Classifying Accurately from Positive and Unlabeled Examples Using Marginal Property of SVM

Hwanjo Yu
Department of Computer Science, University of Illinois, Urbana-Champaign, IL 61801 USA

Sylvian R. Ray
Department of Computer Science, University of Illinois, Urbana-Champaign, IL 61801 USA

Abstract

In many learning problems, positive examples are available with unlabeled data whereas labeled examples are rare or expensive. We present a new algorithm, Mapping-Convergence (M-C), that learns from positive and unlabeled examples as accurately as the traditional SVM (TSVM) that learns from labeled examples. Our experiments show that when the M-C uses the same amount of positive as that of traditional SVM (TSVM), the M-C achieves classification accuracy as high as that of the TSVM, without needing negative examples that are hard to acquire in many practical problems.

1. Introduction

Typical classification (or machine learning) algorithms use only labeled examples (or data). However, people have also tried to use unlabeled data in many classification problems, motivated by the fact that, in many problem domains, it is easy to acquire large sets of unlabeled examples. One sub-area of using unlabeled data in machine learning is called semi-supervised learning – learning from small set of labeled and large set of unlabeled examples. Another one is learning from positive and unlabeled examples, which is motivated by the argument that, in many learning problems, positive examples are available with unlabeled data whereas labeled examples are rare or expensive (DeComite et al., 1999). For example, consider the automatic diagnosis of diseases: unlabeled data are easy to collect (all patients in the database), and positive data are also available (the patients who have the disease), but labeled data are expensive if detection tests for the disease are expensive. We supplement the above argument with the fact that, especially collecting negative training data is hard in many problem domains because (1) negative training data should represent uniformly the universal set excluding the positive class, and (2) manually collected negative training examples could be easily biased because of human’s unrecognizable stereotype, which results in causing bad effect on the classification performance. For example, consider "homepage" classifier in web page classification. Positive training examples are sample of homepages. Negative training examples are sample of non-homepages that should represent the Internet uniformly excluding the homepage, which involves laborious manual classification to collect.

We present a new algorithm that learns from positive and unlabeled data as accurately as the traditional algorithms that learn from labeled data. We assume that unlabeled data are not biased, which is a reasonable assumption since, in most domains of practical problems, the universal set supports random sampling on it. The M-C algorithm uses SVM (Support Vector Machine) technology (Cortes & Vapnik, 1995), especially the marginal property of SVM that ensures the class boundary (hyperplane in the feature space) to converge to the one from labeled data (discussed in details in Section 4). We present the details of the marginal property of the SVM in Section 3.

Our experiments (Section 5) are explored in web page classification because of its high demands. We did the experiments in two different domains of universal sets: one is the Internet (Experiment 1), and the other is computer science department sites (Experiment 2). Both experiments empirically affirmed the practicality of our approach.

2. Related Work

There have been many researches to use unlabeled data for learning classification. Unlabeled data contain
information about the joint distribution over features other than the class label. Clustering techniques utilize the features of unlabeled data to identify natural clusters of the data. However, class labels do not always correspond to the natural cluster of data. When unlabeled data are used with a sample of labeled data, it increases classification accuracy under certain environments, which is called semi-supervised learning. EM algorithm is a representative algorithm of semi-supervised learning or unsupervised learning (Dempster et al., 1977). However, the result depends on the critical assumption that the data are generated using the same parametric model as used in classification. Kamal Nigam inserted two parameters into EM (to relax the generative assumptions): one for controlling the contributions of labeled and unlabeled data, and the other for controlling the number of mixture components corresponding to one class (Nigam, 2000). Transductive SVM is also a kind of application of EM algorithm to SVM (Joachims, 1999b). With careful parameter setting, both work show good results under certain environments. However, when the number of labeled data is getting larger or the generative assumptions are violated badly, semi-supervised learning schemes degrade classification accuracy significantly.

