Todo list

we still have to write the dutch version! v

Chapter 2

A Robust Ensemble Approach to Learn From Positive and Unlabeled Data Using SVM Base Models

{ch:resvm}

This chapter has been previously published as:

Claesen, M., De Smet, F., Suykens, J. A., & De Moor, B. (2015). **A robust ensemble approach to learn from positive and unlabeled data using SVM base models**. *Neurocomputing*, 160, 73-84.

Abstract

We present a novel approach to learn binary classifiers when only positive and unlabeled instances are available (PU learning). This problem is routinely cast as a supervised task with label noise in the negative set. We use an ensemble of SVM models trained on bootstrap resamples of the training data for increased robustness against label noise. The approach can be considered in a bagging framework which provides an intuitive explanation for its mechanics in a semi-supervised setting. We compared our method to state-of-the-art approaches in simulations using multiple public benchmark data sets. The included benchmark comprises three settings with increasing label noise: (i) fully supervised, (ii) PU learning and (iii) PU learning with false positives. Our approach shows a marginal improvement over existing methods in the second setting and a significant improvement in the third.

2.1 Introduction

Training binary classifiers on positive and unlabeled data is referred to as PU learning [59]. The absence of known negative training instances warrants appropriate learning methods. Inaccurate label information can be more problematic than attribute noise [84]. Specialised PU learning approaches are recommended when (i) negative labels cannot be acquired, (ii) the training data contains a large amount of false negatives or (iii) the positive set has many outliers.

Practical applications of PU learning typically feature large, imbalanced training sets with a small amount of labeled (positive) and a large amount of unlabeled training instances. The PU learning problem arises in various settings, including web page classification [83], intrusion detection [52] and bioinformatics tasks such as variant prioritization [75], gene prioritization [1, 65] and virtual screening of drug compounds [74].

Though these applications share a common underlying learning problem, the final evaluation criteria may be fundamentally different. For instance, in prioritization one wishes to obtain high precision since highly ranked targets may be subjected to further biological analysis. Intrusion detection, on the other hand, necessitates high recall to ensure that no anomalies go unnoticed.

Following **(author?)** [64], we will use the term *contamination* to refer to the fraction of mislabeled instances in a given set. We will denote the positive and unlabeled training instances by $\mathcal{P}$ and $\mathcal{U}$, respectively. Contamination in $\mathcal{P}$ refers to false positives while contamination in $\mathcal{U}$ refers to the presence of positives in $\mathcal{U}$. Usually $\mathcal{U}$ contains mostly true negative instances (e.g. contamination below 0.5) and $\mathcal{P}$ is assumed to be uncontaminated.

The distributions of the positive and a contaminated unlabeled set overlap even when those of the positive and underlying negative sets do not, which makes classification more difficult compared to a traditional supervised setting. **(author?)** [36] and **(author?)** [13] report statistical approaches to estimate the contamination of the unlabeled set and additionally show that distinguishing positives from unlabeled instances is a valid proxy for distinguishing positives from negatives.

The assumption in PU learning that $\mathcal{P}$ is uncontaminated may be violated in applications due to various reasons [40]. Additionally, outliers in the positive set may have a similar effect on classification performance [69]. We propose a novel PU learning method that is less vulnerable to potential contamination in $\mathcal{P}$ called the robust ensemble of support vector machines (RESVM). RESVM is compared to other methods in a series of simulations based on several public

data sets.

2.2 Related work

PU learning approaches can be split into two main conceptual categories: (i) approaches that account for the contamination of the unlabeled set explicitly by modeling the label noise and (ii) approaches that try to infer an uncontaminated (negative) subset $\hat{\mathcal{N}}$ from $\mathcal{U}$ and then train supervised algorithms to distinguish $\mathcal{P}$ from $\hat{\mathcal{N}}$. When *very* few labeled examples are available, the structure within the data is the main source of information which can be exploited by semi-supervised clustering techniques [2].

Accounting for the contamination of $\mathcal{U}$ in the modeling process This can be done by weighting individual data points, such as in weighted logistic regression [36, 54]. Another approach is by changing the penalties on misclassification during training, as is done in class-weighted SVM [59], bagging SVM [64] and RT-SVM [61].

Inferring an uncontaminated subset from $\mathcal{U}$ Another class of approaches tries to infer a negative set $\hat{\mathcal{N}}$ from $\mathcal{U}$. After the inferential step, binary classifiers are trained to distinguish $\mathcal{P}$ from $\hat{\mathcal{N}}$ in a supervised fashion. Examples of such two-step approaches include S-EM [60], mapping convergence (MC) [82] and ROC-SVM [55].

Class-weighted SVM and related approaches The approach we suggest belongs to the first class of methods and is closely related to class-weighted SVM and bagging SVM (which uses class-weighted SVM internally). We will discuss both of these approaches in more detail before moving on to the proposed method. We evaluated our method compared to both class-weighted SVM and bagging SVM.

2.2.1 Class-weighted SVM

{bsvm}

Class-weighted SVM (CWSVM) is a supervised technique in which the penalty for misclassification differs per class. **(author?)** [59] first applied class-weighted SVM for PU learning by considering the unlabeled set to be negative with noise on its labels. CWSVM is trained to distinguish $\mathcal{P}$ from $\mathcal{U}$. During training,

misclassification of positive instances is penalized more than misclassification of unlabeled instances to emphasize the higher degree of certainty on positive labels. In the context of PU learning, the optimization problem for training CWSVM can be written as:

$$\min_{\alpha,\xi,b} \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j y_i y_j \kappa(\mathbf{x}_i,\mathbf{x}_j) + C_{\mathcal{P}}\sum_{i\in\mathcal{P}}\xi_i + C_{\mathcal{U}}\sum_{i\in\mathcal{U}}\xi_i, \tag{2.1}$$ {eq:bsvm}

$$\text{s.t. } y_i(\sum_{j=1}^{N}\alpha_j y_j \kappa(\mathbf{x}_i,\mathbf{x}_j) + b) \geq 1-\xi_i, \qquad i=1,\ldots,N,$$

$$\xi_i \geq 0, \qquad i=1,\ldots,N,$$

with $\alpha \in \mathbb{R}^N$ the support values, $\mathbf{y} \in \{-1,+1\}^N$ the label vector, $\kappa(\cdot,\cdot)$ the kernel function, b the bias term and $\xi \in \mathbb{R}^N$ the slack variables. The misclassification penalties $C_{\mathcal{P}}$ and $C_{\mathcal{U}}$ require tuning (typically $C_{\mathcal{P}} > C_{\mathcal{U}}$ to emphasize known labels). SVM formulations with unequal penalties across classes have been used previously to tackle imbalanced data sets [68].

