Pattern Recognition Letters xxx (2013) xxx–xxx

Contents lists available at SciVerse ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

A bagging SVM to learn from positive and unlabeled examples

F. Mordelet [a,*], J.-P. Vert [b,c,d]

[a] Duke University, LSRC Building, 308 Research Drive, Durham, NC 27708, USA
[b] Mines ParisTech, Centre for Computational Biology, 77300 Fontainebleau, France
[c] Institut Curie, 75005 Paris, France
[d] INSERM, U900, 75005 Paris, France

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
PU learning
Bagging
SVM

ABSTRACT

We consider the problem of learning a binary classifier from a training set of positive and unlabeled examples, both in the inductive and in the transductive setting. This problem, often referred to as *PU learning*, differs from the standard supervised classification problem by the lack of negative examples in the training set. It corresponds to an ubiquitous situation in many applications such as information retrieval or gene ranking, when we have identified a set of data of interest sharing a particular property, and we wish to automatically retrieve additional data sharing the same property among a large and easily available pool of unlabeled data. We propose a new method for PU learning with a conceptually simple implementation based on bootstrap aggregating (bagging) techniques: the algorithm iteratively trains many binary classifiers to discriminate the known positive examples from random subsamples of the unlabeled set, and averages their predictions. We show theoretically and experimentally that the method can match and even outperform the performance of state-of-the-art methods for PU learning, particularly when the number of positive examples is limited and the fraction of negatives among the unlabeled examples is small. The proposed method can also run considerably faster than state-of-the-art methods, particularly when the set of unlabeled examples is large.

1. Introduction

In many applications, such as information retrieval or gene ranking, one is given a finite set of data of interest sharing a particular property, and wishes to find other data sharing the same property. In information retrieval, for example, the finite set can be a user query, or a set of documents known to belong to a specific category, and the goal is to scan a large database of documents to identify new documents related to the query or belonging to the same category. In gene ranking, the query is a finite list of genes known to have a given function or to be associated to a given disease, and the goal is to identify new genes sharing the same property (Aerts et al., 2006). In fact this setting is ubiquitous in many applications where identifying a data of interest is difficult or expensive, e.g., because human intervention is necessary or expensive experiments are needed, while unlabeled data can be easily collected. In such cases there is a clear opportunity to alleviate the burden and cost of interesting data identification with the help of machine learning techniques.

More formally, let us assign a binary label to each possible data: positive $(+1)$ for data of interest, negative (-1) for other data. Unlabeled data are data for which we do not know whether they are interesting or not. Denoting $\mathcal{X}$ the set of data, we assume that the "query" is a finite set of data $\mathcal{P} = \{x_1, \ldots, x_m\} \subset \mathcal{X}$ with positive labels, and we further assume that we have access to a (possibly large) set $\mathcal{U} = \{x_{m+1}, \ldots, x_n\} \subset \mathcal{X}$ of unlabeled data. Our goal is to learn, from $\mathcal{P}$ and $\mathcal{U}$, a way to identify new data with positive labels, a problem often referred to as *PU learning*. More precisely we make a distinction between two flavors of PU learning:

- *Inductive PU learning*, where the goal is to learn from $\mathcal{P}$ and $\mathcal{U}$ a function $f : \mathcal{X} \to \mathbb{R}$ able to associate a score or probability to be positive $f(x)$ to any new data $x \in \mathcal{X}$, which may not be in the training set of unlabeled data $\mathcal{U}$. This may typically be the case in an image or document classification system, where a subset of the web is used as unlabeled set $\mathcal{U}$ to train the system, which must then be able to scan any new image or document out of the training set.
- *Transductive PU learning*, where the goal is to estimate a scoring function $s : \mathcal{U} \to \mathbb{R}$ from $\mathcal{P}$ and $\mathcal{U}$, i.e., where we are just interested is finding positive data in the set $\mathcal{U}$. This is typically the case in the disease gene ranking application, where the full set of human genes is known during training and split between known disease genes $\mathcal{P}$ and the rest of the genome $\mathcal{U}$. In that case we are only interested in finding new disease genes in $\mathcal{U}$.

* Corresponding author. Tel.: +1 919 684 2124.
E-mail addresses: fantine.mordelet@duke.edu (F. Mordelet), jean-philippe.vert@mines-paristech.fr (J.-P. Vert).

0167-8655/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.patrec.2013.06.010

A growing body of work has focused on PU learning recently. The fact that only positive and unlabeled examples are available prevents a priori the use of supervised classification methods, which require negative examples in the training set. A first approach to overcome the lack of negative examples is to disregard unlabeled examples during training and simply learn from the positive examples, e.g., by ranking the unlabeled examples by decreasing similarity to the mean positive example (Joachims, 1997) or using more advanced learning methods such as 1-class SVM (Schölkopf et al., 2001; Manevitz and Yousef, 2001; Vert and Vert, 2006; De Bie et al., 2007; Geurts, 2011)

Alternatively, the problem of inductive PU learning has been studied on its own from a theoretical viewpoint (Denis et al., 2005; Scott and Blanchard, 2009), and has given rise to a number of specific algorithms. Several authors have proposed two-step algorithms, heuristic in nature, which first attempt to identify negative examples in the unlabeled set, and then estimate a classifier from the positive, unlabeled and likely negative examples (Manevitz and Yousef, 2001; Liu et al., 2002, 2003; Li and Liu, 2003; Yu et al., 2004). Alternatively, it was observed that directly learning to discriminate $\mathcal{P}$ from $\mathcal{U}$, possibly after rebalancing the misclassification costs of the two classes to account for the asymmetry of the problem, leads to state-of-the-art results for inductive PU learning. This approach has been studied, with different weighting schemes, using a logistic regression or a SVM as binary classifier (Liu et al., 2003; Lee and Liu, 2003; Elkan and Noto, 2008). Inductive PU learning is also related to and has been used for novelty detection, when $\mathcal{P}$ is interpreted as "normal" data and $\mathcal{U}$ contains an arbitrary fraction π of negative or "novel" examples (Scott and Blanchard, 2009), or to data retrieval from a single query, when $\mathcal{P}$ is reduced to a singleton (Shah et al., 2008).

Transductive PU learning is arguably easier than inductive PU learning, since we know in advance the data to be screened for positive labels. Many semi-supervised methods have been proposed to tackle transductive learning when both positive and negative examples are known during training, including transductive SVM (Joachims, 1999), or many graph-based methods, reviewed by Chapelle et al. (2006). Comparatively little effort has been devoted to the specific transductive PU learning problem, with the notable exception of Liu et al. (2002), who call the problem *partially supervised classification* and proposes an iterative method to solve it, and Pelckmans and Suykens (2009) who formulate the problem as a combinatorial optimization problem over a graph. Finally, Sriphaew et al. (2009) recently proposed a bagging approach which shares similarities with ours, but is more complex and was only tested on a specific application.

