A Pairwise Ranking Based Approach to Learning with Positive and Unlabeled Examples

Sundararajan Sellamanickam
Yahoo! Labs
Bangalore, India
ssrajan@yahoo-inc.com

Priyanka Garg
The Chinese University of Hong Kong
New Territories, Hong Kong
priyanka@cse.cuhk.edu.hk

Sathiya Keerthi Selvaraj
Yahoo! Labs
Santa Clara, CA
United States
selvarak@yahoo-inc.com

ABSTRACT

A large fraction of binary classification problems arising in web applications are of the type where the positive class is well defined and compact while the negative class comprises everything else in the distribution for which the classifier is developed; it is hard to represent and sample from such a broad negative class. Classifiers based only on positive and unlabeled examples reduce human annotation effort significantly by removing the burden of choosing a representative set of negative examples. Various methods have been proposed in the literature for building such classifiers. Of these, the state of the art methods are Biased SVM and Elkan & Noto's methods. While these methods often work well in practice, they are computationally expensive since hyperparameter tuning is very important, particularly when the size of labeled positive examples set is small and class imbalance is high. In this paper we propose a pairwise ranking based approach to learn from positive and unlabeled examples (LPU) and we give a theoretical justification for it. We present a pairwise RankSVM (RSVM) based method for our approach. The method is simple, efficient, and its hyperparameters are easy to tune. A detailed experimental study using several benchmark datasets shows that the proposed method gives competitive classification performance compared to the mentioned state of the art methods, while training 3-10 times faster. We also propose an efficient AUC based feature selection technique in the LPU setting and demonstrate its usefulness on the datasets. To get an idea of the goodness of the LPU methods we compare them against supervised learning (SL) methods that also make use of negative examples in training. SL methods give a slightly better performance than LPU methods when there is a rich set of negative examples; however, they are inferior when the number of negative training examples is not large enough.

Categories and Subject Descriptors: I.5.2 [Pattern Recognition] Design Methodology-Classifier design and evaluation

General Terms: Algorithms, Performance, Experimentation

Keywords: Learning with Positive and Unlabeled Examples, Pairwise Ranking, Classification, Support Vector Machines

1. INTRODUCTION

Binary classification problems arise frequently in web applications. Traditionally binary classifiers are built via supervised learning (SL) using a training set that consists of manually collected positive and negative examples. Consider, as an example, the problem of forming a classifier that determines if a page contains reviews of restaurants or not. While it is easy to sample from the positive class, i.e., collect a representative set of pages having reviews of restaurants, it is not that easy to form a representative set of negative examples. What is usually done in practice is to analyze the domain and choose 'boundary' negative examples that are truly negative but which potentially look like positive examples. For instance, in the case of the restaurant reviews example one can choose the following as (boundary) negative examples: pages that describe restaurants but do not contain reviews; and pages that only contain reviews of businesses other than restaurants. There are two problems with this approach. (1) The relative proportion of positive and negative examples in the training set may not be indicative of the corresponding proportion in the space of pages where the classifier will be deployed. (2) It is possible that other types of negative examples that are similar to positive examples are missed during the analysis. Several other interesting web applications of a similar kind can be found in [16][23].

While it is hard to choose representative negative examples, it is usually easy to form a large collection of unlabeled examples from the space of interest. Therefore it is useful to look for methods that learn from positive and unlabeled examples (LPU). The presence of positive examples in the unlabeled set and high class imbalance (ratio of positive to negative examples is small) bring in challenges in building classifiers in the LPU setting. Various approaches have been suggested in the literature [2],[8],[9],[16],[18],[19],[20],[21], [23],[24] for solving this problem. These approaches differ in the way the examples are used, the type of classifiers employed, etc.

The main aim of this paper is to propose a new approach for LPU. This approach is related to two good LPU methods, namely, Biased SVM [18] and a method recently proposed by Elkan and Noto [9]. We will refer to these methods as BSVM and EN respectively. These methods are based on building a binary classifier that separates positive examples from unlabeled examples; the above mentioned papers also provide good theoretical justifications for the proposed idea. The BSVM method has been demonstrated to work well in practice; however, 2-dimensional hyperparameter tuning using cross-validation makes it computationally expensive. In [9] the EN method was demonstrated to work well using a default hyperparameter setting on a dataset that is not imbalanced. In general, for imbalanced datasets EN also requires expensive 2-dimensional hyperparameter tuning to achieve good classifier performance. In addition, EN requires the estimation of some probabilistic quantities, which is of concern in imbalanced situations.

Our approach for LPU is based on ranking and we give a theoretical justification for this. We use the pairwise ranking method [7], [13]. The basic idea is to first build a ranking model that encourages the score of each positive example to be higher than that of each unlabeled example and then estimate a threshold parameter θ to form the final classifier. The approach is general in the sense that any pairwise ranking method can be used. In this paper we use a support vector machine (SVM) based ranking method (RSVM) proposed in [7],[12],[13] for learning, and employ the efficient RSVM solver in [7] for fast solution. The main advantage of the proposed method over BSVM and EN is that it has only one hyperparameter (C); an additional threshold parameter (θ) can also be estimated efficiently. Further, it is easy to implement using publicly available RSVM codes [5],[14]. Experimental results show that our method gives a performance that is competitive with BSVM and EN, while being 3-10 times faster than them.

A second contribution of this paper is the proposal of a new technique for doing feature selection in the LPU setting. Feature selection is useful in web classification since it helps to remove useless/redundant features and thus reduce classifier complexity. Except the recent work of Calvo et al [3], there is no other technique for feature selection in LPU that we are aware of; Calvo et al's technique is limited to discrete variables and also requires additional knowledge on the fraction of positive examples. Our feature selection method is free of such limitations, is based on computing the best AUC value possible with each feature used alone, and works effectively; for instance, an order of magnitude reduction in the number of features can be achieved using it with a loss in F Measure performance of only 2-3%.

A third useful contribution of the paper is a detailed set of experiments that shed insight on various methods in the SL and LPU settings. The paper is organized as follows. Related work is presented in adequate detail in Section 2. Section 3 describes our method in detail followed by a presentation of our AUC based feature selection technique in Section 4. The discussion of the various methods in Section 5 forms a prelude to the detailed empirical evaluation and comparison of methods given in Section 6. We conclude the paper in Section 7.

2. RELATED WORK

Various approaches have been suggested in the literature to solve the problem of positive example based learning [2],[8],[9],[16], [18],[19],[20],[24]. These approaches differ in the way the dataset is constructed and used by the methods, and by the choice of classifiers employed. In the first approach, the dataset consists of only labeled positive examples. One-class SVM [20] is one such approach and it does not use any unlabeled examples U. The classifier is built using only P. In One-class SVM, the origin is considered to be the only member of the negative class. A previous study in [17] shows that the performance of one class SVM is poor. Wu et al [22] studied one-class SVMs in detail and showed that the poor performance can partly be attributed to document representation. They empirically showed that a modified document representation helps in improving classification accuracy.

In the second approach, unlabeled examples (U) are also used along with the labeled positive examples (P). The methods that use this approach differ in the way they use the unlabeled examples. In one such class of methods, the classifier is built iteratively. For example, suppose a set of reliable negative examples $R_N^{(i)}$ is found from the unlabeled set U, say, in i-th iteration and a classifier $\mathcal{C}_i$ is built using P and $R_N^{(i)}$. Then $R_N^{(i)}$ is expanded to $R_N^{(i+1)}$ using predictions of $\mathcal{C}_i$ on $U - R_N^{(i)}$; next a classifier $\mathcal{C}_{i+1}$ is built using P and $R_N^{(i+1)}$. This procedure is repeated until some convergence criterion is met. These techniques are referred to as *2 step techniques* in [9],[18], where identifying R_N is the first step and the second step is building the standard classifier using them. Roc-SVM[17], S-EM[19], PEBL[23] and Support Vector Mapping Convergence (SVMC) [24] are some of the well known 2-step techniques with some variations.