2.2.2 Bagging SVM

{baggingsvm}

Mordelet and Vert introduce bagging SVM as a meta-algorithm which consists of aggregating classifiers trained to discriminate $\mathcal{P}$ from a small, random resample of $\mathcal{U}$ [64]. They posit that PU learning problems have a particular structure that leads to instability of classifiers, namely the sensitivity of classifiers to the contamination of the unlabeled set. Bagging is a common technique used to improve the performance of instable classifiers [16].

In bagging SVM, random resamples of $\mathcal{U}$ are drawn and CWSVM classifiers are trained to discriminate $\mathcal{P}$ from each resample. By resampling $\mathcal{U}$, the contamination is varied. This induces variability in the classifiers which the aggregation procedure can then exploit. The size of the bootstrap resample of $\mathcal{U}$ is a tuning parameter in bagging SVM. The ratio $C_{\mathcal{P}}/C_{\mathcal{U}}$ is fixed so that the following holds:

$$|\mathcal{P}| \times C_{\mathcal{P}} = n_{\mathcal{U}} \times C_{\mathcal{U}}, \tag{2.2}$$ {eq:bagpenalties}

with $|\mathcal{P}|$ the size of the positive set and $n_{\mathcal{U}}$ the size of resamples from the unlabeled set. This choice of weights is common in imbalanced settings [20, 28]. All base models in bagging SVM classify the full set of positives against a subset of unlabeled instances and use a high misclassification penalty on the positives similar to CWSVM.

2.3 Robust Ensemble of SVMs

We propose a new technique called the robust ensemble of SVMs (RESVM). RESVM is a bagging method using CWSVM base models as discussed in Section 2.2.1. Base model training sets are constructed by bootstrap resampling both $\mathcal{P}$ and $\mathcal{U}$ separately, both of which may be contaminated.

The key difference between RESVM and bagging SVM is that the former resamples $\mathcal{P}$ in addition to $\mathcal{U}$ to increase variability between base models. RESVM additionally features an extra degree of freedom to control the relative misclassification penalty between positive and unlabeled instances, which is fixed in bagging SVM. **(author?)** [64] report no significant changes when varying the relative penalty in bagging SVM, though our experiments show that it is important in RESVM (see w_{pos} in Table 2.6).

Before elaborating on the details of RESVM, we briefly illustrate the effect of resampling contaminated sets. Subsequently we summarize the mechanisms of bagging and why they are advantageous when learning with label noise in the RESVM approach. Finally, we provide the full RESVM training approach and the way ensemble decision values are computed based on the base model decision values.

2.3.1 Bootstrap resampling contaminated sets

{resampling}

The RESVM approach resamples both $\mathcal{P}$ and $\mathcal{U}$, both of which are potentially contaminated. Resampling contaminated sets with replacement induces variability in contamination across the resampled sets (e.g. resamples of $\mathcal{U}$ and $\mathcal{P}$ that are used for training). The variability in contamination between resamples increases for increasing contamination of the original set. We assume contamination levels below 50%, e.g. less than half the instances in a given set are mislabeled. Due to the law of large numbers the contamination in bootstrap resamples of increasing size converges to the expected contamination, which equals that of the original set that is being resampled. As a result, the variability in contamination decreases for increasing resample size. Figure 2.1 illustrates this property empirically based on $20,000$ repeated measurements for each resample size: the expected value (mean) equals the original contamination, but the variability in resample contamination decreases for increasing resample size.

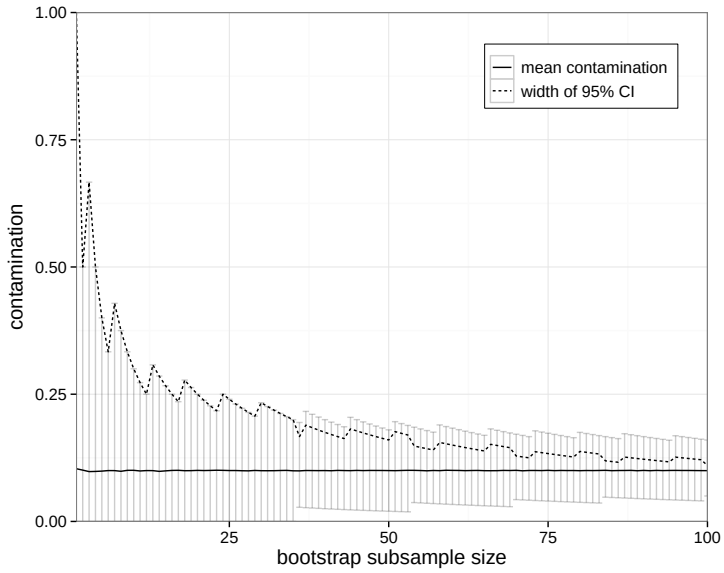

Figure 2.1: Contamination of bootstrap resamples for increasing size of resamples when the original sample has 10% contamination. Errorbars indicate the 95% confidence interval (CI) of contamination in resamples. The contamination varies greatly between small resamples as shown by the CIs.

{fig:contamination}

2.3.2 Bagging predictors

(author?) [16] introduced bagging as a technique to construct strong ensembles by combining a set of base models. **(author?)** [17] stated that "the essential problem in combining classifiers is growing a suitably diverse ensemble of base classifiers" which can be done in various ways [19]. In bagging, the ensemble models use majority voting to aggregate decisions of base models which are trained on bootstrap resamples of the training set. From a Bayesian point of view, bagging can be interpreted as a Monte Carlo integration over an approximated posterior distribution [72].

In his landmark paper, **(author?)** [16] noted that base model instability is an important factor in the success of bagging which led to the use of inherently instable methods like decision trees in early bagging approaches [32, 18]. The main mechanism of bagging is often said to be variance reduction [5, 17]. In more recent work, **(author?)** [42] explained that base model instability is not related to the intrinsic variability of a predictor but rather to the presence of influential instances in a data set for a given predictor (so-called *leverage points*).

The effect of bagging is explained as equalizing the influence of all training instances, which is beneficial when highly influential instances are harmful for the predictor's accuracy.

2.3.3 Justification of the RESVM algorithm

We have shown the effect of resampling contaminated sets and provided some basic insight into the mechanics of bagging. We will now link these two elements to justify bagging approaches in the context of contaminated training sets. Its usefulness can be considered by both the variance reduction argument of **(author?)** [5] and equalizing the influence of training points as described by **(author?)** [42].