Another stream of research to use unlabeled data in classification is learning from positive and unlabeled examples. This research is based on the fact that, in many real-world learning problems, numerous unlabeled and positive examples are usually available while labeled examples are rare and expensive (Letouze et al., 2000; DeComite et al., 1999). In 1998, F. Denis defined the PAC learning model for positive and unlabeled examples, and showed that k-DNF and k-decision lists are learnable from positive and unlabeled examples theoretically (Denis, 1998). Since then, some experiments to learn from positive and unlabeled examples have been tried using decision list or decision trees (Letouze et al., 2000; DeComite et al., 1999). However, these works have some limitations: (1) Their experiments are based on decision list or tree which is not very tolerant at high dimensionality and sparse instance space. Many practical problems such as text or web page classification have those properties. (2) Their algorithms require the information of the proportion of positive examples for training, which is usually not available. (3) They also perform poorer than a traditional learning scheme with labeled examples when the number of labeled data is large enough. The M-C (Mapping-Convergence) algorithm, which we introduce in this paper, relaxes the above three limitations from positive and unlabeled examples. Our approach is fundamentally different from previous approaches. The first part of the M-C is based on k-DNF, which was proved to be learnable previously (Denis, 1998). By using k-DNF, the M-C maps initial class boundary. After that, the M-C converges the class boundary to the one from labeled data by using SVM (Support Vector Machine) technology (Cortes & Vapnik, 1995).

People have recently researched one-class classification, which distinguish one class of data from the rest of the feature space under the assumption that only examples of the target class are available (Tax & Duin, 2001; Manevitz & Yousef, 2001). Since their approach utilizes only one set of data, only one side of the boundary is supported. Therefore, the performance is very dependent on user parameters indicating how strictly the boundary should fit around the data in each of the feature directions. From our experiments of an one-class SVM using the LIBSVM (version 2.33)\(^1\), it showed very poor results with the standard parameters of the algorithm, and even with careful parameter setting, the performance is much worse than our M-C algorithm since it does not utilize the distribution of unlabeled data.

3. Marginal Property of SVM

The SVM is a binary classification algorithm becoming increasingly popular because it has shown outstanding performance in many domains of classification problems (Dumais & Chen, 2000; Joachims, 1998; Yang & Lui, 1999). Especially it tolerates the problem of high dimensions and sparse instance spaces, which are the properties of many real-world problems.

SVM provides several salient properties that other learning algorithms do not have such as maximization of margin and nonlinear transformation of input space to feature space using kernel methods (Cortes & Vapnik, 1995). Consider its simplest form, a linear SVM. A linear SVM is a hyperplane that separates a set of positive examples from a set of negative examples with maximum margin in feature space. The margin (M) here indicates the distance from the hyperplane to the nearest of the positive and negative examples in the feature space. The distance from the hyperplane to an example is determined by the strength of each feature of the example. For instance, consider a resume page classifier in web page classification. If a page has many strong features related to the class of resume e.g. words "resume" or "objective" in headings), the page would belong to positive class (resume

\(^1\)The LIBSVM 2.33 is available at [http://www.csie.ntu.edu.tw/~cjlin/libsvm.](http://www.csie.ntu.edu.tw/~cjlin/libsvm)
class) in the feature space, and the location of the example would be far from the class boundary (the hyperplane).

![Graphical representation of a linear SVM of a two-dimension case](image)

**Figure 1.** Graphical representation of a linear SVM of a two-dimension case. Only two features are considered. M is the distance from the separator to the support vectors in feature space. Figure 1 shows an example of a simple two-dimensional problem that is linearly separable. In cases where points are not linearly separable, SVM has a parameter, C (the penalty imposed on training examples that fall on the wrong side of the decision boundary). The SVM basically computes the hyperplane that maximizes the distances to support vectors for a given parameter setting.

For problems that are not linearly separable, advanced kernel methods can be used to transform the nonlinear input space to linear feature space. We use the linear kernel in our experiments since, in the domain of web page classification that our experiments are based on, the linear kernel shows good performance and advanced kernels tend to overfit the concept easily. The choice of kernels for different problem domains is one of the future work.

4. Mapping-Convergence (M-C) Algorithm

We now proceed to the main topic of this paper: how to achieve classification accuracy (from positive and unlabeled examples) as high as that from labeled (positive and unbiased negative) examples. Our approach to achieve it is to use the Mapping-Convergence (M-C) algorithm that we discuss in this section. We first define several concepts, and then present the details of the M-C algorithm.