In another class of methods, Denis et al [8] showed how the Naive Bayes algorithm can be adapted (called positive Naive Bayes (PNB)) for learning from positive and unlabeled examples. Subsequently, Calvo et al [2] enhanced the PNB algorithm and also proposed a Bayesian approach to deal with the prior probability of the positive class.

Semi-supervised learning methods can be used to solve LPU; see [21]. But these methods require a knowledge of the fraction of positive examples in the distribution.

Biased SVM (BSVM) [18] falls under another class of positive example based learners. It treats all the unlabeled examples U as negative examples; that is, $\{y_i = 1 : i \in P\}$ and $\{y_i = -1 : i \in U\}$. The SVM classifier is built by giving appropriate weights to the positive examples P and unlabeled examples U. It solves the following optimization problem:

$$\min \quad \frac{1}{2}||\mathbf{w}||^2 + C_P \sum_{i \in P} \xi_i + C_U \sum_{i \in U} \xi_i \qquad (1)$$
$$s.t. \;\; y_i(\mathbf{w}^T \mathbf{x}_i + \theta) \geq 1 - \xi_i; \xi_i \geq 0, \forall i$$

Here C_P and C_U are the weights given to the classification error on P and U respectively; $\mathbf{w}$ and θ denote the weight vector and threshold parameter of the SVM classifier; $\mathbf{x}_i$ and ξ_i denote the i-th input feature vector and slack variable respectively. Intuitively, C_P is expected to be given a higher value than C_U. This is because P is noise-free and U contains *noise* as it contains positive examples as well. Liu et al [18] showed that the BSVM method outperforms the 2-step techniques.

Elkan & Noto [9] also considered the problem of learning from positive and unlabeled examples. Let us represent each training example by $(\mathbf{x}, y, s)$ where $y \in \{-1, 1\}$ is the class label of the example and let s be a binary variable that denotes if the example is labeled ($s = 1$) or unlabeled ($s = 0$). By making the (reasonable) assumptions that (i) P is chosen by labeling randomly chosen positive examples and (ii) $p(s = 1|y = -1) = 0$ (that is, negative examples are never labeled), they derived the following useful result:

$$p(s = 1|\mathbf{x}) = Kp(y = 1|\mathbf{x}) \quad \text{where} \quad K = p(s = 1|y = 1) \qquad (2)$$

Note that K is independent of $\mathbf{x}$ and so $p(s = 1|\mathbf{x})$ and $p(y = 1|\mathbf{x})$ are proportional to each other.

Using the above result Elkan & Noto [9] suggested two methods. In their first method (EN1), a first stage classifier that predicts probability of an example as *labeled* $p(s = 1|\mathbf{x})$ is learnt by treating each positive example as belonging to *labeled* class and each unlabeled example as belonging to *unlabeled* class. Now given a test example $\mathbf{x}$, $p(s = 1|\mathbf{x})$ is estimated using this classifier; then $p(y = 1|\mathbf{x})$ is estimated as: $p(y = 1|\mathbf{x}) = p(s = 1|\mathbf{x})/K$. The constant $K = p(s = 1|y = 1)$ is estimated using a validation set. In the second method (EN2), another classifier (second stage) is learnt by giving each training example a different weight using the estimated probability from the first stage classifier. EN2 is much more complex than EN1 because it involves two stages of classification. In the rest of the paper we will use EN1 to represent Elkan and Noto's method, and simply refer to it as EN.

Since the new approach that we propose for LPU in section 3 is based on ranking, it is useful to review some related work on ranking methods. Learning to rank is an important problem [1],[7],[11],[12],[13] in various applications like web search ranking, information retrieval etc. For example, in web search ranking the training data consists of a number of queries and for each query there is an associated set of returned documents. For each (query, document) pair there is a feature vector $\mathbf{x}_i, i = 1, \ldots, n$ and a relevancy judgment of how suitable the document is to the query. The aim is to build a ranking model that ranks a set of documents based on relevance scores for a given query. There are several methods known as *pointwise* methods, *pairwise* methods, etc., to address this problem. See [7] and references given there. Here we consider only the pairwise ranking method. In this method first a set of preference pairs $\mathbf{Q}$ is constructed by comparing the relevance of the documents associated with a given query. If $(i, j) \in \mathbf{Q}$ then document i is preferred over document j. A specific example is the ranking model based on SVM (RSVM) that is built by minimizing the following objective function [7]:

$$\frac{1}{2}||\mathbf{w}||^2 + C \sum_{(i,j)\in\mathbf{Q}} g(\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_j) \qquad (3)$$

where $g(\cdot)$ is a suitable loss function. Typical loss functions are $L1$ and $L2$ loss functions defined as: $g(t) = max(0, 1-t)$ and $g(t) = max(0, 1-t)^2$. The use of the $L1$ loss leads to the method in [13]. In the next section we show how this formulation can be used to learn from the labeled positive and unlabeled examples. Other pairwise rank modeling methods like RankNet [1] and RankBoost [11] can also be used.

3. PROPOSED METHOD

Our aim is to find a classifier function $f(\mathbf{x}) = \mathbf{w}^T\mathbf{x}+\theta$ such that $f(\mathbf{x}) = 0$ represents the boundary between the positive and negative classes; we want to do this using only positive and unlabeled examples. The essence of our approach consists of two steps. First we find $\mathbf{w}$ such that, along the $\mathbf{w}^T\mathbf{x}$ score axis positive examples are placed higher than negative examples. Given the lack of negative examples, we instead work on placing the positive examples higher than the unlabeled examples and this can be achieved using a ranking method. In subsection 3.1 we borrow ideas from Elkan and Noto [9] to give a theoretical justification for this. A ranking method such as RSVM is used to form $\mathbf{w}$. In the second step we choose the threshold θ by maximizing a proxy F Measure. We give all the details associated with these steps as well as the complete design in the rest of the section.

3.1 Pairwise Ranking Based Approach

Let us begin by recalling (2). If we take two examples $\mathbf{x}_i$ and $\mathbf{x}_j$, (2) implies

$$\frac{p(y=1|\mathbf{x}_i)}{p(y=1|\mathbf{x}_j)} = \frac{p(s=1|\mathbf{x}_i)}{p(s=1|\mathbf{x}_j)} \qquad (4)$$

In fact Elkan & Noto [9] make the observation that *if the classifier is only used to rank examples* $\mathbf{x}$ *according to the chance that they belong to class* $y = 1$, *then the classifier* $p(s = 1|\mathbf{x})$ *can be used directly instead of* $p(y = 1|\mathbf{x})$.

In linear classifiers such as logistic regression (where the probability function is in-built) and large margin methods such as SVM (where the probability function is formed after the classifier is designed), the probability of belonging to the positive class at a given $\mathbf{x}$ has a monotonic relationship with the scoring function $\tilde{f}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$.

For us the take away from the above two results is the following. Let P, N and U denote positive, negative and unlabeled example index sets. We are interested in building a classifier that ranks positive examples higher than negative examples, i.e., $\tilde{f}(\mathbf{x}_i) > \tilde{f}(\mathbf{x}_j)$ for all $(i,j) \in P \times N$. By the monotonic relation between $\tilde{f}(\mathbf{x})$ and $p(y = 1|\mathbf{x})$ this is same as requiring $\frac{p(y=1|\mathbf{x}_i)}{p(y=1|\mathbf{x}_j)} > 1$ for all $(i,j) \in P \times N$. By (4), this is equivalent to asking for $\frac{p(s=1|\mathbf{x}_i)}{p(s=1|\mathbf{x}_j)} > 1$ to be satisfied, and, given only P and U we can use $(i,j) \in P \times U$ to achieve this. By invoking the monotonic relation between $\tilde{f}(\mathbf{x})$ and $p(s = 1|\mathbf{x})$ we can see that building such a classifier is same as building a classifier that satisfies $\tilde{f}(\mathbf{x}_i) > \tilde{f}(\mathbf{x}_j)$ for all $(i,j) \in P \times U$.