Variance reduction Resampling a contaminated set yields different levels of contamination in the resamples as explained in Section 2.3.1. Varying the contamination between base model training sets induces variability between base models without increasing bias. This observation enables us to create a diverse set of base models by resampling both $\mathcal{P}$ and $\mathcal{U}$. The variance reduction of bagging is an excellent mechanism to exploit the variability of base models based on resampling [5, 17]. In the context of RESVM, a tradeoff takes place between increased variability (by training on smaller resamples, see Figure 2.1) and base models with increased stability (larger training sets for the SVM models).

Equalizing influence The influence of a training instance on an SVM model can be quantified in terms of its dual weight (the associated α value). Three distinct cases can be distinguished: (i) the training instance is correctly classified and not within the margin ($\alpha = 0$, not a SV), (ii) the training instance lies on the margin and is correctly classified ($\alpha \in [0, C]$, free SV) and (iii) the training instance is incorrectly classified or within the margin ($\alpha = C$, bounded SV), where C is the misclassification penalty associated to the training instance [14]. Instances that are misclassified during training become bounded SVs, which have the maximal α value and can therefore be considered leverage points of the SVM model. When learning with label noise, the mislabeled training instances are likely to end up as bounded SVs. In a best case scenario, the mislabeled training instances are classified in concordance to their true label by the SVM model (which means they must be a bounded SV as the training procedure identifies this as a misclassification). As such, mislabeled training instances act as leverage points for SVM models. Following **(author?)** [42], bagging equalizes the influence of training instances (e.g. lowers the influence of

mislabeled leverage points in comparison to the rest of the data) which yields improved robustness against contamination in the context of RESVM.

2.3.4 RESVM training

RESVM uses CWSVM base models trained on resamples from the original training set, where both $\mathcal{P}$ and $\mathcal{U}$ are being resampled. The technique involves 5 hyperparameters: 3 to define the resampling strategy and 2 for the base models. Additional hyperparameters may be involved, for example γ for the RBF kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\gamma\|\mathbf{x}_i - \mathbf{x}_j\|^2)$.

The number of base models to include in the ensemble, $n_{nmodels}$, is the first hyperparameter. Using more base models improves the stability of the ensemble (up to a certain plateau) at a linear increase in computational cost for training and prediction. n_{models} is not a traditional hyperparameter in the sense that a good value can be determined during training, for example based on out-of-bag error estimates [4].[1]

By resampling $\mathcal{P}$, RESVM takes potential contamination of the labeled instances into account by design. Since the contamination between $\mathcal{P}$ and $\mathcal{U}$ can vary, the ability to vary the size of resamples from $\mathcal{P}$ and $\mathcal{U}$ separately is required. This results in two tuning parameters: n_{pos} and n_{unl}. In general, using small base model training sets results in increased base model variability which then necessitates using more base models in the ensemble to obtain a given level of stability. In our experiments, we have tuned n_{pos} and n_{unl} but it is also possible to obtain good values using out-of-bag techniques [62].[1]

RESVM additionally inherits at least 2 hyperparameters from its SVM base models, namely misclassification penalties for both classes and, if applicable, hyperparameters related to the kernel function. We define the CWSVM penalties in see Eq. (2.1) based on 2 hyperparameters $C_{\mathcal{U}}$ and w_{pos}:

$$C_{\mathcal{P}} = C_{\mathcal{U}} \times w_{pos} \times \frac{n_{unl}}{n_{pos}}. \tag{2.3}$$ {eq:resvmpenalties}

w_{pos} enables reweighting labeled and unlabeled instances after equalizing class imbalance. In bagging SVM, w_{pos} is always fixed to 1.

[1] Note that the error estimates in out-of-bag techniques must account for potential contamination. See our discussion of hyperparameter tuning for a possible score function. {labelednote}

The RESVM training approach has been summarised in Algorithm 1. The algorithm uses 5 hyperparameters plus additional kernel parameters.

{RESVM}

Algorithm 1: Training procedure for RESVM.

Data: $\mathcal{P}$: the set of positive instances.
$\mathcal{U}$: the set of unlabeled instances.
Input: n_{models}: number of base models to include in the ensemble.
n_{unl}: size of bootstrap resamples of $\mathcal{U}$.
n_{pos}: size of bootstrap resamples of $\mathcal{P}$.
$C_{\mathcal{U}}$: misclassification penalty for $\mathcal{U}$ in class-weighted SVM.
w_{pos}: relative positive misclassification penalty coefficient.
$\kappa(\cdot,\cdot)$: kernel function to be used by base models.
Output: Ω: RESVM with n_{models} base models.
begin
  $\Omega \leftarrow \emptyset$;
  $C_{\mathcal{P}} \leftarrow C_{\mathcal{U}} \times w_{pos} \times \frac{n_{unl}}{n_{pos}}$;
  for $i \leftarrow 1$ **to** n_{models} **do**
    `// create base model training set from` $\mathcal{P}$ `and` $\mathcal{U}$.
    $\mathcal{P}^{(i)} \leftarrow$ sample n_{pos} instances from $\mathcal{P}$ with replacement;
    $\mathcal{U}^{(i)} \leftarrow$ sample n_{unl} instances from $\mathcal{U}$ with replacement;
    `// train CWSVM base model` $\psi^{(i)}$ `and add to ensemble` Ω.
    $\psi^{(i)} \leftarrow$ train CWSVM for $\mathcal{P}^{(i)}$ vs. $\mathcal{U}^{(i)}$ (parameters $C_{\mathcal{P}}$, $C_{\mathcal{U}}$, κ);
    $\Omega \leftarrow \{\Omega, \psi^{(i)}\}$;

{RESVM}

2.3.5 RESVM prediction

{ensembledecvals}

RESVM uses majority voting to aggregate base model predictions. By default, the returned label is the one predicted by most base models. The fraction of positive votes for a test instance $\mathbf{x}$ can be written as:

{eq:majorityvote}

$$v(\mathbf{x}) = \frac{n_{models} + \sum_{i=1}^{n_{models}} \mathtt{sgn}(\psi^{(i)}(\mathbf{x}))}{2n_{models}}, \tag{2.4}$$

where $\mathtt{sgn}(\cdot)$ is the sign function and $\psi^{(i)}$ denotes the decision function of SVM base model i with codomain $\mathbb{R}$. $v(\cdot)$ has the interval $[0, 1]$ as codomain.