Several methods for PU learning, reviewed above, reduce the problem to a binary classification problem where we learn to discriminate $\mathcal{P}$ from $\mathcal{U}$. This can be theoretically justified, at least asymptotically, since the ratio between the conditional distributions of positive and unlabeled examples is monotonically increasing with the ratio of positive and negative examples (Elkan and Noto, 2008; Scott and Blanchard, 2009), and has given rise to state-of-the-art methods such as biased SVM (Liu et al., 2003) or weighted logistic regression (Lee and Liu, 2003). Although this reduction suggests that virtually any method for (weighted) supervised binary classification can be used to solve PU learning problems, we put forward in this paper that some methods may be more adapted than others in a non-asymptotic setting, due to the particular structure of the unlabeled class. In particular, we investigate the relevance of methods based on aggregating classifiers trained on artificially perturbed training sets, in the spirit of bagging (Breiman, 1996, 2001). Such methods are known to be relevant to improve the performance of unstable classifiers, a situation which, we propose, may occur particularly in PU learning. Indeed, in addition to the usual instability of learning algorithms confronted to a finite-size training sets, the content of a random subsample of unlabeled data in positive and negative examples is likely to strongly affect the classifier, since the contamination of $\mathcal{U}$ with positive examples makes the problem more difficult. Variations in the contamination rate of $\mathcal{U}$ may thus have an important impact on the trained classifier, in that a higher contamination rate makes the problem harder in practice (Scott and Blanchard, 2009), a situation which bagging-like classifiers may benefit from.

Based on this idea, we propose a general and simple scheme for inductive PU learning, akin to an asymmetric form of bagging for supervised binary classification. The method, which we call *bagging SVM*, consists in aggregating classifiers trained to discriminate $\mathcal{P}$ from a small random subsample of $\mathcal{U}$, where the size of the random sample plays a specific role. This method can naturally be adapted to the transductive PU learning framework. We demonstrate on simulated and real data that bagging SVM performs at least as well as existing methods for PU learning, while being often faster in particular when $|\mathcal{P}| \ll |\mathcal{U}|$.

This paper is organized as follows. We present and study theoretically the bagging SVM for inductive PU learning in Section 2.1. Its extension to transductive PU learning is considered in Section 2.2. Experimental results are presented in Section 3, followed by a discussion in Section 4.

2. Methods

2.1. Bagging for inductive PU learning

Our starting point to learn a classifier in the PU learning setting is the observation that learning to discriminate positive from unlabeled samples is a good proxy to our objective, which is to discriminate positive from negative samples. Even though the unlabeled set is contaminated by hidden positive examples, it is generally admitted that its distribution contains some information which should be exploited. That is for instance, the foundation of semi-supervised methods.

Indeed, let us assume for example that positive and negative examples are randomly generated by class-conditional distributions $\mathbb{P}_+$ and $\mathbb{P}_-$ with densities h_+ and h_-. If we model unlabeled examples as randomly sampled from $\mathbb{P}_+$ with probability γ and from $\mathbb{P}_-$ with probability $1-\gamma$, then the distribution of unlabeled has a density

$$h_u = \gamma h_+ + (1-\gamma)h_- . \tag{1}$$

Now notice that

$$\frac{h_u(x)}{h_+(x)} = \gamma + (1-\gamma)\frac{h_-(x)}{h_+(x)} \tag{2}$$

showing that the ratio between the conditional distributions of positive and unlabeled examples is monotonically increasing with the ratio of positive and negative examples (Elkan and Noto, 2008; Scott and Blanchard, 2009). Hence any estimator of the conditional probability of positive vs. unlabeled data should in theory also be applicable to discriminate positive from negative examples. This is the case for example of logistic regression or some forms of SVM (Steinwart, 2003; Bartlett and Tewari, 2007). In practice it seems useful to train classifiers to discriminate $\mathcal{P}$ from $\mathcal{U}$ by penalizing more false negative than false positive errors, in order to account for the fact that positive examples are known to be positive, while unlabeled examples are known to contain hidden positives. Using soft margin SVM while giving high weights to false negative errors and low weights to false positive errors leads to the biased SVM approach described by Liu et al. (2003), while the same strategy using a logistic regression leads to the weighted logistic regression approach of Lee and Liu (2003). Both methods, tested on text categorization benchmarks, were shown to be very efficient in practice, and in particular outperformed all approaches based on heuristic identifications of true negatives in $\mathcal{U}$.

Among the many methods for supervised binary classification which could be used to discriminate $\mathcal{P}$ from $\mathcal{U}$, bootstrap aggregating or "bagging" is an interesting candidate (Breiman, 1996). The idea of bagging is to estimate a series of classifiers on datasets obtained by perturbing the original training set through bootstrap resampling, and to combine these classifiers by some aggregation technique. The method is conceptually simple, can be applied in many settings, and works very well in practice (Breiman, 2001; Hastie et al., 2001). Bagging generally improves the performance of individual classifiers when they are not too correlated to each other, which happens in particular when the classifier is highly sensitive to small perturbations of the training set. For example, Breiman (2001) showed that the difference between the expected mean square error (MSE) of a classifier trained on a single bootstrap sample and the MSE of the aggregated predictor increases with the variance of the classifier.

We propose that, by nature, PU learning problems have a particular structure that leads to instability of classifiers, which can be advantageously exploited by a bagging-like procedure which we now describe. Intuitively, an important source of instability in PU learning situations is the empirical contamination $\hat{\gamma}$ of $\mathcal{U}$ with positive examples, i.e., the percentage of positive examples in $\mathcal{U}$ which on average equals γ in (1). If by chance $\mathcal{U}$ is mostly made of negative examples, i.e., has low contamination by positive examples, then we will probably estimate a better classifier than if it contains mostly positive examples, i.e., has high contamination. Moreover, we can expect the classifiers in these different scenarios to be little correlated, since intuitively they estimate different log-ratios of conditional distribution. Hence, in addition to the "normal" instability of a classifier trained on a finite-size sample, which is exploited by bagging in general, we can expect an increased instability in PU learning due to the sensitivity of the classifier to the empirical contamination $\hat{\gamma}$ of $\mathcal{U}$ in positive examples. In order to exploit this sensitivity in a bagging-like procedure, we propose to randomly subsample $\mathcal{U}$ and train classifiers to discriminate $\mathcal{P}$ from each subsample, before aggregating the classifiers. By subsampling $\mathcal{U}$, we hope to vary in particular the empirical contamination between samples. This will induce a variety of situations, some lucky (small contamination), some less lucky (large contamination), which eventually will induce a large variability in the classifiers that the aggregation procedure can then exploit.

In opposition to classical bagging, the size K of the samples generated from $\mathcal{U}$ may play an important role to balance the accuracy against the stability of individual classifiers. On the one hand, larger subsamples should lead on average to better classifiers, since any classification method generally improves on average when more training points are available. On the other hand, the empirical contamination varies more for smaller subsamples.