While the above results and arguments suggest how to build a ranking model ($\tilde{f}$) in the LPU setting, we need a classifier model (f) for which we need to set a value for the threshold parameter θ. We determine this parameter using an idea given by Liu et al [19]; we will give the details in subsection 3.3.

3.2 RSVM Formulation

A ranking model based on SVM (RSVM) [7],[12],[13] is constructed by minimizing a regularized margin based pairwise loss as given in (3). Our binary classification context correspond to a simpler case of the more general ranking model given in [7] with a single query and two relevance values; the given n examples can be partitioned into two sets that define a binary classification problem: $A = \{i : \mathbf{x}_i\ is\ in\ the\ higher\ relevance\ class\}$ and $B = \{i : \mathbf{x}_i\ is\ in\ the\ \ lower\ relevance\ class\}$. In our context, we set $A = P$ and $B = U$; that is, the labeled positive examples belong to the higher relevance class and the unlabeled examples belong to the lower relevance class. Then from (3) we have the following objective function:

$$\frac{1}{2}||\mathbf{w}||^2 + C \sum_{(i\in P, j\in U)} g(\mathbf{w}^T\mathbf{x}_i - \mathbf{w}^T\mathbf{x}_j) \qquad (5)$$

where $g(\cdot)$ is a suitable loss function and we choose $g(\cdot)$ to be the L2-loss function. One can also use L1-loss function; often there is very little difference in their performances. This objective function penalizes any violation of $\tilde{f}(\mathbf{x}_i) > \tilde{f}(\mathbf{x}_j), i \in P, j \in U$ with some margin where $\tilde{f}(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$; it can be minimized using the Truncated Newton method efficiently [7]. A key aspect of this objective function is that the summation is over all pairs of examples in the sets P and U; therefore, the computational cost can be prohibitively high if done crudely. In [7] a very efficient solution to this problem is obtained with computational complexity $O(n \log n + n_{nz})$ where n and n_{nz} denote the number of training examples and number of non-zero elements in the $(|P| + |U|) \times d$ data matrix respectively. (Here d is the number of features, i.e., the dimension of $\mathbf{x}$ and $\mathbf{w}$.)

3.3 Hyperparameter, Threshold parameter Optimization

The choice of the hyperparameter C in (5) and the threshold parameter θ play important roles in the classification performance. They can be selected using standard 5-fold cross-validation (CV) technique by computing a suitable measure. In this technique we partition the sets P and U separately into 5-folds. We build five models where each model is built using one combination of 4-folds of data and then evaluated on the corresponding left-out 5th fold of data.

In traditional binary classification problems, measures like validation set accuracy and F Measure are typically used. While the accuracy is defined as the percentage of examples correctly classified, F Measure is defined as $F = \frac{2pr}{p+r}$ where p and r denote precision and recall (with respect to the positive class) respectively. Precision p is defined as: $p = \frac{TP}{TP+FP}$ and recall r is defined as: $r = \frac{TP}{|P|}$ where TP and FP denote the number of true positive and false positive examples respectively.

$\hat{F}$ **Optimization** Unlike standard binary classification problems, we do not have all the examples labeled; also, we have *only* positive examples. Therefore, it is not possible to find FP and hence we cannot directly use the F Measure. Simply using the accuracy measure based on the labeled positive examples is not good enough. This is because we also would like to to minimize the number of unlabeled examples getting classified as positive. This requirement is motivated by the key observation made by Liu et al [18],[19]: "*If the sample size is large enough, minimizing the number of unlabeled examples classified as positive while constraining the positive examples to be correctly classified will give a good classifier.*" While using this principle as a guideline it is also necessary to be in tune with F Measure (giving appropriate importance to the positive class). The following F Measure like quantity (we call it as *proxy F Measure*) due to Lee and Liu [16][1] satisfies these requirements:

$$\hat{F} = \frac{r^2}{\hat{P}(f(\mathbf{x}) \geq 0)} \tag{6}$$

where $\hat{P}(f(\mathbf{x}) \geq \mathbf{0}) = \frac{\mathbf{TP+UP}}{\mathbf{n}}$ is the fraction of examples labeled as positive; UP (Unlabeled Positive) denote the number of unlabeled examples classified positive and $r = \frac{TP}{|P|}$ as earlier. Therefore, we can rewrite $\hat{F}$ as: $\hat{F} = \frac{TP^2}{TP+UP}\frac{n}{|P|^2}$. Since n and $|P|$ are fixed, it is seen that *the numerator maximizes the number of labeled positive examples to be correctly classified as positive and the denominator minimizes the number of unlabeled examples classified as positive*. This measure when used to optimize both C and θ in the LPU setting gives a good classifier. For a given C value we find the optimal value for the threshold parameter θ using 5-fold CV technique. The threshold value that gives the maximum average $\hat{F}$ over the 5-folds is chosen as the optimal threshold value $\theta^*(C)$. The optimal C value is chosen similarly as the one that gives the best $\hat{F}$ value. In all our experiments we used this approach to choose the hyperparameter. Finally we note that it is also possible to use other measure like AUC (Area under the ROC) [13] (to choose C) given below:

$$AUC = \frac{|(i,j) \ such \ that \ i \in P, j \in U, \mathbf{w}^T\mathbf{x}_i > \mathbf{w}^T\mathbf{x}_j|}{|P||U|} \tag{7}$$

We can estimate C by maximizing a 5-fold CV estimate of AUC; however, given that AUC is unaffected by θ, estimation of θ has to be done using an alternate measure such as $\hat{F}$.

3.4 Implementation

The algorithmic implementation of the proposed method for learning with positive and unlabeled examples (LPU) is given in algorithm 3.1. This algorithm is quite easy to implement with a wrapper module (to perform optimization of C and θ) and having the publicly available RSVM codes [5] and [14] to solve (5) as the core module. As mentioned earlier, an efficient solution to solve (5) is very useful to reduce the computational complexity. Some variations of algorithm 3.1 include using the AUC-score (7) to find the optimal C value, choice of N_f, choice of C values in $\bar{\mathbf{C}}$, etc.

[1]Lee and Liu [16] designed this measure for selecting hyperparameters; they did not use it for tuning θ.

Algorithm 3.1 Pairwise RSVM based LPU Algorithm

```
• Set $\bar{\mathbf{C}} = \{C_{min}, \ldots, 10^{-5}, 10^{-4}, 10^{-3}, \ldots, C_{max}\}$ and $N_f$=5
• Partition the dataset $P$ and $U$ into $N_f$ partitions, $Q^{(i)} = (P^{(i)}, U^{(i)}), i = 1, \ldots, N_f$
• For each hyperparameter value $C \in \bar{\mathbf{C}}$
    1. Perform $N_f$ fold CV with the $\hat{F}$ Measure (6) using the partitions $\{Q^{(i)}, i = 1, \ldots, N_f\}$
        (a) Solve (5) using any pairwise RSVM method for each split of train set $(P - P^{(i)}, U - U^{(i)})$
        (b) Compute predictions $\tilde{f}^{(i)}(\mathbf{x}_j)$, $\{j, j \in Q^{(i)}\}$
    2. Find the optimal threshold value $\theta^*(C)$ that maximizes the average $\hat{F}$ Measure (6) using the predictions
    3. Update the best $C$ value ($C^*$) that maximizes the average $\hat{F}$ Measure using the predictions
• Solve (5) with the entire dataset $\{P, U\}$ with $C^*$
• Output: $\mathbf{w}$ and $\theta^*(C^*)$
```

4. FEATURE SELECTION - LPU SETTING

Feature selection is useful in web classification since it helps to remove useless/redundant features and thus reduce classifier complexity. While there have been several techniques proposed in the literature to address the problem of feature selection in the supervised learning (SL) setting, class imbalanced condition, etc (see eg., [10], [25]), little has been done in the LPU setting. We are not aware any work other than that of Calvo et al [3], who proposed a correlation based filter selection (CFS) technique which forwardly selects the features by maximizing a merit function; this function is based on the correlation between each feature and the class and on the correlation among the features. The technique has two limitations: (i) it works only on discrete data; and (ii) it requires additional information in the form of either the overall probability of positive examples or some (parameterized) beta distribution that models this probability.