The RESVM decision value for a test instance $\mathbf{x}$ is defined as the fraction of votes in favor of the positive class $v(\mathbf{x})$ unless the result is unanimous. In the case of a unanimous vote, the ensemble decision value is based on the decision values of its base models to increase the model's ability to differentiate. In case

of a unanimous negative vote, the sum of the decision values of the base models is taken (each SVM base model decision value is negative in this case). In case of a unanimous positive vote, the sum of the decision values of the base models (all positive) plus one is taken. The decision value $d(\cdot)$ has codomain $\mathbb{R}$ and is computed as follows:

$$d(\mathbf{x}) = \begin{cases} v(\mathbf{x}) & \text{if } 0 < v(\mathbf{x}) < 1, \\ \sum_{i=1}^{n_{models}} \psi^{(i)}(\mathbf{x}) & \text{if } v(\mathbf{x}) = 0, \\ 1 + \sum_{i=1}^{n_{models}} \psi^{(i)}(\mathbf{x}) & \text{if } v(\mathbf{x}) = 1. \end{cases} \tag{2.5}$$ {eq:resvmdecval}

The resulting label for a given decision threshold T can be written as follows:

$$l(\mathbf{x}) = \mathtt{sgn}\big(d(\mathbf{x}) - T\big). \tag{2.6}$$ {eq:resvmlabel}

The default decision value threshold for positive classification is $T = 0.5$ (this is majority voting, e.g. positive iff more than half of all base models predict positive). Using the modified decision values $d(\mathbf{x})$ instead of the votes $v(\mathbf{x})$ does not affect the predicted labels for typical choices of the threshold T (e.g. $T \in (0, 1)$). It does, however, affect performance measures that use the entire range of decision values such as area under the PR curve. Using $d(\mathbf{x})$ enables us to rank different instances that received all positive or all negative votes by base models (e.g. $v(\mathbf{x}) = 1$ and $v(\mathbf{x}) = 0$, respectively).

2.4 Experimental setup

RESVM has been compared to class-weighted SVM (CWSVM) and bagging SVM (BAG) in a number of simulations to assess the merits of our modifications compared to conceptually comparable algorithms. In this Section we will summarize the experimental setup (training set construction, model selection and performance evaluation) and the data sets we used.

2.4.1 Simulation setup

Our experiments consist of repeated simulations on a variety of data sets under different settings. Briefly, in each iteration hyperparameters were optimized per approach based on cross-validation on the training set (using identical folds for all approaches). Subsequently, a model with the optimal parameters is trained on the full training set and used to predict an independent test set. An overview of the experiments is shown in Figure 2.2. Every experiment consists of 20 repetitions.

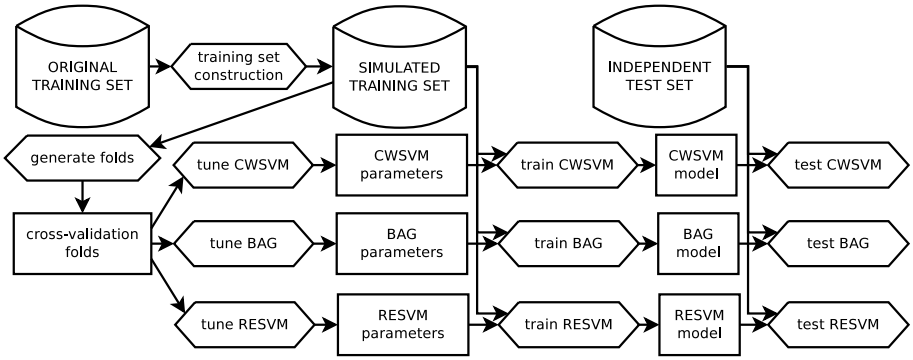

Figure 2.2: Overview of a single benchmark iteration. {fig:benchmark}

To assess what situations are favorable per approach, we have investigated three different settings with distinct label noise configurations. For every data set, we performed 10 iterations per simulation in the following settings:

1. **supervised**: no contamination in $\mathcal{P}$ or $\mathcal{U}$ ($\mathcal{U}$ is the negative class).
2. **PU learning**: contamination in $\mathcal{U}$ but not in $\mathcal{P}$.
3. **semi-supervised**: contamination in both $\mathcal{P}$ and $\mathcal{U}$. The contamination levels in $\mathcal{P}$ and $\mathcal{U}$ were always chosen equal.

The contamination levels we used were chosen per data set based on when differences between the three approaches become visible. A summary is available in Table 2.1 in Section 2.4.2. When applicable, contamination was introduced by flipping class labels (e.g. true positives in $\mathcal{U}$ and true negatives in $\mathcal{P}$). This effectively changes the empirical densities of the classes in the training set (illustrated in Figure 2.3 in the next Section).

Every binary learning task was repeated 20 times to get reliable assessments of all methods. Each repetition involves redoing all steps shown in Figure 2.2, including resampling of training sets based on the known true positives and true negatives. Contamination was introduced at random where applicable by flipping class labels.

Hyperparameter selection In every iteration, hyperparameters were tuned per setting using 10-fold cross-validation over a grid of parameter tuples. To ensure a fair comparison, one set of folds is generated in each iteration and used by all methods. We ensured that the optimal values that were found during

tuning in any setting were never on the edge of the search grid. The search resolution in comparable parameters between methods was always defined to be identical (for example γ in the case of an RBF kernel).

The same search grids were used in all three settings for a given data set to illustrate that a method can work well in a supervised setting with a given search grid but degrade when label noise is added. Since negative labels are unavailable in PU learning, we used the following score function in all learning settings which only requires positive labels for hyperparameter selection [54]:

$$\texttt{pu\_score} = \frac{\texttt{precision} \times \texttt{recall}}{Pr(y=1)} = \frac{\texttt{recall}^2}{Pr(\hat{y}=1)}, \tag{2.7}$$ {puscore}

where $Pr(y=1)$ is the fraction of known positive labels in the predicted set and $Pr(\hat{y}=1)$ is the fraction of positive predictions made by the classifier. Note that this score function is not ideal when $\mathcal{P}$ is contaminated, though we obtained good results even in that setting.

The following parameters were tuned per method: (CWSVM) $C_{\mathcal{P}}$ and $C_{\mathcal{U}}$, (BAG) $C_{\mathcal{U}}$ and $n_{\mathcal{U}}$ and (RESVM) $C_{\mathcal{U}}$, w_{pos}, n_{pos} and n_{unl}. In both ensemble approaches we consistently used 50 base models.

Performance assessment Models are trained with the optimal hyperparameters on the full training set and subsequently tested on the independent test set. We use the known test labels to compute the area under the Precision-Recall curve (AUC) for each model. We opted to use PR curves because they capture the performance of interest of models over their entire operating range and work well for imbalanced data [29].