To formalize a bit more this line of thought, let us denote by $\hat{\gamma}$ the true contamination rate in $\mathcal{U}$, that is, the true proportion of positive examples hidden in $\mathcal{U}$. Whenever a bootstrap sample $\mathcal{U}_t$ of size K is drawn from $\mathcal{U}$, its empirical number of positive examples is a binomial random variable $\sim B(K, \hat{\gamma})$, leading to a contamination rate $\hat{\gamma}_t$ with mean and variance:

$$\mathbb{E}(\hat{\gamma}_t) = \hat{\gamma} \quad \text{and} \quad \mathbb{V}(\hat{\gamma}_t) = \frac{1}{K}\hat{\gamma}(1-\hat{\gamma}).$$

The intuition that less contaminated samples allow to estimate better classifiers can be formalized in many different ways. Here we follow the analysis of Scott and Blanchard (2009) who study the inductive PU learning framework and consider the setting where we want to learn a function $f : \mathcal{X} \to \mathbb{R}$ which has a small probability of predicting negative examples as positives $R_-(f) = \mathbb{P}_-(f(X) > 0)$ for a probability of predicting positive examples as negatives $R_+(f) = \mathbb{P}_+(f(X) < 0)$ bounded by a fixed level $\alpha > 0$. For any level $\alpha > 0$, denoting by $R^*_{-,\alpha} = \inf_{f:R^+(f)\leqslant\alpha} R_-(f)$ the smallest possible risk, Scott and Blanchard (2009, Theorem 2) show that the excess risk of a function $\hat{f}_t$ trained to discriminate a set $\mathcal{P}$ of size P of positive examples from a subsample $\mathcal{U}_t$ of size K and contamination $\hat{\gamma}_t$ is upper bounded with large probability δ as follows:

$$L_{-,\alpha}(\hat{f}_t) = R_-(\hat{f}_t) - R^*_{-,\alpha} \leqslant \frac{\varepsilon_P + \varepsilon_K}{1-\hat{\gamma}_t},$$

where ε_i is an upper bound on the excess risk due to a finite sample of size i, typically proportional to $i^{-1/2}$ for a classifier trained to discriminate $\mathcal{P}$ from $\mathcal{U}_t$ by empirical risk minimization on a finite set (Scott and Nowak, 2005). This leads to an upper bound of the excess risk of the form:

$$L_{-,\alpha}(\hat{f}_t) \leqslant c\frac{P^{-1/2} + K^{-1/2}}{1-\hat{\gamma}_t}, \tag{3}$$

where c is a constant. Details relating constant c and the probability δ can be found in Scott and Blanchard (2009, Theorem 2). Eq. (3) shows that the quality of the estimator increases when the size of the unlabeled sample K increases and its contamination $\hat{\gamma}_t$ decreases, as expected. When we aggregate different classifiers $\hat{f}_t$ trained on subsamples with varying contamination $\hat{\gamma}_t$, we can expect that the excess risk of the aggregated classifier reaches the performance of individual classifiers with smaller-than-average contamination, typically with contamination $\hat{\gamma} - c_2\sqrt{\hat{\gamma}(1-\hat{\gamma})/K}$ where $c_2 > 0$ is a constant (independent of c). Plugging this estimate into (3), we obtain that the excess risk of the aggregated classifier is upper bounded with large probability by

$$c\frac{P^{-1/2} + K^{-1/2}}{1-\hat{\gamma} + c_2\sqrt{\hat{\gamma}(1-\hat{\gamma})}K^{-1/2}}. \tag{4}$$

Now we see that, when $P > \hat{\gamma}c_2^2/(1-\hat{\gamma})$, the upper bound (4) is a decreasing function of K and there is no apparent gain in subsampling and aggregating with $K < N$. On the other hand, when $P < \hat{\gamma}c_2^2/(1-\hat{\gamma})$, the upper bound (4) is an *increasing* function of K, suggesting that choosing $K < N$ may lead to more accurate classifiers. In words, when P is not too large, it may be better to subsample the set of unlabeled examples with $K < N$ samples and aggregated the resulting classifiers, because the gain in performance due to the stochastic decrease in contamination in a fraction of subsamples can be exploited by aggregation and outperforms the loss due the fact that each classifier is trained on a smaller data set.

In summary, the method we propose for PU learning is presented in Algorithm 1. It creates a series of classifiers trained to discriminate $\mathcal{P}$ from random subsamples of $\mathcal{U}$. The output of each of these classifiers is a function f_t that assigns a prediction score to any example. The score function f of the final aggregated classifier is simply defined as the average score of the individual classifiers.

Algorithm 1. Inductive bagging PU learning

```
INPUT: $\mathcal{P}, \mathcal{U}, K$ = size of bootstrap samples, $T$ = number of
  bootstraps
OUTPUT: a function $f : \mathcal{X} \to \mathbb{R}$
for $t = 1$ to $T$ do
  Draw a subsample $\mathcal{U}_t$ of size $K$ from $\mathcal{U}$.
  Train a classifier $f_t$ to discriminate $\mathcal{P}$ against $\mathcal{U}_t$.
end for
Return
  $f = \frac{1}{T}\sum_{t=1}^{T} f_t$
```

We call it bagging SVM when the classifier used to discriminate $\mathcal{P}$ from a random subsample of $\mathcal{U}$ is a biased SVM. It is akin to bagging to learn to discriminate $\mathcal{P}$ from $\mathcal{U}$, with two important specificities. First, only $\mathcal{U}$ is subsampled. This is to account for the fact that elements in $\mathcal{P}$ are known to be positive, and moreover that the number of positive examples is often limited. Second, the size of subsamples is a parameter K whose effect needs to be studied. If an optimal value exists, then this parameter may need to be adjusted.

The number T of bootstrap samples is also a user-defined parameter. Intuitively, the larger T the better, although we observed empirically little improvement for T larger than 100 (see Section 3, Fig. 3). Finally, although we propose to aggregate the T classifiers by a simple average, other aggregation rules could easily be used. On preliminary experiments on simulated and real data, we did not observed significant differences between the simple average and majority voting, another popular aggregation method.

2.2. Bagging SVM for transductive PU learning

We now consider the situation where the goal is only to assign a score to the elements of $\mathcal{U}$ reflecting our confidence that these elements belong to the positive class. Liu et al. (2002) have studied the same problem which they call "partially supervised classification". Their proposed technique combines Naive Bayes classification and the Expectation–Maximization algorithm to iteratively produce classifiers. The training scores of these classifiers are then directly used to rank $\mathcal{U}$. Following this approach, a straightforward solution to the transductive PU learning problem is to train any classifier to discriminate between $\mathcal{P}$ and $\mathcal{U}$ and to use this classifier to assign a score to the unlabeled data that were used to train it. Using SVMs this amounts to using the biased SVM training scores. We will subsequently denote this approach by transductive biased SVM.

However, one may argue that assigning a score to an unlabeled example that has been used as negative training example is problematic. In particular, if the classifier fits too tightly to the training data, a false negative x_i will hardly be given a high training score when used as a negative. In a related situation in the context of semi-supervised learning, Zhang et al. (2009) showed for example that unlabeled examples used as negative training examples tend to have underestimated scores when an SVM is trained with the classical hinge loss. More generally, most theoretical consistency properties of machine learning algorithms justify predictions on samples outside of the training set, raising questions on the use of all unlabeled samples as negative training samples at the same time.