Here we propose an AUC based feature selection (AUCFS) technique. It is generic in the sense that it can also be used in the supervised learning setting. It is free of the type of limitations mentioned for Calvo et al's method [3].

The AUCFS technique is a filter technique where we take each feature one by one and compute achievable AUC score using only that feature on the training set. This is done as follows. In the single feature scenario the scoring function is given by $\tilde{f}(x) = w \cdot x$ where w and x are scalar quantities. Then the AUC score is computed using (7). Further with w a scalar and $x_{i,k}$ and $x_{j,k}$ denoting k-th feature value of i-th and j-th examples, $w.x_{i,k} < w.x_{j,k}$ imply either $x_{i,k} < x_{j,k}$ or $x_{i,k} > x_{j,k}$ depending on the sign of w. Therefore only the sign of w matters and not the magnitude of w. Thus we *do not have* to find w for any individual feature. Now to take care of the sign part, we can compute two AUC scores one each by considering the actual feature value and its negated value separately. This is necessary because the values for some feature could be higher for positive class examples and lower for negative class examples, and vice-versa. Let us denote $AUC_P^{(k)}$ and $AUC_N^{(k)}$ as the AUC scores corresponding to using the k-th feature value and its negated value. Then we compute the AUC score for the k-th feature as: $AUC^{(k)} = \max(AUC_P^{(k)}, AUC_N^{(k)})$; we choose max because with appropriate sign for the weight, the maximum value can be achieved. These quantities can be computed efficiently by sorting the feature values. After doing this computation for all features, the features are ranked by sorting $AUC^{(k)}, \forall k$ in decreasing order and the required number of features are selected from the top of the list. We do not claim any optimal properties for this selection; nevertheless, we found this approach to work well in our experiments.

We note that the AUCFS technique can be applied in the SL setting also. While the procedure to compute the scores remains the same, we have $j \in N$ (negative examples) instead of $j \in U$. We feel that, compared to other feature selection metrics AUCFS has the innate ability to be less affected by differences in the distribution of positive and negative examples in the training and test sets. This needs to be tested. For the LPU setting, although it is possible to consider other measures like proxy F Measure, here we restrict our attention to AUCFS technique only. In subsection 6.5 we do experiments to demonstrate the usefulness of AUCFS.

5. DISCUSSION

A highlight of this paper is a set of experiments that shed insight into various methods in the SL and LPU settings. These experiments and the conclusions derived from them are described in section 6. In this section we discuss various practical aspects of the methods and raise questions that form a prelude to the experiments.

SL with SVM and RSVM Since SVM and RSVM (as applied to binary classification in the SL setting) form the basis for LPU methods, it is useful to understand them comparatively. Although these methods are well known we are unaware of any results that compare binary classifiers built using them under varying train/test distributions. Note that in the SL setting we have both positive and negative examples for training. Suppose our aim is to maximize F Measure. Since the loss functions associated with SVM or RSVM are not truly designed to optimize F Measure, it is useful to adjust the classifier threshold θ after the models are trained. We can use 5-fold CV based F Measure to estimate θ; the same measure can also be used to optimize the hyperparameters. Hyperparameter optimization is easier with RSVM since it involves only one hyperparameter; SVM requires two hyperparameters to *effectively* handle imbalance data. Experimental results comparing SVM and RSVM in the SL setting are given in subsection 6.2.

SL versus LPU Given that SL also has knowledge of negative examples, one expects that SL will perform better than LPU. Is this true? If so how much is the difference? Also, if the distribution of positive and negative examples in the test set differs from that in the SL train set, does SL suffer, say to the point of being even worse than LPU? One of the weaknesses of LPU is that, given the lack of negative examples, it doesn't have clear cues to set the classifier threshold θ. The proxy F Measure in (6) is what RSVM-LPU (RSVM in the LPU setting) use, but how good is it for choosing θ? These are questions worth answering, and are taken up in subsections 6.3 and 6.6.

Hyperparameter Optimization Note that there are three terms in (1); therefore, two hyperparameters will be needed under any reparametrization, unless one hyperparameter is set as *fixed constant* times the other hyperparameter. For example, we can also optimize $\frac{C_w}{2}||\mathbf{w}||^2 + \tilde{C}_P \sum_{i \in P} \xi_i + \sum_{i \in U} \xi_i$. In standard SVM, $\tilde{C}_P$=1 and equal importance is given to both positive and negative examples. However, it is important to have two hyperparameters when the class imbalance is high, which is the case in our setting with large number of unlabeled examples. Furthermore, there is some amount of noise present in the unlabeled examples (when treated as negative examples) since they can contain positive examples. Therefore, for achieving good performance, it is necessary to have two hyperparameters C_P and C_U in (1). In BSVM the unlabeled examples are treated as negative examples, and the solution is obtained by using any SVM solver as the core module and optimizing the hyperparameters C_P and C_U using the proxy F Measure(6) [18]. Two dimensional grid search to optimize the hyperparameters C_P and C_U makes BSVM computationally expensive.

In [9] where EN is proposed, there is no discussion of how the hyperparameters must be chosen; in the datasets used there, there is no class imbalance and $|P|$ is not small, and so default hyperparameters give good performance. In our experiments with EN we observed that when the percentage of positive labeled examples is small compared to the unlabeled set size, the performance loss due to the use of default hyperparameters as compared to *with hyperparameter optimization* can be quite high. Thus, like BSVM, optimizing the two hyperparameters C_P and C_U is necessary for EN too. In fact, the situation is worse with EN since it requires more parameters to be estimated. The additional parameters are: K; and the parameters in the probability model that converts the decision function score to the probability score ($p(s=1|\mathbf{x})$). These parameters are again estimated using cross-validation. Any estimation error in these parameters compounds the performance loss. In our implementation of EN we used the proxy F Measure for selecting the hyperparameters. Since EN is a probabilistic method, probabilistic measures (e.g., likelihood) may be worth exploring.

For RSVM we have only one hyperparameter C to be optimized and this reduces the computational complexity significantly. The following two observations are also relevant here. (i) The computational complexity of solving (5) may seem higher compared to solving a standard SVM problem due to pairwise comparison in the data fitting term. However, as pointed out at the end of subsection 3.2, (5) can actually be solved efficiently. (ii) Even though RSVM needs to optimize the threshold parameter θ separately, this step is not expensive because the CV data used for C tuning is also used for tuning θ (see steps 2 and 3 of Algorithm 3.1).

The experiments of subsection 6.4 give more insight on the LPU methods.