4.1 Definitions

We start the algorithm by using J-DNF to map the obvious negative examples that are far from the positive (POS) in universal set. We call those obvious negative examples **strong negatives**. For instance, consider a class of resume page in web page classification. Assume that there are two negatives (non-resume pages) in the feature space: one is "how to write resume" page and the other is "how to write article" page. The article writing page is assumed to be located farther than the resume writing page from the resume class in the feature space because the resume writing page may have more features related to resume (e.g. words "resume" in text) though it is not an actual resume page. The following definition quantizes the level of the strength of negative into a discrete levels without concern for the exact boundary of each level.

**Definition 1.** (Strength of Negative) The map of **strongest negative**, \( M_1(neg) \), is farthest from POS (the positive) in the feature space of universal set, \( U \). A map of negatives, \( M_i(neg) \), is farther than \( M_{i+1}(neg) \) from POS. The map of **weakest negative**, \( M_n(neg) \), is nearest to POS.

Figure 2 visualizes the strength of negative.

![Strength of negative](image)

\[ \sum_{i=1}^{n} M_i(neg) \] is equivalent to the set of the unbiased negative, something that excludes positives in universal set.

We say that the universal set excluding the strongest negatives \( (M_1(neg)) \) **maximally subsumes** POS (the positive), and say that POS plus the nearest negatives \( (M_n(neg)) \) **minimally subsume** POS. We call \( S_1(pos) \) the set that maximally subsumes POS, and call \( S_{n-1}(pos) \) the set that minimally subsumes POS. The following definition formalizes these concept.

**Definition 2.** (Subsumption of Positive) A map of maximally subsuming POS, \( S_1(pos) \), contains POS and \( \sum_{k=1}^{n} M_k(neg) \). A map of POS, \( S_i(pos) \), contains POS and \( \sum_{k=i+1}^{n} M_k(neg) \). A map which minimally subsumes POS, \( S_{n-1}(pos) \), contains POS and the weakest negative, \( M_n(neg) \). The following induced formulae help to understand these concepts.
\[ S_{i-1}(pos) = POS + M_i(neg) \]
\[ S_i(pos) = S_{i+1}(pos) + M_{i+1}(neg) \]
\[ S_1(pos) = S_2(pos) + M_2(neg) \]
\[ U = S_1(pos) + M_1(neg) \]

4.2 Mapping-Convergence (M-C) Algorithm

Our approach to achieve high accuracy from positive and unlabeled examples is to induce unbiased negative training examples (NEG) from samples of the universal set \( (U) \) using the given positive \( (POS) \) and \( U \). We do this in two steps. The first step is to map the strongest negative, \( M_1(neg) \), from \( U \) using 1-DNF without concern for the quality of mapping. We call this stage the mapping stage. The other, excluding the strongest negative \( (M_1(neg)) \) from \( U \), would be \( S_1(pos) \) that maximally subsumes \( POS \) as defined above. k-DNF is proved to be learnable from positive and unlabeled data in previous work (Denis, 1998), which supports the feasibility of the mapping stage when not concerning the quality very much. The goal of the mapping stage in the algorithm is to ensure that \( M_1(neg) \) does not include any positives in \( U \) not matter how small the amount of negatives that it maps. The quality of the mapping is not critical to the performance of the M-C algorithm. We briefly discuss the 1-DNF mapping in Section 4.3.

The second step is to train SVM repeatedly to aggregate mapped negatives \( (M_1(neg)) \) as close as possible to the unbiased negatives \( (NEG) \) – we call this stage convergence stage. We illustrate this through the following example.

```
Input:
- positive training examples, POS
- unlabeled examples (sample of universal set), U

Output:
- unbiased negative examples, NEG, and the SVM trained from POS and the NEG

Algorithm:
DL := construct_DL(POS, U);
M_1(neg), S_1(pos) := DL.classify(U);
/* classify U into the strongest negative(M_1(neg)) and maximally subsuming positive(S_1(pos)). */
NEG := \( \emptyset \), i := 1;
do {
  NEG := NEG + M_i(neg);
  SVM := construct_SVM(POS, NEG);
  M_i+1(neg), S_i+1(pos) := SVM.classify(S_i(pos));
  /* classify S_i(pos) into M_i+1(neg) and S_i+1(pos). */
  i := i + 1;
} while \( (M_i(neg) \neq \emptyset) \);
return NEG, SVM;
```