Alternatively, the inductive bagging PU learning lends itself particularly well to the transductive setting, through the procedure described in Algorithm 2. Each time a random subsample $\mathcal{U}_t$ of $\mathcal{U}$ is generated, a classifier is trained to discriminate $\mathcal{P}$ from $\mathcal{U}_t$, and used to assign a predictive score to any element of $\mathcal{U} \setminus \mathcal{U}_t$. At the end the score of any element $x \in \mathcal{U}$ is obtained by aggregating the predictions of the classifiers trained on subsamples that did not contain x (the counter $n(x)$ simply counts the number of such classifiers). As such, no point of $\mathcal{U}$ is used simultaneously to train a classifier and to test it. In practice, it is useful to ensure that we average the predictions over a sufficient number of classifiers. Typically, if we wish to average over n scores, we need to choose T such that $T(1 - \frac{K}{|\mathcal{U}|}) \gg n$.

Algorithm 2. Transductive bagging PU learning

```
INPUT: $\mathcal{P}, \mathcal{U}, K$ = size of bootstrap samples, $T$ = number of
  bootstraps
OUTPUT: a score $s : \mathcal{U} \rightarrow \mathbb{R}$
Initialize $\forall x \in \mathcal{U},\ n(x) \leftarrow 0,\ f(x) \leftarrow 0$
for $t = 1$ to $T$ do
  Draw a bootstrap sample $\mathcal{U}_t$ of size $K$ in $\mathcal{U}$.
  Train a classifier $f_t$ to discriminate $\mathcal{P}$ against $\mathcal{U}_t$.
  For any $x \in \mathcal{U} \setminus \mathcal{U}_t$, update:

    $f(x) \leftarrow f(x) + f_t(x),$

    $n(x) \leftarrow n(x) + 1.$

end for
Return $s(x) = f(x)/n(x)$ for $x \in \mathcal{U}$
```

3. Results

In this section we investigate the empirical behavior of our bagging algorithm on one simulated dataset (Section 3.1) and two real applications: text retrieval with the 20 newsgroup benchmark (Section 3.2), and reconstruction of gene regulatory networks (Section 3.3). We compare the new bagging SVM to the state-of-the-art biased SVM, and also add in the comparison for real data two one-class approaches, namely, ranking unlabeled examples by decreasing mean similarity to the positive examples (called *Baseline* below), and the one-class SVM (Schölkopf et al., 2001). The biased SVM consists in training a soft margin SVM to discriminate between $\mathcal{P}$ and $\mathcal{U}$. Both bagging and biased methods involve an SVM with asymmetric penalties C_+ and C_- for the positive and negative class, respectively. By default we always set them to ensure that the total penalty is equal for the two classes, i.e., $C_+n_+ = C_-n_-$, where n_+ and n_- are the number of positive and negative examples fed to the SVM, and optimized the single parameter $C = C_+ + C_-$ over a grid. We checked on all experiments that this choice was never significantly outperformed by another penalty ratio C_+/C_-.

All methods were implemented in MATLAB, using the LIBSVM software (Chang and Lin, 2011) to train one- and two-class SVM. All experiments were run under Linux on a machine with two 4-core Intel Xeon 3.16 GHz processors and 16 Gb or RAM.

3.1. Simulated data

A first series of experiments were conducted on simulated data to compare our bagging procedure to the biased approach in an inductive setting. We consider the simple situation where the positive examples are generated from an isotropic Gaussian distribution in $\mathbb{R}^p : \mathcal{P} \sim \mathbb{P}_+ = \mathcal{N}(0_p, \sigma * I_p)$, with $p = 50$ and $\sigma = 0.6$, while the negative examples are generated from another Gaussian distribution with same isotropic covariance and a different mean, of norm 1. We replicate the following iteration 50 times for different values of γ:

- Draw a sample $\mathcal{P}$ of 5 positives examples, and a sample $\mathcal{U}$ of 50 unlabeled examples from $\gamma * \mathbb{P}_+ + (1 - \gamma) * \mathbb{P}_-$.

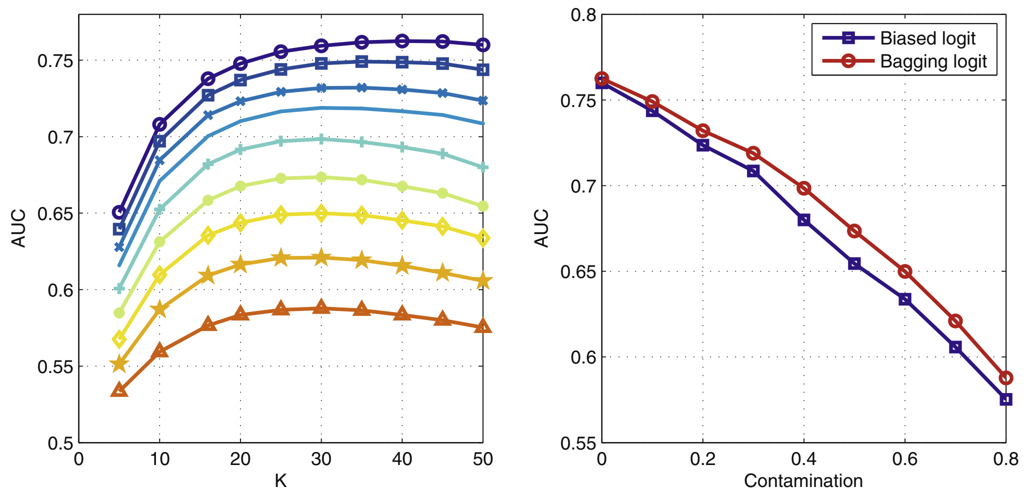

Fig. 1. Results on simulated data. Left: AUC of the bagging logit as a function of K, the size of the bootstrap samples, on simulated data. Each curve, from top to bottom, corresponds to a contamination level $\gamma \in \{0; 0.1; 0.2; \ldots; 0.8\}$. Right Performance of two methods as a function of γ, the contamination level, on simulated data. The performance of bagging logit was taken at the optimal K value.

- Train respectively the biased and bagging logit (with 200 bootstraps).[1]
- Compare their performance on a test set of 1000 examples containing 50% positives.

For K, we tested equally spaced values between 1 and 50, and we varied γ on the interval $[0; 0.8]$. The performance is measured by computing the area under the Receiving Operator Characteristic curve (AUC) on the independent test set. Fig. 1 (left) shows the performance of bagging logit for different levels of contamination of $\mathcal{U}$, as a function of K, the size of the random samples. The uppermost curve thus corresponds to $\gamma = 0$, i.e., the case where all unlabeled data are negative, while the bottom curve corresponds to $\gamma = 0.8$, i.e., the case where 80% of unlabeled data are positive. Note that $K = 50$ corresponds to classical bagging on the biased logit classifier, i.e., to the case where all unlabeled examples are used to train the classifier.

We observe that in the classical setting of supervised binary classification where $\mathcal{U}$ is not contaminated by positive samples ($\gamma = 0$), the bagging procedure does not improve performance, whatever the size of the bootstrap samples. On the other hand, as contamination increases, we observe an overall decrease of the performance, confirming that the classification problem becomes more difficult when contamination increases. In addition, the bagging logit always succeeds in reaching at least the same performance for a value of K below 50, even for high rates of contamination. Fig. 1 (right) shows the evolution of AUC as γ increases, for both methods. For the bagging logit we report the AUC reached for the best K value. We see that bagging logit slightly outperforms the biased logit method.