6. EXPERIMENTS

We conducted experiments on six benchmark datasets (details are given in Table 1) to answer the following five questions: (1) In a supervised learning (SL) setting how good is RSVM when compared to SVM? (2) How does LPU compare with SL, given the expectation that SL will do much better because it has a knowledge of negative training examples? (3) In the LPU setting how do BSVM, EN and RSVM compare and, how important is hyperparameter optimization? (4) How effective is the AUC based LPU feature selection method? and (5) How good is the proxy F Measure when it comes to setting the classifier threshold correctly? In these experiments RSVM is used both in SL and LPU settings; so, to avoid confusion we will refer to RSVM in the LPU setting as RSVM-LPU and RSVM in the SL setting simply as RSVM. Throughout, classifier performance is measured in terms of F Measure. When the underlying problem is a multi-class problem (News20 and WebKB) the Macro F Measure (average of the F Measures of all One-vs-All binary classification problems) is used; for computational effort we report the average time over all these problems. While reporting training time, the time taken by the RSVM based method is considered as the reference and is taken as one unit; then the times taken by the other methods are given relative to this unit. The time is computed as the *total* training time taken by any given method

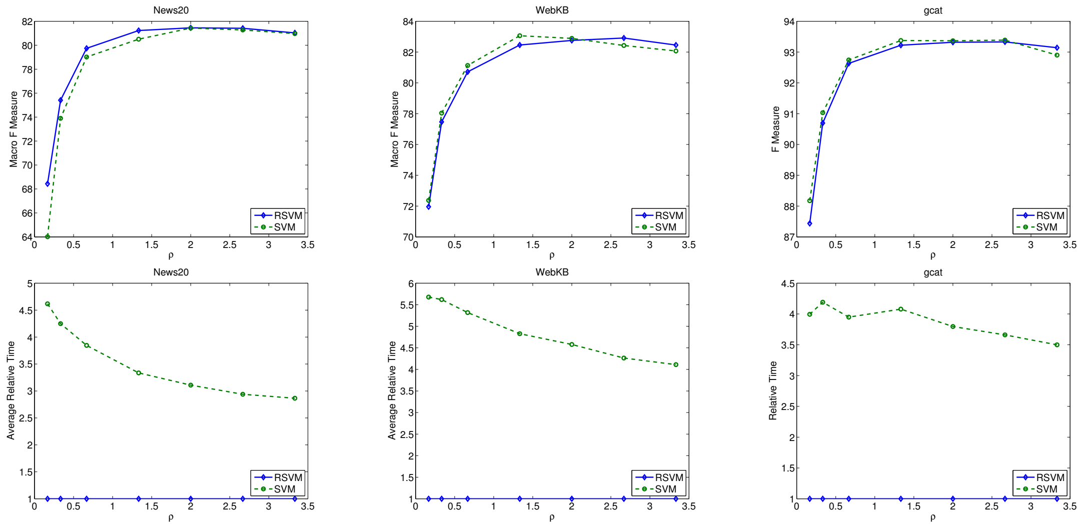


Figure 1: Macro F Measure and Training time (relative to RSVM) of RSVM and SVM on News20, WebKB and gcat for $X = 30$. ρ is as in (8).

as it searches over its grid of hyperparameter values. For each hyperparameter setting, 5-fold CV estimate of F Measure (for the SL setting) or proxy F Measure (for the LPU setting) is used. The search is done in two stages. In the first stage a coarse grid is used to narrow down the region around the optimal value. Then a finer grid search is carried out in the second stage. For BSVM and EN, the coarse grid is set to $C_U = \{0.0001, 0.001, 0.01, 0.1, 1, 10\}$ and $J = \{1, 3, 9, 27, 81, 243\}$; note that $C_P = J \cdot C_U$. In the SL setting we have C_N in place of C_U. In the case of RSVM it is set to $C = \{10^{-7}, 10^{-6}, \ldots, 10^{-2}\}$. The finer grid contained 6 values around the optimal value chosen from the coarser grid. For BSVM and EN we used *Liblinear* (`http://www.csie.ntu.edu.tw/~cjlin/liblinear`) since it uses state-of-the-art SVM solvers and is suited for text classification. We integrated the probability estimation module from libsvm [4] (`http://www.csie.ntu.edu.tw/~cjlin/libsvm`) with *Liblinear* to get the probability estimates for EN. For RSVM, we used the C implementation of [5] in our experiments. In the case of BSVM we used 5-fold CV based proxy F Measure estimate to choose the hyperparameters. In the case of EN method we used 5-fold CV based F Measure estimate since the classification problem is *labeled versus unlabeled* in the first stage (that is, estimating $p(s = 1|\mathbf{x})$). In all the experiments with random partitions average performance over 10 partitions is reported. Finally, due to space limitation we present plots only for some datasets and report observations on other datasets when some behavioral variations were observed.

6.1 Description of Datasets

The News20 [15] dataset contains text documents from 20 different newsgroups. We combined the preprocessed train and test sets obtained from [4] to form the whole dataset. Each newsgroup is used as the positive class and the rest as the negative class. This results in 20 One-vs-All binary classification problems. The WebKB [6] dataset is a web page classification problem where the web pages are collected from university websites and these web pages belong to one of 7 classes. For our experiments we considered only the classes 1, 3, 4 and 7. This is because the number of positive examples was very small (for example, to do meaningful cross-validation) for other classes. We considered 4 One-vs-All binary classification problems treating examples belonging to each of the 4 classes (1, 3, 4 and 7) as the positive class and the rest of the examples as negative class. The *adult* dataset is a binary classification problem. We combined the train and test splits of $a5a$ available from [4] to form the whole dataset. The *ccat* and *gcat* datasets are binary classification problems that classify articles into corporate versus non-corporate and government versus non-government articles respectively. The classes corporate and government are top-level categories in the RCV1 training dataset. The *realsim* dataset is also a binary classification problem and is obtained from a collection of UseNet articles from four discussion groups, for simulated auto racing, simulated aviation, real autos, and real aviation [21]. The *ccat*, *gcat* and *realsim* datasets are available at `http://people.cs.uchicago.edu/~vikass/svmlin.html`; see original source details there-in. These datasets cover a wide range of problems, class imbalance and n. The construction of train/test sets are explained for each experiment below in detail.

Table 1: Datasets Description: n_+ and n_- denote the number of positive and negative examples in the overall (train+test) dataset. $u(\%)$ denotes the percentage of positive examples $(\frac{100n_+}{n_+ + n_-})$. For News20 each of the 20 classes has close to 1000 examples; so the indicated values are approximate. For the WebKB dataset the class number is indicated in parentheses.

Dataset	n_+	n_-	$u(\%)$
News20	1000	18928	5
WebKB (1)	929	6506	12.5
WebKB (3)	1124	6311	15.1
WebKB (4)	3741	3694	50.32
WebKB (7)	1641	5794	22.07
adult	6414	26147	19.70
ccat	10786	12363	46.59
gcat	6970	16179	30.11
realsim	22238	50071	30.75

6.2 RSVM versus SVM

In this experiment we compare RSVM and SVM in the SL setting. Apart from a basic comparison, the intention is to study how well these methods perform under different train and test distribution conditions; this is important for web classification because, many a times there is lack of clarity in the space of negative examples and so it is often the case that the negative examples chosen for training do not represent that space well.