We used statistical analyses to determine whether one approach trumps another while accounting for the variability between simulations. The nonparametric Wilcoxon signed-rank test is recommended for pairwise comparisons between learning algorithms [31]. In every setting per data set we performed a paired one-tailed Wilcoxon signed-rank test comparing the area under the PR curve of bagging SVM and RESVM with alternative hypothesis h_1 : $AUC^{RESVM} > AUC^{BAG}$ (pairs being iterations). Low p-values indicate a statistically significant improvement.

Implementation details We used the class-weighted SVM implementation available in LIBLINEAR [38] and LIBSVM [21] for models using the linear and RBF kernel, respectively. Bagging SVM and RESVM were implemented using the EnsembleSVM library [23].[2] The decision values of bagging SVM

[2]Python code for RESVM is available at `https://github.com/claesenm/resvm`.

used to compute PR curves were defined in the same way as for RESVM (see Section 2.3.5).

{data}

2.4.2 Data sets

We used a synthetic data set and 5 publicly available data sets:[3]

- `synthetic`: a 2-D binary data set. Positive instances are sampled from a standard normal distribution. Negative instances are sampled from a circle centered at the origin with radius 4 with 2-D noise superimposed from a standard normal distribution. Training and testing data was generated in every iteration. Figure 2.3 shows densities for all settings.
- `cancer`: the Wisconsin breast cancer data set related to breast cancer diagnosis. It consists of 10 features and 683 instances without an explicit train/test partitioning so we partitioned it at random in every iteration.
- `ijcnn1`: used for the IJCNN 2001 neural network competition [71], comprising 2 classes, 22 features and $49,990/91,701$ training/testing instances.
- `covtype`: a common classification benchmark about predicting forest cover types based on cartographic information [12]. We used a subsample of $100,000/40,000$ training/testing instances.
- `mnist`: a digit recognition task [53]. This data set contains 10 classes (one for each digit), 780 features, $60,000$ training instances and $10,000$ test instances with an almost uniform class distribution. We performed one-versus-all classification for each digit.
- `sensit`: SensIT Vehicle (combined), vehicle classification [33]. This data set contains 3 classes with an uneven distribution. We performed one-versus-all classification for each class. This data set has 100 features, $78,823$ training instances and $19,705$ testing instances.

Most data sets have a prespecified test set, except for `synthetic` and `cancer`. We used the prespecified test sets when available. We used the RBF kernel for all data sets except `mnist` (linear kernel). Note that both RESVM and bagging SVM models are always implicitly nonlinear due to their majority voting scheme, even when using linear base models.

[3]Public data at: `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`.

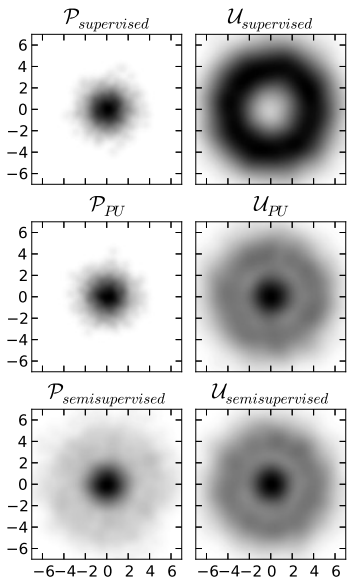

Figure 2.3: Empirical densities of the `synthetic` data used for training per problem setting (visualized in input space). The supervised densities (top row) are based on samples of the underlying positive and negative classes. The use of high contamination (30%) induces similar empirical densities for $\mathcal{P}$ and $\mathcal{U}$ in the semi-supervised setting (bottom row).

{fig:densities}

In every setting each original data set was resampled without replacement to construct training sets to use in the simulations. The resampled training sets are typically significantly smaller than what is available in the original data sets to show that some methods can obtain good models even with few training instances. An overview of the actual training sets we constructed is presented in Table 2.1.

data set	d	contamination in percent	training set $\lvert\mathcal{P}\rvert$	training set $\lvert\mathcal{U}\rvert$	test set $\lvert\mathcal{P}\rvert$	test set $\lvert\mathcal{N}\rvert$
`synthetic`	2	30	100	200	$5,000$	$5,000$
`cancer`	10	30	50	200	100	100
`ijcnn1`	22	10	100	$10,000$	$8,712$	$82,989$
`covtype`	54	30	100	1,000	20,000	20,000
`mnist`	780	10	50	$2,000$	$\approx 1,000$	$\approx 9,000$
`sensit 1`	100	30	100	$1,000$	$4,575$	$15,130$
`sensit 2`	100	30	100	$1,000$	$5,520$	$14,455$
`sensit 3`	100	30	100	$1,000$	$9,880$	$9,825$

Table 2.1: Overview of the data sets used in simulations: number of features, contamination (when applicable), training set size as used in the experiments and test set size. The `mnist` data set consists of 10 classes and the test set is almost uniformly distributed. The `sensit` data set has 3 classes with uneven class distribution in the test set, so we treat it separately here. {table:datasets}

2.5 Results and discussion

We will summarize all results of our simulation experiments comparing class-weighted SVM (CWSVM), bagging SVM (BAG) and the robust ensemble of SVMs (RESVM). First we will show the results of each setting separately. Subsequently we present an overview of the number of wins per setting for each method across all data sets. Section 2.5.6 shows the results of an experiment to assess the effect of contamination in $\mathcal{P}$ and $\mathcal{U}$ on all methods. Finally, we include an interesting observation regarding the optimal hyperparameters of RESVM that were found using cross-validation on the `mnist` data set per setting in Table 2.6.

2.5.1 Results for supervised classification

Table 2.2 summarizes our results in a fully supervised setting. In these experiments both $\mathcal{P}$ and $\mathcal{U}$ are uncontaminated. Based on the number of wins per simulation and the confidence intervals, we can conclude that all methods are competitive in this setting.

The confidence intervals show that all methods obtain comparable results for all simulations except `mnist` digit 8, where CWSVM performs poorly compared to the others. This performance difference could be caused by the fact we used linear class-weighted SVM while both ensemble methods implicitly yield

nonlinear decision boundaries. A linear model may be too simple to properly distinguish this digit from the others.