To further illustrate the assumption that motivated bagging SVM, namely that decreasing K would decrease the average performance of single classifiers but would increase their variance due to the variations in contamination, we take a closer look at the successive classifiers learnt when training Algorithm 1. Each classifier corresponds to a random bootstrap subsample $\mathcal{U}_t$. We show in Fig. 2 a scatter plot of the AUC of these individual classifiers as a function of $\hat{\gamma}$, the empirical contamination of the bootstrap sample $\mathcal{U}_t$, for two values of K (10 and 40). Here the mean contamination was set to $\gamma = 0.2$. Obviously, the variations of $\hat{\gamma}$ are much larger for $K = 10$ (between 0 and 0.5) than for $K = 40$ (between 0.1 and 0.25). The correlation coefficient between $\hat{\gamma}$ and the performance (reported above each plot) is strongly negative, in particular for smaller K. It is quite clear that less contaminated subsamples tend to yield better classifiers, and that the variation in the contamination is an important factor to increase the variance between individual predictors, which aggregation can benefit from.

3.2. Newsgroup dataset

The 20 Newsgroup benchmark is widely used to test PU learning methods. The version we used is a collection of 11,293 articles partitioned into 20 subsets of roughly the same size (around 500),[2] corresponding to post articles of related interest. For each newsgroup, the positive class consists of those ~500 articles known to be relevant, while the negative class is made of the remainder. After pre-processing, each article is represented by a 8165-dimensional feature vector, based on word counts, using the TFIDF representation over a dictionary of 8165 words (Joachims, 1997).

To simulate a PU learning problem, we applied the following strategy. For a given newsgroup, we created a set $\mathcal{P}$ of known positive examples by randomly selecting a given number of positive examples, while $\mathcal{U}$ contains the non-selected positive examples and all negative examples. We varied the size N_P of $\mathcal{P}$ in $\{5, 10, 20, 50, 100, 200, 300\}$ to investigate the influence of the number of known positive examples. For each newsgroup and each value of N_P, we train all 4 methods described above (bagging SVM, biased SVM, baseline, one-class SVM) and rank the samples in $\mathcal{U}$ by decreasing score (transductive setting). We then compute the AUC, and average this measure over 10 replicates of each newsgroup and each value of N_P. For bagging and biased SVM, we varied the C parameter over the grid $\{e^{-12}, e^{-10}, \ldots, e^{2}\}$, while we vary parameter ν in $\{0.1, 0.2, \ldots, 0.9\}$ for 1-class SVM. We only used the linear kernel.

We first investigated the influence of T. Fig. 3 shows, for the first newsgroup (alt.atheism), the performance reached as a function of T, for different settings in N_P and K. As expected we observe that in general the performance increases with T, but quickly reaches a plateau beyond which additional bootstraps do not improve

[1] The bagging logit corresponds to the procedure described above, when the classifier is a logistic regression. This is the same for the biased logit, see also Lee and Liu (2003).

[2] We used the Matlab pre-processed version available at http://renatocorrea.googlepages.com/ng2011293x8165itrn.mat

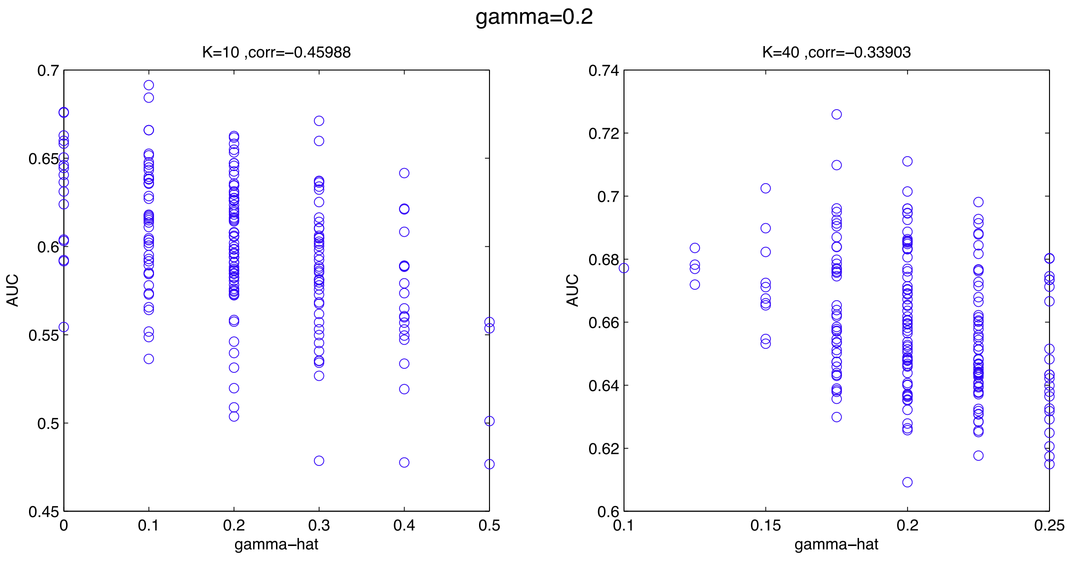

Fig. 2. Distribution of AUC and $\hat{\gamma}$ over the 500 iterations of one bootstrap loop on the simulated dataset, $\gamma = 0.2$.