Experimental Setup We evaluate the performances under different scenarios of train and test distributions. We fix the test distribution and vary the train distribution. The train and test sets are constructed as follows. First we randomly split the dataset into two sets in a 80:20 proportion, while maintaining the same class imbal-

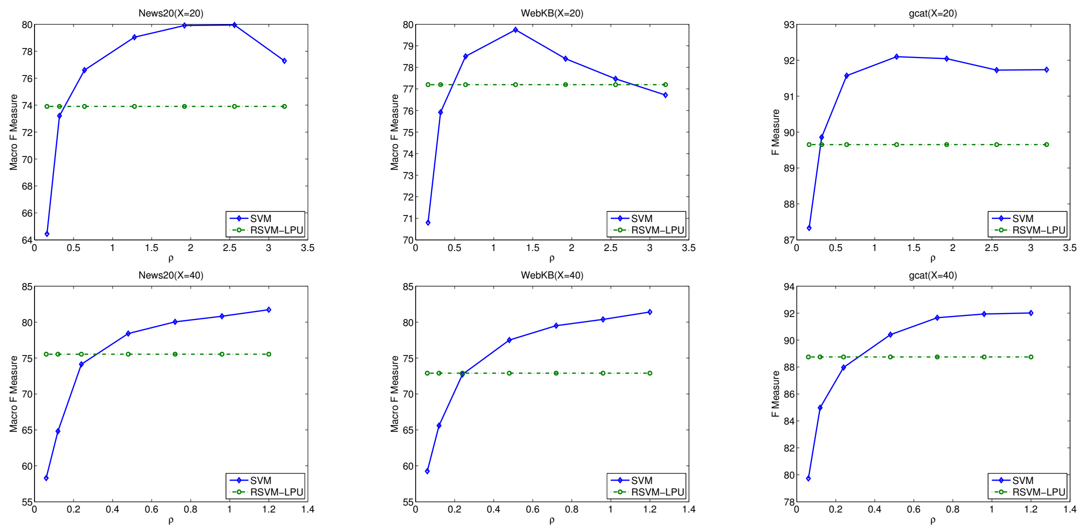


Figure 2: Macro F Measure of SVM and RSVM-LPU on News20, WebKB and gcat. ρ is as in (8). The top and bottom rows correspond to the cases, X=20 and 40 respectively.

ance ratio between the positive and negative examples in the two sets. Let us denote these sets as $\mathbf{S}$ and $\mathbf{T}$. Now keeping $\mathbf{T}$ as the test set, the train set is constructed from $\mathbf{S}$ as follows. We randomly select $X\%$ of positive examples from $\mathbf{S}$. Let us denote this set as $\mathbf{P}$. Now keeping $\mathbf{P}$ fixed, we randomly select the negative example set $\mathbf{N}$; to vary the train distribution we vary the size of $\mathbf{N}$ (denoted as $|\mathbf{N}|$). We conduct this experiment on 10 random partitions of negative examples for each size and repeat this experiment for 3 different sizes of $\mathbf{P}$ (denoted as $|\mathbf{P}|$). Since we did not find any noticeable behavioral changes for different $|\mathbf{P}|$ we report results only for one $|\mathbf{P}|$ value (corresponding to X=30). The results are given in figure 1. In the figure

$$\rho = r(\text{test})/r(\text{train}) \qquad (8)$$

where $r(A)$ is the class ratio in A, i.e., the ratio of the number of positive examples and the number of negative examples in set A. Thus when the class ratios in the train and test distributions are the same we have $\rho = 1$.

Observations The F Measures of RSVM and SVM are very close. One key observation is that the performances of both the methods drop rapidly as ρ falls below 1; note that this corresponds to the situation where there are relatively lesser negative examples in the train distribution as compared to the test distribution. For very small values of ρ, we also observed that the performance of RSVM was better, particularly on the News20 and WebKB datasets as X decreased. The performances of both the methods improve for $\rho \geq 1$ (the situation where there are relatively more negative training examples) and their performances are very close. The results suggest that an insufficient number of negative training examples ($\rho < 1$) (with respect to the test set distribution) has significant impact on performance. RSVM is significantly faster (>3 times). In fact, on the adult dataset the speed-up was around 10 times; on this dataset the SVM classifier chooses a high value of C_P and C_N for some of the folds, resulting in larger training times. Note that RSVM is faster only when two hyperparameters are needed in the standard SVM (i.e., when the class imbalance is high).

6.3 Supervised Learning versus LPU

To represent SL we choose SVM since it is more well known standard; also, in section 6.2 we saw that its performance is close to that of RSVM. For LPU we choose RSVM-LPU to represent it. Our intention is to see how well RSVM-LPU performs in comparison to SVM and see if there are any specific conditions under which it even performs better.

Experimental Setup We construct the train and test sets as follows. For the LPU setting, we first randomly pick $X\%$ of positive examples from the overall dataset. Let us denote this dataset as $\mathbf{P}$. Then we split the remaining examples in 80:20 unlabeled-test set proportion. Let us denote this split as $\mathbf{U}$ and $\mathbf{T}$. The main motivation behind this construction is that the unlabeled set is collected in such a way as to represent the test set in practice. Therefore *the same class ratio is maintained in the unlabeled and test sets*. Thus $(\mathbf{P}, \mathbf{U})$ form the training set and $\mathbf{T}$ forms the test set. In the SL setting, we randomly sample negative examples $\mathbf{N}_{sl}$ from the negative example set $\mathbf{N}_u$ in $\mathbf{U}$. Thus, $(\mathbf{P}, \mathbf{N}_{sl})$ form the training set and $\mathbf{T}$ again forms the test set. We conduct this experiment on 10 random partitions for different sizes of $\mathbf{N}_{sl}$ and repeat this experiment on three different sizes of $\mathbf{P}$ (corresponding to X=20,30 and 40). The F Measure values corresponding to the cases X=20 and 40 are given in figure 2. The performance results corresponding to X=30 turned out to be closer to the case of X=40. Note that the performance in the LPU setting does not change as the ρ-axis reflects only the change in the number of negative training examples.

Observations Though unimportant for comparing RSVM-LPU and SVM, it is worth noting that different values of X correspond to test sets with different class ratios. Low values of ρ correspond to lesser number of negative examples. Interestingly, on all the datasets, RSVM-LPU performs significantly better when ρ is small. SVM improves as more negative examples are added and the performance almost stabilizes after some point when X=40. When the number of positive examples is less (X=20), there is some drop in the performance seen when large number of negative examples is added; of course the drop is not as high as the one observed when ρ is small. In general, the peak performance of SVM was observed approximately around ρ=1, which is expected since the distribution used for parameter tuning in training matches the distribution evaluated in testing. Overall, around $\rho = 1$ SVM does better than RSVM-LPU. While this improvement is significant on News20 and WebKB datasets it is not that significant (<3%) on other datasets like gcat, ccat and realsim.

6.4 Comparison of Various LPU Methods

In this section we present two sets of experiments. In the first experiment we demonstrate the need for hyperparameter optimization

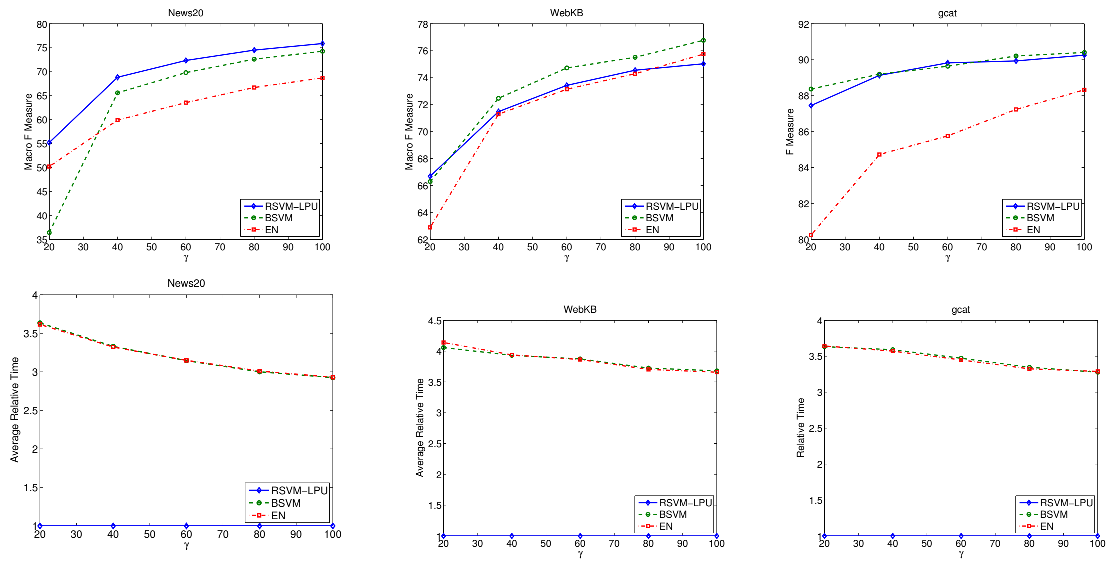


Figure 3: Macro F Measure and Training (relative) time of the LPU methods (RSVM, BSVM and EN) on News20, WebKB and gcat for X=30. γ is the percentage of positive labeled examples with reference to the available positive labeled examples $|P|$.

in LPU methods. In the second experiment we compare the performances of the proposed RSVM-LPU method with the BSVM and EN methods.

