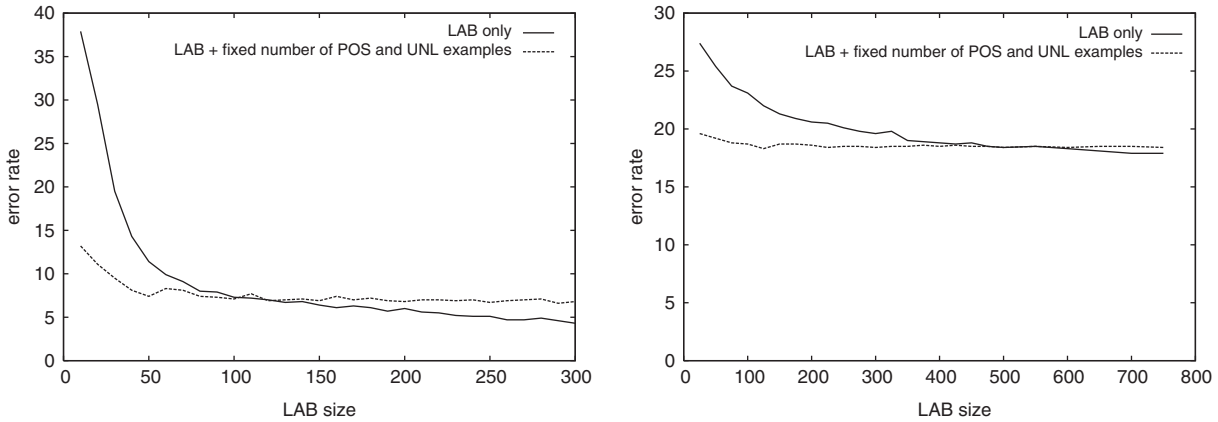

Fig. A.3. Error rate of C4.5 and C4.5POSUNL(*LAB*) averaged over 100 trials on the `kr-vs-kp` data set (left plot) and on the `adult` data sets (right plot).

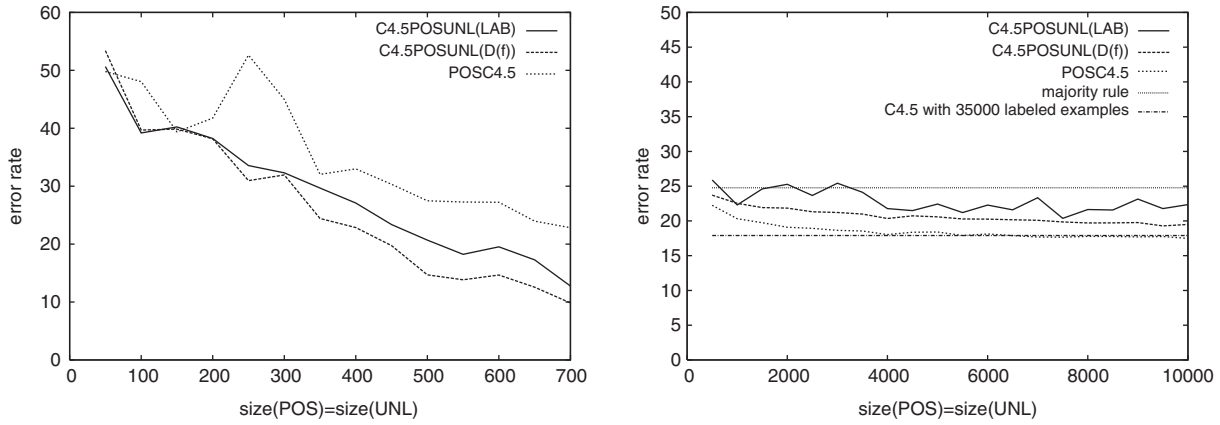

Fig. A.4. The `kr-vs-kp` data set corresponds to the left plot where $size(LAB) = 25$, $size(POS) = size(UNL)$ ranges from 50 to 700 by step 50 and $D(f) \approx 0.5$; the `adult` data set corresponds to the right plot where $size(LAB) = 25$, $size(POS) = size(UNL)$ ranges from 500 to 10 000 by step 500 and $D(f) \approx 0.75$.

References

[1] D. Angluin, P. Laird, Learning from noisy examples, Machine Learning 2 (4) (1988) 343–370.
[2] A. Blum, A. Kalai, H. Wasserman, Noise-tolerant learning, the parity problem, and the statistical query model, in: Proc. 32nd Annu. ACM Symp. on Theory of Computing, 2000, pp. 435–440.
[3] A. Blum, T. Mitchell, Combining labeled and unlabeled data with co-training, in: Proc. 11th Annu. Conf. on Computational Learning Theory, 1998, pp. 92–100.
[4] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and regression trees, Technical Report, Wadsworth International, Monterey, CA, 1984.
[5] S.E. Decatur, Pac learning with constant-partition classification noise and applications to decision tree induction, in: Proc. 14th Internat. Conf. on Machine Learning, 1997, pp. 83–91.
[6] F. DeComité, F. Denis, R. Gilleron, F. Letouzey, Positive and unlabeled examples help learning, in: Proc. 10th Internat. Conf. on Algorithmic Learning Theory, 1999, pp. 219–230.
[7] F. Denis, PAC learning from positive statistical queries, in: Proc. 9th Internat. Conf. on Algorithmic Learning Theory, 1998, pp. 112–126.
[8] S. Goldman, Y. Zhou, Enhancing supervised learning with unlabeled data, in: Proc. 17th Internat. Conf. on Machine Learning, 2000, pp. 327–334.
[9] D. Haussler, M. Kearns, N. Littlestone, M.K. Warmuth, Equivalence of models for polynomial learnability, Inform. Comput. 95 (2) (1991) 129–161.
[10] J. Jackson, On the efficiency of noise-tolerant PAC algorithms derived from statistical queries, in: Proc. 13th Annu. Conf. on Computational Learning Theory, 2000, pp. 7–15.
[11] T. Joachims, Transductive inference for text classification using support vector machines, in: Proc. 16th Internat. Conf. on Machine Learning, 1999, pp. 200–209.
[12] M. Kearns, Efficient noise-tolerant learning from statistical queries, in: Proc. 25th ACM Symp. on the Theory of Computing, 1993, pp. 392–401.
[13] C.J. Merz, P.M. Murphy, UCI repository of machine learning databases, 1998.
[14] T. Mitchell, Machine learning and data mining, Commun. ACM 42 (11) (1999) 30–36.
[15] K. Nigam, R. Ghani, Analyzing the applicability and effectiveness of co-training, in: Proc. 9th Internat. Conf. on Information and Knowledge Management, 2000, pp. 86–93.
[16] K. Nigam, A.K. McCallum, S. Thrun, T.M. Mitchell, Text classification from labeled and unlabeled documents using EM, Machine Learning 39 (2/3) (2000) 103–134.
[17] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, Los Altos, CA, 1993.
[18] L.G. Valiant, A theory of the learnable, Commun. ACM 27 (11) (1984) 1134–1142.