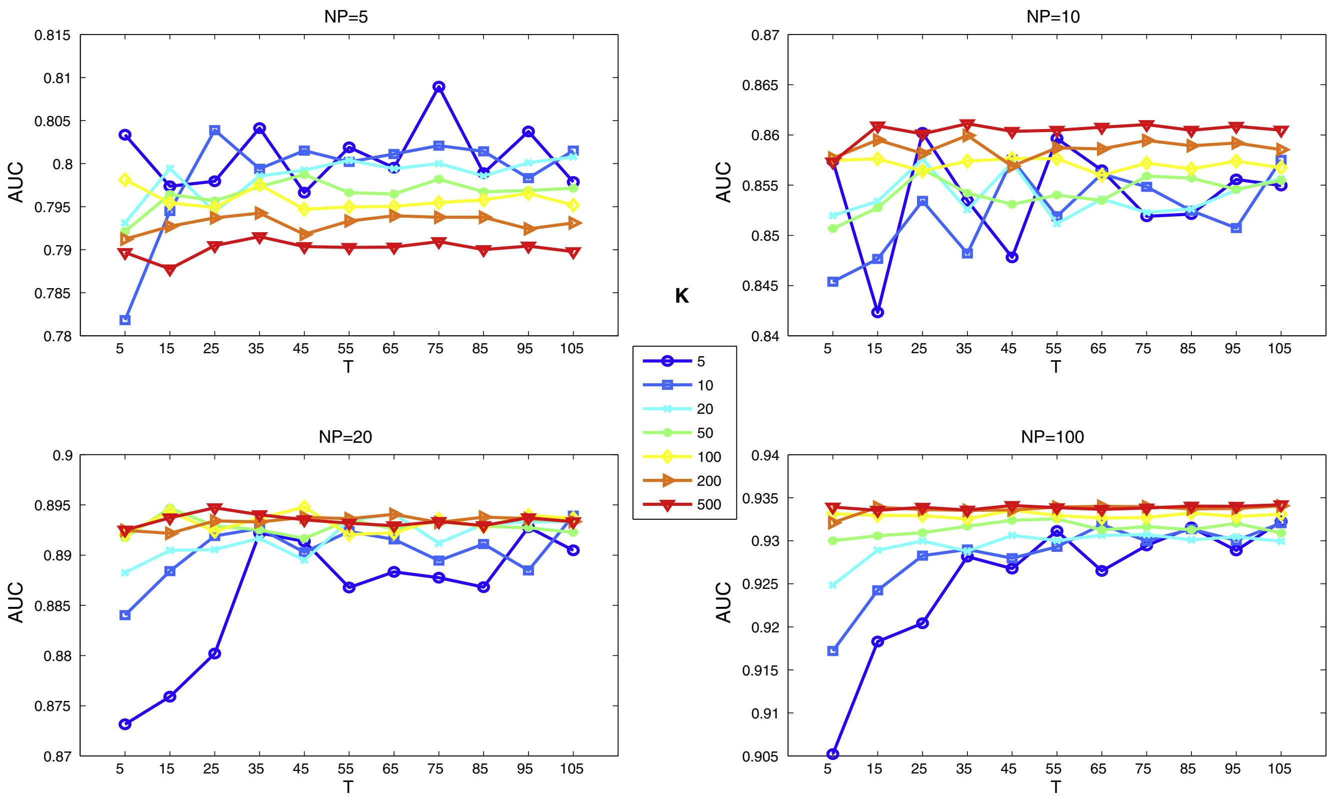

Fig. 3. Performance on newsgroup 1 (alt.atheism) as a function of the number of bootstraps T, for different values of N_P and K.

performance. Overall the smaller K, the larger T must be to reach the plateau. From these preliminary results we set $T = 35$ for $K \leqslant 20$, and $T = 10$ for $K > 30$, and kept it fix for the rest of the experiments. To further clarify the benefits of bagging, we show in Fig. 4 the performance of the bagging SVM versus the performance of a SVM trained on a single bootstrap sample ($T = 1$), for different values of K and a fixed number of positives $N_P = 10$. We observe that, for K below 200, aggregating classifiers over several bootstrap subsamples is clearly beneficial, while for larger values of K it does not really help. This is coherent with the observation that SVM usually rarely benefit from bagging: here the benefits come from our particular bagging scheme. Interestingly, we see that very good performance is reached even for small values of K with the bagging.

Fig. 5 shows the mean AUC averaged over the 10 folds and the 20 newsgroups for bagging SVM as a function of K, and compares it to that of the biased SVM. More precisely, each point on the curve corresponds to the performance averaged over the 20 Newsgroups after choosing a posteriori the best C parameter for each newsgroup. This is equivalent to comparing optimal cases for both methods. Contrary to what we observed on simulated data, we observe that K has in general very little influence on the performance.

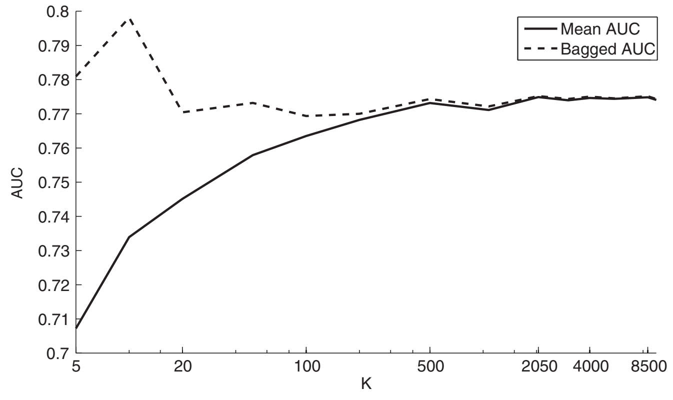

Fig. 4. Performance on one newgroup of bagging SVM (*bagged AUC*) vs a SVM trained on a single bootstrap sample (*mean AUC*), for different values of *K*.

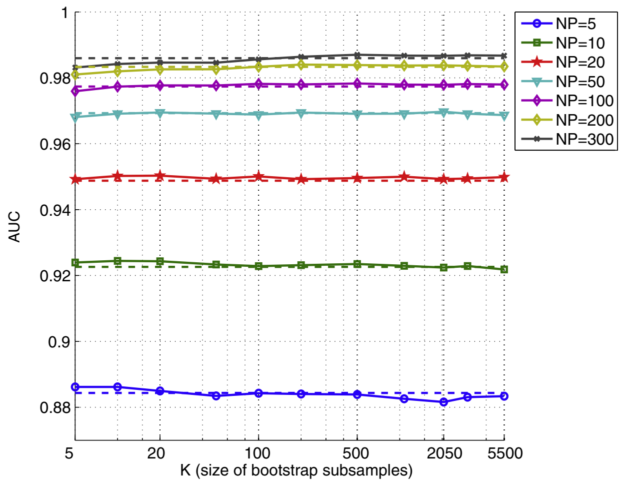

Fig. 5. Macro averaged performance of the bagging SVM as a function of *K*. The dashed horizontal lines show the AUC level of the biased SVM. The curves are plotted for different values of N_P, the size of the positive set.

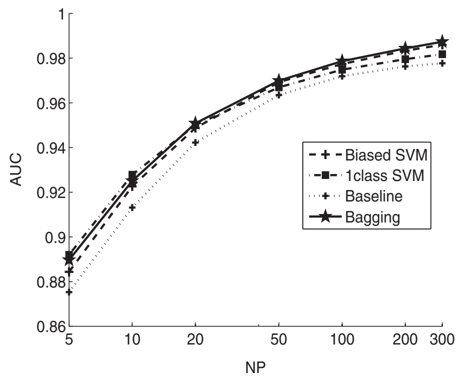

Fig. 6. Performance of the baseline method, the 1-class SVM, the biased SVM and the newly proposed bagging SVM methods on the 20 Newsgroups dataset. Each curve shows how the mean AUC varies with the number of positive training examples N_P. For each value of N_P, the performance of bagging SVM is computed at the optimal value for *K*, as shown in Fig. 5.

The AUC of the bagging SVM is similar to that of the biased SVM for most values of *K*, although for N_P larger than 50, a slight advantage can be observed for the biased SVM over bagging SVM when *K* is too small. We conclude that in practice, parameter *K* may not need to be finely tuned since it does not always have a big impact on the performance. In all cases, $K = N_P$ seems to be a safe choice for the bagging SVM.

Finally, Fig. 6 shows the average AUC over the 20 newsgroups for all four methods, as a function of N_P. Overall all methods are very similar, with the Baseline slightly below the others. In details, the bagging SVM curve dominates all other methods for $N_P \geqslant 20$, while the 1-class SVM is the one which dominates for smaller values of N_P. Although the differences in performance are small, the bagging SVM outperforms the biased SVM significantly for $N_P > 20$ according to a Wilcoxon paired sample test (at 5% confidence). For small values of N_P however, no significant difference can be proven in either way between bagging SVM and 1-class SVM, which remains a very competitive method.