Experimental Setup The construction of the sets $\mathbf{P}$, $\mathbf{U}$ and $\mathbf{N}$ remains the same as in subsection 6.3. In this experiment we also study the performances of the LPU methods as the number of positive examples is increased. We construct the positive labeled set $\tilde{\mathbf{P}}$ by randomly sampling γ percentage of samples from $\mathbf{P}$. Thus $(\tilde{\mathbf{P}}, \mathbf{U})$ forms the training set and $\mathbf{T}$ forms the test set. Note that $|\tilde{\mathbf{P}}| = \frac{\gamma}{100}|\mathbf{P}|$ and $\gamma = 100$ corresponds to the case in subsection 6.3 (that is, $\mathbf{P} = \tilde{\mathbf{P}}$). We conducted this experiment on 10 random partitions of $\tilde{\mathbf{P}}$ and repeated this experiment for each size of $\tilde{\mathbf{P}}$. The results given in the figure are average F Measure values obtained from these partitions. We also varied $|\mathbf{P}|$ by varying $\mathbf{X}$ (corresponding to X=20,30 and 40) and since there were no changes in the behavioral patterns of methods (relative to one another) we only present the results corresponding to X=30.

Observations on Hyperparameter Optimization Hyperparameter tuning turns out to be important for all three LPU methods, viz. BSVM, EN and RSVM-LPU. We take EN to illustrate this. We conducted an experiment to study the effect of hyperparameter optimization as γ is varied. Default hyperparameter values for EN are: $C_P = 1$ and $C_U = 1$. We observed significant performance improvements (more than 10% F Measure value in several cases) with hyperparameter optimization for EN. The performance difference with and without hyperparameter optimization was more at lower values of γ (that is, when the number of labeled positive examples is low).

Observations on RSVM-LPU, BSVM and EN Methods In this experiment we performed hyperparameter optimization for all the methods. Since no specific behavioral differences were seen for different X values, we give results only for X=30; see figure 3. All the methods improve their F Measure as γ increases. At lower γ values, the F Measure values of all the methods fall relatively sharply. This is both due to the paucity in (positive) labeled examples as well as the fact that this paucity causes CV based hyperparameter tuning to become inferior.

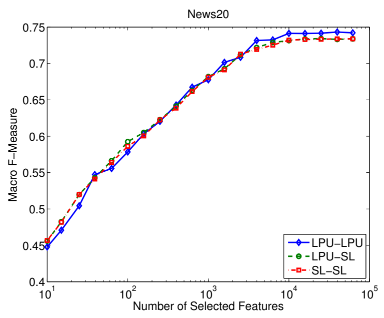
Overall RSVM-LPU and BSVM are quite close in classifier performance. There are exceptions too: on News20 RSVM-LPU is significantly better and, on WebKB BSVM is significantly better. EN is generally inferior, sometimes quite badly (see, for example, the results in figure 3 for News20 and gcat). We believe that the inferior performance of EN is due to inaccurate estimates of both K and $p(s = 1|\mathbf{x})$, and this affects the final probability score: $p(y = 1|\mathbf{x}) = p(s = 1|\mathbf{x})/K$; refer to sections 2 and 5 for details. In general, we observed that the performance comes closer to other methods as γ increases and this happens due to improved parameter estimates.

The training times for BSVM and EN are close; RSVM-LPU is significantly faster than them. As can be seen from figure 3, the speed-up is more than 3 times; on the adult dataset it was about 10 times. On the adult dataset, the BSVM and EN methods picked large C_U as the optimal value on some of the folds, resulting in longer training time; on the other hand the RSVM-LPU method picked the hyperparameter value more consistently.

6.5 Feature Selection in the LPU Setting

We did the feature selection experiment on WebKB and News20 in both LPU and SL settings with RSVM. We note that the feature selection technique proposed by Calvo et al [3] cannot be *directly* compared with our method for the reasons explained earlier (see section 5). For reference we present results from SL setting using the proposed AUC based feature selection technique. Note that our intention here *is not to compare* AUC based feature selection method with other feature selection methods in supervised setting, a study that can be taken up separately.

Experimental Setup The experimental setup is essentially same as the one described in section 6.4. We set aside 20% of positive examples (X=20) and set $\gamma = 100$. In the SL setting we have both positive and negative labeled examples, and we set $\rho = 1$. We fixed C at $5 \cdot 10^{-4}$. We ranked the features using the AUC score of each individual feature observed in the training set as discussed in Section 4. We evaluated the Macro F Measure on the test set as a function of number of selected features. To assess the quality of features selected in the LPU setting, we can also evaluate the performance of SL classifier built with features selected in the LPU setting and compare it with the performance obtained using features selected in the SL setting. We considered the following variations of LPU-LPU, LPU-SL and SL-SL. The prefixes LPU and SL correspond to the cases where the features are selected from positive/unlabeled and positive/negative examples respectively. The suffixes LPU and



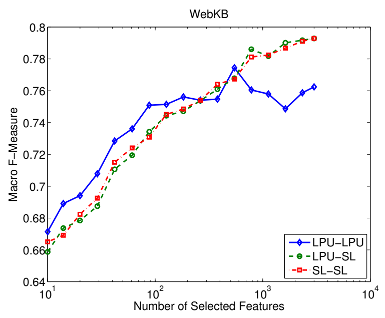
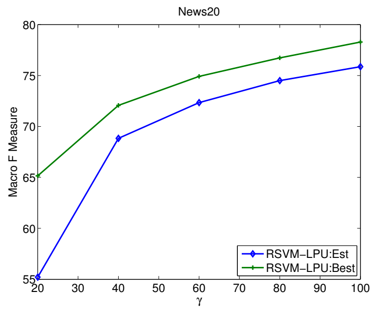


Figure 4: Macro F Measure of RSVM classifier in both SL and LPU settings with the proposed AUC based feature selection technique on News20 and WebKB. The prefixes LPU and SL correspond to the selection of features using positive/unlabeled and positive/negative examples respectively. The suffixes LPU and SL correspond to the classifier being built in the LPU and SL setting respectively.



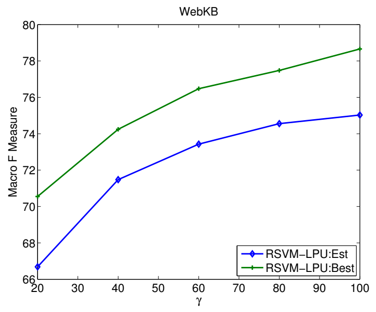


Figure 5: Macro F Measure of RSVM in LPU setting with estimated and best threshold values as a function of γ for X=30 on News20 and WebKB; definition of γ is as in the caption of Figure 3.

SL correspond to the cases where the classifier is built in the LPU and SL setting respectively.

Observations The results corresponding to these variations are shown in figure 4. On News20, the performances of all variations are very close. On WebKB, it is observed that the LPU classifier performance is slightly better than the SL classifier when the number of features is small. The SL classifier starts performing better after a sufficient number of features have been added. On both the datasets, the usefulness of the proposed technique is clearly seen: even with an order of magnitude reduction in the number of features the loss in performance is within 2-3%. Also, the SL classifier performance with LPU selected features is almost same as the performance achieved with SL selected features.