The overall good results in the supervised setting confirm that the score function in Equation (2.7) is a good choice for tuning. In these supervised experiments we could have used a traditional score like accuracy, area under the ROC curve or F-measure, but these would no longer be useful in the other settings. The performance in these supervised experiments can be considered an objective baseline for comparison in the PU learning and semi-supervised setting since only levels of contamination are varied.

	area under PR curve				number of wins		
data	CWSVM	BAG	RESVM	p	CWSVM	BAG	RESVM
synthetic	98.1–98.7	98.7–98.8	98.7–98.8		2	12	6
cancer	98.4–98.8	98.4–98.7	98.3–98.7		8	12	0
ijcnn1	85.3–87.4	79.1–81.6	82.3–86.2	• • •	16	0	4
covtype	77.1–78.3	76.8–78.5	76.8–78.7		8	6	6
mnist (positive = x)							
0	96.9–97.5	96.9–97.4	96.9–97.4		7	8	5
1	98.1–98.3	98.3–98.5	98.2–98.5		0	8	12
2	87.3–89.1	88.5–89.8	89.6–90.5	•	2	6	12
3	83.7–85.9	86.9–88.7	88.8–90.1	• • •	0	5	15
4	88.8–90.2	89.8–91.1	90.8–92.2	• • •	1	3	16
5	78.7–80.9	79.2–81.0	81.4–83.2	• •	3	3	14
6	92.4–93.4	93.9–94.7	94.3–94.9		0	8	12
7	92.2–92.9	92.6–93.2	93.1–93.7	• • •	1	3	16
8	56.5–58.9	74.3–76.1	79.6–80.5	• • •	0	0	20
9	72.5–75.6	77.8–80.3	81.5–82.6	• • •	0	2	18
sensit (positive = x)							
1	80.5–81.4	79.8–80.7	80.5–81.3	•	10	2	8
2	65.7–75.4	72.6–74.0	73.5–74.9	• • •	15	0	5
3	35.5–56.1	92.3–92.7	91.7–92.3		0	15	5

Table 2.2: 95% CIs for mean test set performance in a fully supervised setup, the results of a paired one-tailed Wilcoxon signed-rank test comparing the AUC of BAG and RESVM with alternative hypothesis $h_1 : AUC^{RESVM} > AUC^{BAG}$ and the number of times each approach had best test set performance. Test result encoding: • $p < 0.05$, • • $p < 0.01$ and • • • $p < 0.001$. {table:supervised}

2.5.2 Results for PU learning

The results of our experiments in a PU learning setting are shown in Table 2.3. In the pure PU learning setting, $\mathcal{P}$ is uncontaminated but $\mathcal{U}$ is contaminated. Class-weighted SVM tends to suffer from the largest loss in performance between supervised learning and pure PU learning based on area under PR curves. Class-weighted SVM obtains less wins than it did in the supervised simulations (21 wins in PU learning compared to 73 in the supervised setting), except on the `cancer` data set. Bagging SVM and RESVM maintain strong performance. Bagging SVM obtains a comparable number of wins and RESVM gains many compared to the supervised setting.

On the `mnist` data, RESVM consistently exhibits the best performance (based on the Wilcoxon signed-rank test), though the effective improvement over bagging SVM is marginal. On `sensit` with classes 2 or 3 as positive, bagging SVM obtains the majority of wins though the confidence intervals of its area under the PR curve overlap completely with those of RESVM. On the other data sets, no worthwile differences were obtained between both ensemble methods.

2.5.3 Results of semi-supervised classification

In the semi-supervised setting we deliberately violated the assumption of an uncontaminated positive training set by contaminating $\mathcal{P}$ and $\mathcal{U}$. The results listed in Table 2.4 confirm that both class-weighted and bagging SVM are vulnerable to contamination in $\mathcal{P}$ and experience very large performance losses. We believe this is induced by using high misclassification penalties for training instances in $\mathcal{P}$ without any resampling to account for potential false positives. In bagging SVM this leads to a systematic bias in all base models. The resampling strategy of RESVM prevents systematic bias over all base models.

The results clearly show that RESVM is more robust to false positives, evidenced by a much lower drop in predictive performance for almost all data sets. The performance difference between bagging SVM and RESVM is statistically significant for all data sets except `covtype` and `sensit`. Surprisingly, CWSVM obtains 8 wins on `sensit` with class 2 as positive. RESVM shows the best and most consistent performance overall.

On the `mnist` data, RESVM not only achieved consistently higher area under the PR curve, but visual inspection showed that its PR curves almost always dominated the others over the entire range. This means that in this experiment, RESVM models are always better than the others regardless of design priorities (high precision versus high recall). As an illustration, Figure **??** shows the PR

data	area under PR curve			p	number of wins		
	CWSVM	BAG	RESVM		CWSVM	BAG	RESVM
synthetic	96.9–98.4	97.9–98.6	98.2–98.5		6	8	6
cancer	98.2–98.5	87.5–98.4	96.1–98.1		10	7	3
ijcnn1	71.2–76.5	73.4–78.2	72.6–80.7	•	1	5	14
covtype	65.2–67.9	70.2–72.2	71.4–73.0		0	6	14
mnist (positive = x)							
0	74.1–77.8	90.5–93.3	94.6–95.5	• • •	0	5	15
1	89.1–91.2	95.2–96.7	96.4–97.3	• •	0	5	15
2	55.2–60.1	75.5–80.0	84.2–86.1	• • •	0	0	20
3	54.6–60.2	74.5–80.3	83.6–86.2	• • •	0	2	18
4	57.8–62.5	73.9–80.3	83.9–85.9	• • •	0	2	18
5	53.3–56.7	63.8–70.3	69.1–72.6	•	0	7	13
6	66.9–71.0	85.9–89.7	90.6–92.5	• •	0	4	16
7	71.4–74.8	84.0–88.0	90.0–91.4	• • •	0	1	19
8	34.8–38.8	63.5–69.1	72.2–74.8	• • •	0	4	16
9	50.5–54.8	66.2–71.0	74.2–76.4	• • •	0	1	19
sensit (positive = x)							
1	61.6–73.0	70.6–75.3	72.5–76.2	•	2	7	11
2	58.6–68.1	68.5–70.5	67.8–70.0		2	10	8
3	33.2–50.2	90.2–91.8	89.7–91.1		0	14	6

Table 2.3: 95% CIs for mean test set performance in a PU learning setup, the results of a paired one-tailed Wilcoxon signed-rank test comparing the AUC of BAG and RESVM with alternative hypothesis $h_1 : AUC^{RESVM} > AUC^{BAG}$ and the number of times each approach had best test set performance. Test result encoding: • $p < 0.05$, • • $p < 0.01$ and • • • $p < 0.001$.

{table:pulearning}

and ROC curves of a representative simulation with digit 7 as positive. Since the PR curve of RESVM completely dominates the others we know that its ROC curve does too [29].