3.3. *Escherichia coli dataset: inference of transcriptional regulatory network*

In this section we test the different PU learning strategies on the problem of inferring the transcription regulatory network of the bacteria *E. coli* from gene expression data. The problem is, given a transcription factor (TF), to predict which genes it regulates. Following Mordelet and Vert (2008), we can formulate this problem as transductive PU learning by starting from known regulated genes (considered positive examples), and looking for additional regulated genes in the bacteria's genome.

To represent the genes, we use a compendium of microarray expression profiles provided by Faith et al. (2008), in which 4345 genes of the *E. coli* genome are represented by vectors in 445 dimensions, corresponding to their expression level in 445 different experiments. We extracted the list of known regulated genes for each TF from the RegulonDB (Salgado et al., 2006).

For each TF, we ran a double 3-fold cross validation with an internal loop on each training set to select parameter *C* of the SVM (or ν for the 1-class SVM). To make this possible, we restrict ourselves to 31 TFs with at least 8 known regulated genes. Following Mordelet and Vert (2008), we normalize the expression data to unit norm, use a Gaussian RBF kernel with $\sigma = 8$, and perform a particular cross-validation scheme to ensure that operons are not split between folds. Finally, following our previous results on simulated data and the newsgroup benchmark, we test two variants of bagging SVM, setting *K* successively to N_P and $5 * N_P$. These choices are denoted respectively by *bagging1 SVM* and *bagging5 SVM*.

Fig. 7 shows the average precision/recall curves of all methods tested. Overall we observe that all three PU learning methods give significantly better results than the two methods which use only positive examples (Wilcoxon paired sample test at 5% significance level). No significant difference was found between the three PU learning methods. This confirms again that for different values of *K* bagging SVM matches the performance of biased SVM.

4. Discussion

The main contribution of this work is to propose a new method, bagging SVM, both for inductive and transductive PU learning, and to assess in detail its performance and the influence of various parameters on simulated and real data.

The motivation behind bagging SVM was to exploit an intrinsic feature of PU learning to benefit from classifier aggregation through a random subsample strategy. Indeed, by randomly sampling *K* examples from the unlabeled examples, we can expect various contamination rates, which in turn can lead to very different single classifiers (good ones when there is little contamination, worse ones when contamination is high). Aggregating these classifiers can in turn benefit from the variations between them. This

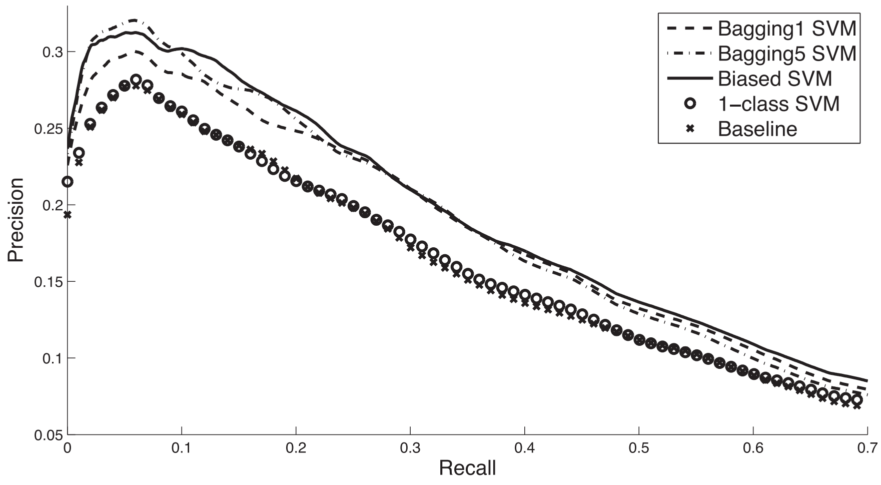

Fig. 7. Precision-recall curves to compare the performance between the baggin1 SVM, the bagging5 SVM, the biased SVM, the 1-class SVM and the baseline method.

Table 1
CPU time (in s) and performance measures (AUC: area under the ROC curve and AUP: area under the precision/recall curve) for different settings of T and K for bagging SVM.

Bagging		CPU			AUC–AUP		
		$K = 10$	$K = 50$	$K = 200$	$K = 10$	$K = 50$	$K = 200$
T	35	13	39	91	0.921–0.531	0.917–0.524	0.902–0.518
	50	18	54	127	0.920–0.539	0.914–0.522	0.904–0.522
	200	72	170	473	0.918–0.539	0.910–0.528	0.904–0.511

suggests that K may play an important role in the final performance of bagging SVM, since it controls the trade-off between the mean and variance of individual classifiers. While we showed on simulated data that this is indeed the case, and that there can be some optimum K to reach the best final accuracy, the two experiments on real data did not show any strong influence of K and suggested that $K = N_P$ may be a safe default choice. This is a good news since it does not increase the number of parameters to optimize for the bagging SVM and leads to balanced training sets that most classification algorithms can easily handle. Regarding parameter C optimization, our experiments on the Newsgroup dataset were designed so as to compare optimal cases and therefore did not include any parameter selection strategy. Hence they were rather intended as a proof of concept, to show that if one is able to successfully select optimal parameters C, then one would be able to reach same performance with the bagging SVM scheme as with the biased SVM method. However note that in practice, parameter optimization is a crucial step which may be carried out using cross validation, as was done on the *E. coli* dataset.

The comparison between different methods is mitigated. While bagging SVM outperforms biased SVM on simulated data, they are not significantly different on the two experiments with real data. Interestingly, while these PU learning methods were significantly better than two methods that learned from positive examples only on the gene regulatory network example, the 1-class SVM behaved very well on the 20 newsgroup benchmark, even outperforming the PU learning methods when less than 10 training examples were provided. Many previous works, including Liu et al. (2003) and Yu et al. (2004) discard 1-class SVMs for showing a bad performance in terms of accuracy, while Manevitz and Yousef (2001) report the lack of robustness of this method arguing that it has proved very sensitive to changes of parameters. Our results suggest that there are cases where it remains very competitive, and that PU learning may not always be a better strategy than simply learning from positives.

Finally, the main advantage of bagging SVM over biased SVM is that it greatly alleviates the computation burden, in particular when there are far more unlabeled than positive examples. Indeed, a typical algorithm, such as an SVM, trained on N samples, has time complexity proportional to N^{β}, with β between 2 and 3. Therefore, biased SVM has complexity proportional to $(P + U)^{\beta}$ while bagging SVM's complexity is proportional $T * (P + K)^{\beta}$. With the default choice $K = P$ ratio of CPU time to train the biased SVM vs the bagging SVM can therefore be expected to be $((P + U)/(2P))^{\beta}/T$. Then we conclude that bagging SVM should be faster than biased SVM as soon as $U/P > 2T^{1/\beta} - 1$. For example, taking $T = 35$ and $\beta = 3$, bagging SVM should be faster than biased SVM as soon as $U/P > 6$, a situation very often encountered in practice where the ratio U/P is more likely to be several orders of magnitude larger. In the two real datasets, this was always the case. Table 1 reports CPU time in seconds and performance measure for training bagging SVM on the first fold of newsgroup 1 with C fixed at its best value a posteriori and $N_P = 10$.