6.6 Estimated Threshold vs Best Threshold

In this experiment we compare the performances of RSVM classifier in the LPU setting with estimated and best thresholds. The experimental setup for the LPU setting remains the same as in subsection 6.4. Recall that in the LPU setting we estimate the threshold using 5-fold CV proxy F Measure. To assess the quality of the threshold estimate we compared the F Measure performances (as a function of γ) achievable by the estimated threshold and the best threshold on the test set. The results are given in figure 5. Note that the proxy F Measure estimate and hence the threshold estimate are expected to be poor at lower values of γ. This is because the number of examples available to perform 5-fold CV is very small and the proxy F Measure cannot be reliably estimated. This effect is more clearly seen in the News20 dataset as compared to the WebKB dataset. We believe this is due to the reason that the class ratio is lower in the News20 dataset as compared to the WebKB dataset. The results demonstrate that when γ is reasonably high, the performance difference is almost same and is within approximately 2-3%. Overall, the results in this subsection seem to indicate that further research to come up with a measure better than proxy F Measure for tuning threshold is worthwhile. Suppose one has a knowledge of the fraction of positive examples in the train/test distributions. How this can be effectively used to choose the threshold is worth investigating.

6.7 Summary

In this subsection we summarize the key observations from all the experiments discussed in the previous subsections.

- **RSVM vs SVM** The performances of both the methods are very close on the F Measure. RSVM is significantly faster than SVM due to easier one dimensional hyperparameter optimization than the two dimensional search needed with the SVM (when the class imbalance is high).
- **SL vs LPU** The performance of RSVM-LPU is better than SVM operating in the SL setting for small values of ρ. Overall there is performance improvement with the SL-setting around ρ=1 over the LPU setting. While this improvement is significant on News20 and WebKB it is not that significant ($<$3%) on other datasets.
- **Comparison of LPU Methods**
  - **Hyperparameter Optimization:** This is important for LPU methods. The performance improves a lot with hyperparameter optimization. As in the case of supervised setting optimization of hyperparameter is easier with RSVM-LPU compared to BSVM and EN.
  - **Training Time Comparison:** With hyperparameter optimization BSVM and EN take similar training time. RSVM-LPU is faster than BSVM and EN by more than 3-10 times.
  - **Performance Measures:** The performances of RSVM and BSVM are very close. While EN lags behind on some datasets at lower values of γ, it catches up as γ increases; we believe that the lagging behavior at lower γ is due to inaccurate probability estimates obtained.

- **Feature Selection:** The proposed AUC based feature selection technique in the LPU setting works quite well; an order of magnitude reduction in the number of features is achieved without significant degradation from the peak performance. The feature selection from the LPU setting as tested in the SL setting is interestingly *as good as* the features selected using supervised learning data.

- **Threshold Setting:** When γ is reasonably large the performance degradation with the estimated threshold value is typically within 2-3% of the performance obtained with the best threshold.

7. CONCLUSION

In this paper we proposed a pairwise ranking based SVM (RSVM) method to build classifier models from positive and unlabeled examples. We build a ranking model by encouraging the positive examples to score higher than the unlabeled examples. Then the final classifier is built by estimating a threshold parameter. The proposed method is fast and also easy to implement via publicly available RSVM codes. We also proposed an AUC based feature selection technique for the LPU setting and demonstrated its usefulness. We conducted comprehensive experiments with various methods in both SL and LPU settings on several benchmark datasets and made several important observations. These experiments show that RSVM-LPU is worthy of inclusion in the repertoire of good methods for solving binary classification problems in web applications.

Acknowledgments: The authors are thankful to the anonymous reviewers for their helpful comments.

8. REFERENCES

[1] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *ICML '05: Proceedings of the Twenty Second International Conference on Machine Learning*, 2005.

[2] B. Calvo, P. Larranaga, and J. A. Lozano. Learning Bayesian classifiers from positive and unlabeled examples. *Pattern Recognition Letters*, 28:2375–2384, 2007.

[3] B. Calvo, P. Larranaga, and J. A. Lozano. Feature subset selection from positive and unlabeled examples. *Pattern Recognition Letters*, 30:1027–1036, 2009.

[4] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[5] O. Chapelle. *Software for Rank SVM*. {`http://olivier.chapelle.cc/primal/ranksvm.m`}.

[6] O. Chapelle and S. S. Keerthi. Multi-class feature selection with support vector machines. In *Proceedings of the American Statistical Association*, 2008.

[7] O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with SVMs. *To appear in Information Retrieval Journal, Special Issue on Learning to Rank*, 2009.

[8] F. Denis, R. Gilleron, and M. Tommasi. Text classification classifiers from positive and unlabeled examples. In *The 9th International Conf. Information Processing and Management of Uncertainty in Knowledge Based Systems*, pages 1927–1934, 2002.

[9] C. Elkan and K. Noto. 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*, pages 213–220, New York, NY, USA, 2008. ACM.

[10] G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, pages 1289–1306, 2003.

[11] Y. Freund, R. Iyer, R. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

[12] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, pages 115–132. MIT Press.

[13] T. Joachims. A support vector method for multivariate performance measures. In *ICML '05: Processing of International conference on Machine learning*, pages 377–384, New York, NY, USA, 2005. ACM.

[14] T. Joachims. Training linear SVMs in linear time. In *KDD '06: Processing of Knowledge Discovery and Data Mining*. ACM, 2006. Software available at {`http://www.cs.cornell.edu/People/tj/svm_light/svm_rank.html`}.

[15] K. Lang. Newsweeder: Learning to filter netnews. In *ML '95: Proceedings of the 12th International Machine Learning Conference*, pages 331–339, 1995.

[16] W. S. Lee and B. Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *ICML '03: Proceedings of the Twentieth International Conference on Machine Learning*, pages 448–455, 2003.

[17] X. Li and B. Liu. Learning to classify texts using positive and unlabeled data. In *Proceedings of International Joint Conferences on Artificial Intelligence*, pages 587–594, 2003.

[18] B. Liu, Y. Dai, X. Li, W. S. Lee, and P. S. Yu. Building text classifiers using positive and unlabeled examples. In *ICDM '03: Proceedings of the Third IEEE International Conference on Data Mining*, pages 179–186. IEEE Computer Society, 2003.

[19] B. Liu, W. S. Lee, P. S. Yu, and X. Li. Partially supervised classification of text documents. In *ICML '02: Proceedings of the Nineteenth International Conference on Machine Learning*, pages 387–394, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[20] L. M. Manevitz, M. Yousef, N. Cristianini, J. Shawe-taylor, and B. Williamson. One-class SVMs for document classification. *Journal of Machine Learning Research*, 2:139–154, 2001.

[21] V. Sindhwani and S. S. Keerthi. Large scale semi-supervised linear SVMs. Technical report, 2006. available at `http://www.keerthis.com/semisup_techreport_06.ps`.

[22] X. Wu, R. K. Srihari, and Z. Zheng. Document representation for one-class SVM. In *Proceedings of European Conference on Machine Learning*, pages 489–500, 2004.

[23] H. Yu, J. Han, and K. C.-C. Chang. PEBL: Web page classification without negative examples. *IEEE Transactions on Knowledge and Data Engineering*, 16:70–81, 2004.

[24] H. Yu, C. Zhai, and J. Han. Text classification classifiers from positive and unlabeled documents. In *Proc. 12th International Conf. Information and Knowledge Management*, pages 232–239, 2003.

[25] Z. Zheng, X. Wu, and R. Srihari. Feature selection for text categorization on imbalanced data. In *ACM KDD Explorations Newsletter*, pages 80–89, 2004.