Finally, it is worth noting that the confidence intervals of RESVM tend to be narrower than those of both other approaches. Even though RESVM base models have more variability compared to bagging SVM base models, the overall performance of RESVM is more reliable. This constitutes an important practical advantage since assessing different models is not trivial outside of simulation studies (e.g. when no negative labels are available).

data	area under PR curve			p	number of wins		
	CWSVM	BAG	RESVM		CWSVM	BAG	RESVM
synthetic	83.6–90.0	91.9–94.9	96.4–97.4	• • •	3	2	15
cancer	62.5–80.2	91.1–96.7	96.2–97.6	•	1	8	11
ijcnn1	69.8–73.4	67.4–70.4	72.0–75.2	• • •	5	2	13
covtype	58.1–61.8	61.2–64.2	60.4–65.7		4	4	12
mnist (positive = x)							
0	59.9–64.1	72.8–81.1	91.4–93.4	• • •	0	0	20
1	80.3–82.7	90.6–93.4	96.1–97.4	• • •	0	0	20
2	42.3–48.0	55.1–63.7	79.8–83.0	• • •	0	0	20
3	43.8–47.6	59.9–66.0	78.1–81.1	• • •	0	0	20
4	52.4–56.2	66.4–72.8	79.7–83.4	• • •	0	0	20
5	40.5–45.2	56.0–61.1	65.8–69.4	• • •	0	2	18
6	52.4–57.3	72.9–79.3	87.9–90.9	• • •	0	0	20
7	58.7–61.6	69.9–77.3	87.9–90.2	• • •	0	1	19
8	29.7–33.9	48.3–55.3	68.0–71.0	• • •	0	0	20
9	42.1–44.9	52.5–59.0	68.7–72.7	• • •	0	0	20
sensit (positive = x)							
1	34.5–49.4	59.6–69.0	60.6–66.4		3	12	5
2	44.9–53.7	46.4–53.4	50.1–56.7	•	8	4	8
3	44.5–61.1	75.4–83.5	80.5–84.9	•	1	7	12

Table 2.4: 95% CIs for mean test set performance in a semi-supervised setup, the results of a paired one-tailed Wilcoxon signed-rank test comparing the AUC of BAG and RESVM with alternative hypothesis $h_1 : AUC^{RESVM} > AUC^{BAG}$ and the number of times each approach had best test set performance. Test result encoding: • $p < 0.05$, • • $p < 0.01$ and • • • $p < 0.001$.

{table:semisupervised}

2.5.4 A note on the number of repetitions per experiment

The tightness of the confidence intervals of generalization performance allow us to conclude that the number of repetitions (20) is sufficient to demonstrate the merits of RESVM (see Tables 2.2–2.4). Increasing the number of repetitions further would yield even narrower confidence intervals and increase the amount of statistically significant results in the Wilcoxon signed-rank test comparing bagging SVM and RESVM (due to increased power). All key conclusions remain valid if the number of repetitions would be increased.

Additional statistically significant results may only be obtained in experiments where the improvement offered by RESVM is too small to be of practical significance (as large improvements already yield significant test results). Failure

to reject the null hypothesis ($h_0 : AUC^{BAG} \geq AUC^{RESVM}$) in our current results indicates that (i) bagging SVM is effectively better than RESVM, (ii) they are comparable or (iii) the performance improvement of RESVM is too small to yield a significant test result given the current sample size (number of repetitions). Increasing the number of repetitions can only lead to additional statistically significant results in the latter situation.

To illustrate our claims, we performed 100 repetitions for `covtype` in the semi-supervised setting. This yielded the following CIs and win counts: CWSVM 59.0–60.5% (8 wins), bagging SVM 62.3–63.5% (21 wins), RESVM 63.8–65.8% (71 wins). The p-value of the Wilcoxon signed-rank test becomes 2×10^{-5}, while the p-value was insignificant with 20 repetitions (Table 2.4).

2.5.5 Trend across data sets

In the previous tables we have shown the results per data set for each setting. In this section we summarize the results across all data sets, using critical difference diagrams [31] in Section 2.5.5 and an overview of win counts in Section 2.5.5.

Critical difference diagrams

{cds}

In every setting, we compared the performance of the three learning approaches across all data sets using non-parametric statistical tests. For each data set, approaches were ranked based on their mean area under the PR curve across all iterations. Multiclass data sets count once per class. Friedman tests per setting yielded significant evidence of differences between the three learning approaches at the $\alpha = 0.05$ level, though this was marginal in the supervised setting ($p = 0.034$). The Nemenyi post-hoc test [67] was used after each omnibus test to assess differences between all approaches. The critical difference diagrams in Figure **??** visualize the results.

Critical difference diagrams were introduced by **(author?)** [31] to visualize a comparison of multiple learning approaches over multiple data sets. These diagrams depict the average rank of each approach (lower is better) along with the critical difference (CD). The critical difference is the minimum difference in average ranks that yields a significant result in the Nemenyi post-hoc test. It depends on the significance level ($\alpha = 0.05$), the number of learning approaches (3) and the number of data sets (17).

From Figure **??** we can conclude that bagging SVM and RESVM are comparable in the PU learning setting (both significantly better than CWSVM). In the semi-supervised setting, bagging SVM is statistically significantly better than

CWSVM and RESVM is significantly better than both other approaches across all data sets.

{wins}

Win counts

The number of wins per method across all data sets are summarized in Table 2.5. The top half shows the total number of wins across all data sets, which weights `mnist` and `sensit` heavier than the other data sets since we performed several one-vs-all experiments. Because RESVM consistently performed very strong on `mnist`, the top half is an overly optimistic representation.

The bottom half of Table 2.5 contains normalized results, where every data set contributes equally. Based on these numbers we can conclude that there is little difference between the three methods in a supervised setting. In the PU learning setting, ensemble methods become favorable over CWSVM (bagging SVM and RESVM being competitive). Finally, in the semi-supervised setting RESVM pulls far ahead of both other methods and obtains 65% of the normalized wins, which is over three times more than bagging SVM and over five times more than class-weighted SVM.

	CWSVM		bagging SVM		RESVM	
setting	count	win %	count	win %	count	win %
supervised	73	21	93	27	**174**	**51**
PU learning	21	6	88	26	**231**	**68**
semi-supervised	25	7	42	12	**273**	**80**
supervised	**44.8**	**37.3**	40.3	33.6	36.0	30.0
PU learning	18.3	15.3	39.4	32.8	**62.2**	**51.8**
semi-supervised	17.0	14.2	24.0	20.0	**79.0**	**65.8**

Table 2.5: Number of wins in simulations for each method per setting. The bottom half shows normalized number of wins, where wins in multiclass data sets (`mnist` and `sensit`) are divided by the number of classes.