In comparison, the biased SVM's CPU time is 227 s for $\text{AUC} = 0.932$ and $\text{AUP} = 0.491$. This confirms that for reasonable values of T and K, the bagging SVM is much faster than the biased SVM for a comparable performance.

5. Conclusion

We have presented an original approach to the problem of learning from positive and unlabeled examples. Our approach uses a bagging-like strategy to exploit the availability of the numerous unlabeled examples. Extensive experiments on simulated and real data have allowed us to assess the sensitivity of the algorithm to its parameter and to compare its performance to existing methods. We have provided safe choices of the parameters, thus reducing the number of parameters to optimize and making our algorithm simple to implement and to apply. We have shown that our bagging SVM outperforms existing approaches on simulated data. On real data, the results were more mitigated, but the bagging SVM remained competitive being either the dominant method or performing equally (no significance difference was found). Finally, our method greatly improves over the state-of-art biased SVM in terms of computation time.

References

Aerts, S., Lambrechts, D., Maity, S., Van Loo, P., Coessens, B., De Smet, F., Tranchevent, L.-C., De Moor, B., Marynen, P., Hassan, B., Carmeliet, P., Moreau, Y., 2006. Gene prioritization through genomic data fusion. Nat. Biotechnol. 24 (5), 537–544.

Bartlett, P.L., Tewari, A., 2007. Sparseness vs estimating conditional probabilities: some asymptotic results. J. Mach. Learn. Res. 8, 775–790.

Breiman, L., 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32.

Chang, C.-C., Lin, C.-J., 2011. LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. 2 (3), 27:1–27:27.

Chapelle, O., Schölkopf, B., Zien, A., 2006. Semi-Supervised Learning. MIT Press, Cambridge, MA.

De Bie, T., Tranchevent, L.-C., van Oeffelen, L.M.M., Moreau, Y., 2007. Kernel-based data fusion for gene prioritization. Bioinformatics 23 (13), i125–i132.

Denis, F., Gilleron, R., Letouzey, F., 2005. Learning from positive and unlabeled examples. Theor. Comput. Sci. 348 (1), 70–83.

Elkan, C., Noto, K., 2008. Learning classifiers from only positive and unlabeled data. In: KDD '08: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM, New York, NY, USA, pp. 213–220.

Faith, J., Driscoll, M., Fusaro, V., Cosgrove, E., Hayete, B., Juhn, F., Schneider, S., Gardner, T., 2008. Many microbe microarrays database: uniformly normalized affymetrix compendia with structured experimental metadata. Nucleic Acids Res. 36 (Database issue), D866–D870.

Geurts, P., 2011. Learning from positive and unlabeled examples by enforcing statistical significance. J. Mach. Learn. Res. – Proc. Track 15, 305–314.

Hastie, T., Tibshirani, R., Friedman, J., 2001. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer.

Joachims, T., 1997. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization. In: Fisher, D.H. (Ed.), ICML '97: Proceedings of the 14th International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., Nashville, TN, USA, pp. 143–151.

Joachims, T., 1999. Transductive inference for text classification using support vector machines. In: Fürnkranz, J., Kubat, M. (Eds.), ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 200–209.

Lee, W.S., Liu, B., 2003. Learning with positive and unlabeled examples using weighted logistic regression. In: Fawcett, T., Mishra, N. (Eds.), ICML 2003: Proceedings of the 20th International Conference on Machine Learning. AAAI Press, pp. 448–455.

Li, X., Liu, B., 2003. Learning to classify texts using positive and unlabeled data. In: Gottlob, G., Walsh, T. (Eds.), IJCAI'03: Proceedings of the 18th International Joint Conference on Artificial Intelligence. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 587–592.

Liu, B., Lee, W.S., Yu, P.S., Li, X., 2002. Partially supervised classification of text documents. In: Sammut, C., Hoffmann, A.G. (Eds.), ICML'02: Proceedings of the 19th International Conference on Machine Learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 387–394.

Liu, B., Dai, Y., Li, X., Lee, W.S., Yu, P.S., 2003. Building text classifiers using positive and unlabeled examples. In: Wu, X., Tuzhilin, A., Shavlik, J. (Eds.), ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining. IEEE Computer Society, Washington, DC, USA, pp. 179–186.

Manevitz, L.M., Yousef, M., 2001. One-class SVMs for document classification. J. Mach. Learn. Res. 2, 139–154.

Mordelet, F., Vert, J.-P., 2008. SIRENE: supervised inference of regulatory networks. Bioinformatics 24 (16), i76–i82.

Pelckmans, K., Suykens, J., 2009. Transductively learning from positive examples only. In: ESANN 2009: Proceedings of the 17th European Symposium on Artificial Neural Networks.

Salgado, H., Gama-Castro, S., Peralta-Gil, M., Díaz-Peredo, E., Sánchez-Solano, F., Santos-Zavaleta, A., Martínez-Flores, I., Jiménez-Jacinto, V., Bonavides-Martínez, C., Segura-Salazar, J., Martínez-Antonio, A., Collado-Vides, J., 2006. RegulonDB (version 5.0): *Escherichia coli* K-12 transcriptional regulatory network, operon organization, and growth conditions. Nucleic Acids Res. 34 (Database issue), D394–D397.

Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C., 2001. Estimating the support of a high-dimensional distributions. Neural Comput. 13, 1443–1471.

Scott, C., Blanchard, G., 2009. Novelty detection: unlabeled data definitely help. In: van Dyk, D., Welling, M. (Eds.), AISTATS '09 Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, JMLR: W&CP 5, vol. 5. Clearwater Beach, FL, pp. 464–471.

Scott, C., Nowak, R., 2005. A Neyman–Pearson approach to statistical learning. IEEE Trans. Inf. Theory 51 (11), 3806–3819.

Shah, A.R., Oehmen, C.S., Webb-Robertson, B.-J., 2008. SVM-HUSTLE – an iterative semi-supervised machine learning approach for pairwise protein remote homology detection. Bioinformatics 24 (6), 783–790.

Sriphaew, K., Takamura, H., Okumura, M., 2009. Cool blog classification from positive and unlabeled examples. In: Theeramunkong, T., Kijsirikul, B., Cercone, C., Ho, T.-B. (Eds.), PAKDD 2009: Proceedings of the 13th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining. Springer-Verlag, Berlin, Heidelberg, pp. 62–73.

Steinwart, I., 2003. Sparseness of support vector machines. J. Mach. Learn. Res. 4, 1071–1105.

Vert, R., Vert, J.-P., 2006. Consistency and convergence rates of one-class SVMs and related algorithms. J. Mach. Learn. Res. 7, 817–854.

Yu, H., Han, J., Chang, K.C.-C., 2004. PEBL: web page classification without negative examples. IEEE Trans. Knowl. Data Eng. 16 (1), 70–81.

Zhang, K., Tsang, I., Kwok, J., 2009. Maximum margin clustering made practical. IEEE Trans. Neural Networks 20 (4), 583–596.