{table:wins}

{varycontamination}

2.5.6 Effect of contamination

In this Section we show the effect of different levels of contamination in $\mathcal{P}$ and $\mathcal{U}$ on the `synthetic` data set. In these simulations, we fixed the contamination level in one part of the training set ($\mathcal{P}$ or $\mathcal{U}$) and the contamination of other was varied. The fixed contamination was set to 30%. Twenty simulations were run per contamination setting.

In these experiments, we used random search to tune hyperparameters of each method [6] using the Optunity package.[4] Briefly, hyperparameters were searched by random sampling 100 tuples uniformly within a given box and subsequently the best tuple was selected as before. We ensured that the optimal hyperparameters were never too close to the edge of the feasible region (if so, the box was expanded). Note that this approach of testing a fixed number of tuples favors methods with less hyperparameters. Even though RESVM has more hyperparameters than the other methods, good models can be obtained at the same search cost.

The results are shown in Figure **??**. In general, contamination in $\mathcal{P}$ causes larger performance losses than the same level of contamination in $\mathcal{U}$ for all algorithms. As expected, the difference in sensitivity to contamination in $\mathcal{P}$ and $\mathcal{U}$ is smallest for RESVM in which $\mathcal{P}$ and $\mathcal{U}$ are resampled similarly. At high contamination levels, RESVM is the only method that still works well (even at 60%).

Figure **??** illustrates that RESVM and bagging SVM behave in a similar fashion at contamination levels of $\mathcal{U}$ up to 50% and both outperform class-weighted SVM. RESVM outperforms bagging SVM for contamination levels of 30–50% but the consistency (width of CI) and performance losses of both methods are comparable. Figure **??** shows the increased robustness of RESVM to contamination in $\mathcal{P}$ resulting in reduced loss of generalization performance for increasing contamination.

2.5.7 RESVM optimal parameters

As an illustration of the implicit mechanism of RESVM we show some of the optimal tuning parameters for every setting in Table 2.6. These parameters were obtained by performing 10-fold cross-validation on the training set.

An interesting observation is that the size of the training sets that are being used decreases for increasing contamination. Increasing label noise induces RESVM to favor smaller base model training sets for which the variability in contamination is larger (see Figure 2.1). Though this may appear counterintuitive, bagging approaches are known to exhibit a bias-variance tradeoff [5] for which using weaker base models with increased variability may yield better ensembles [49].

The optimal value of the misclassification penalty for positive training instances relative to unlabeled instances, w_{pos}, changes between learning settings (see Equation (2.3)). It exhibits expected behaviour: the maximum value is obtained when the certainty on $\mathcal{P}$ relative to $\mathcal{U}$ is largest (e.g. the pure PU learning

[4]Optunity is available at: `http://www.optunity.net`.

	0	1	2	3	4	5	6	7	8	9	mean
$\mathbf{n_{pos}}$											
supervised	20	20	20	20	20	20	10	20	20	10	18
PU learning	10	10	10	10	10	15	10	10	10	10	10.5
semi-superv.	10	5	10	10	10	10	10	10	10	10	9.5
$\mathbf{n_{unl}/n_{pos}}$											
supervised	10	10	10	10	10	10	10	10	10	10	10
PU learning	5	5	5	5	5	5	5	5	5	5	5
semi-superv.	5	5	5	8	5	5	5	5	5	5	5.25
$\mathbf{w_{pos}}$											
supervised	1.6	1.6	1.6	3.2	3.2	3.2	3.2	1.6	3.2	2.4	2.48
PU learning	4.8	6.4	3.2	6.4	4.8	6.4	4.8	4.8	6.4	6.4	5.44
semi-superv.	12.8	6.4	4.8	2.1	4.8	6.4	4.8	3.2	3.2	3.2	5.17

Table 2.6: Medians of optimal hyperparameters per digit obtained via cross-validation and mean of all medians per setting. The normalized relative weight on positives versus unlabeled instances (w_{pos}) is associated with the relative size and contamination of the positive and unlabeled training sets. {table:hyperparameters}

setting). This parameter implicitly balances empirical certainty on $\mathcal{P}$ and $\mathcal{U}$ and is an important degree of freedom in RESVM. In bagging SVM, this parameter is implicitly fixed to 1 via Equation (2.2) [64]. Note that w_{pos} need not be larger than 1 (which would place extra emphasisis on the known labels after accounting for class imbalance). In highly imbalanced settings where $n_{unl} \gg n_{pos}$, the optimal value of w_{pos} may well be less than 1.

2.6 Conclusion

We have introduced a new approach for learning from positive and unlabeled data, called the robust ensemble of SVMs (RESVM). RESVM constructs an ensemble model using a bagging strategy in which the positive and unlabeled sets are resampled to obtain base model training sets. By resampling both $\mathcal{P}$ and $\mathcal{U}$, our approach is more robust against false positives than others.

The robustness of our approach to potential contamination in both $\mathcal{P}$ and $\mathcal{U}$ can be attributed to the synergy between our resampling scheme and voting aggregation. The resampling itself strongly resembles a typical bootstrap approach. RESVM uses class-weighted SVM base models though the resampling scheme is likely to work well with other types of base models.

RESVM was compared with class-weighted SVM and bagging SVM on several data sets under different label noise conditions. The trends across data sets show that bagging SVM and RESVM outperform class-weighted SVM in PU learning. In a pure PU learning setting the average improvement over existing methods is modest though RESVM classifiers exhibit lower variance in performance making it more reliable.

In the semi-supervised setting, label noise was introduced in $\mathcal{P}$ to highlight the improved robustness of RESVM compared to the other methods. Our experimental results show that RESVM remains very strong in the semi-supervised setting while both other approaches degrade dramatically. Statistical analysis showed that RESVM is significantly better than both other approaches across all data sets.

Visual inspection of the PR curves shows that in the majority of experiments the curve for RESVM not only has higher AUC but completely dominates the other curves. As such RESVM models are a good approach regardless of design priorities (high recall versus high precision).

A weakness of RESVM is its amount of hyperparameters (5 plus potential kernel parameters), though RESVM models are less sensitive to accurate tuning of these parameters than standard SVM. Our experiments indicated that although RESVM has more hyperparameters, good models can be obtained at the same search effort than the other approaches (e.g. testing the same number of hyperparameter tuples). An interesting question is whether prior knowledge regarding contamination of $\mathcal{P}$ and $\mathcal{U}$ can help in limiting the search scope for some of the hyperparameters (n_{pos}, n_{unl} and w_{pos} specifically).

List of publications

Input file chapters/publications/publications.tex does not exist. Make sure its starts with “\chapter{List of publications}”. To not include this chapter in the table of contents, use the starred version of the \chapter command...

Instructies van de faculteit:

Lijst van de publicaties door de doctorandus/a (auteur of co-auteur).