Figure 4. Mapping-Convergence (M-C) Algorithm

Example 1. (Convergence) Consider classifying “faculty web pages” (positive) in a university site (universal set). POS is a given sample of faculty (positive) pages. \( U \) is a sample of the university site (universal set). NEG is a set of other pages in the university site excluding faculty pages (unbiased negative pages that we target to collect). Assume that the university site has NEG with four different levels (n=4) of negative strength as following:\footnote{In real cases, the boundaries between levels may not be clear, but we assume that there exist continuous levels measured by strength of negative.}

- staff pages: \( M_4(neg) \), the weakest negative pages considered most similar to "faculty pages"
- student pages: \( M_3(neg) \)
- project or course pages: \( M_2(neg) \)
- information or facility pages: \( M_1(neg) \), the strongest negative pages considered most disparate to “faculty pages”

Next, consider how we collect NEG(unbiased negative examples) from \( U \)(sample of universal set). Let’s say 1-DNF identifies only the strongest negative, sample of information or facility pages \( (M_1(neg)) \). (The 1-DNF algorithm is briefly discussed in Section 4.3.) We save the \( M_1(neg) \) into NEG, and the other will be \( S_1(pos) \) containing all other pages except \( M_1(neg) \). \( S_1(pos) \) maximally subsumes the positive in \( U \). We train a SVM with the NEG (currently containing faculty pages\( (M_1(neg)) \)) and given POS. The SVM generates a hyperplane between NEG and POS, which keeps maximal margin between them in the feature space. When, using the SVM, we test \( S_1(pos) \) that contains all other pages except \( M_1(neg) \), the SVM divides the \( S_1(pos) \) into \( M_2(neg) \) (e.g. project or course pages) and \( S_2(pos) \) (e.g. the others containing student, staff, and faculty pages) (See the right most diagram of Figure 3.(a)). We now accumulate the \( M_2(neg) \) into NEG, and then we re-train the SVM with the NEG (currently containing \( M_1(neg) \)(information or facility pages) and \( M_2(neg) \)(project or course pages)) and given POS. The SVM generates another hyperplane between NEG and POS also keeping maximal margin between them in the feature space (See Figure 3.(b)). When, using the SVM, we test \( S_2(pos) \) that contains faculty, staff, and student pages, the SVM divides the \( S_2(pos) \) into \( M_3(neg) \)(e.g. student pages) and \( S_3(pos) \)(e.g. staff and faculty pages). We now accumulate the \( M_3(neg) \) into NEG again, and re-train the SVM with the NEG(currently containing \( M_1(neg), M_2(neg), \) and \( M_3(neg) \)) and given POS. We iterate these processes until the \( M_i(neg) \) becomes empty set. Figure 4 shows an outline of the M-C algorithm.
Example 1 may discretize too much the boundary of each level of strength of negative. However, in real cases with continuous levels, as you see in Figure 3, the marginal property of SVM urges the hyperplane to converge into the real boundary of the two classes (positive and negative classes) in the feature space. Our experiments in Section 5 show that the M-C algorithm from positive and unlabeled data actually performs as well as the traditional SVM from labeled data.

4.3 1-DNF Mapping

The goal of mapping stage is to subsume positive regardless of the quality of mapping as we discussed in Section 4. We do this by building a disjunction list (DL) of positive features, which is equivalent to 1-DNF. After we construct the DL (disjunction list) of positive features, we map the maximally subsuming positive ($S_1(pos)$) and strongest negative ($M_1(neg)$) by filtering the universal set, $U$, into the DL. We have explored many heuristics to build the DL. The choice of DL or $k$-DNF algorithm is, however, not critical to the performance of the M-C as long as the $S_1(pos)$ subsumes POS, because the M-C will converge $\sum_i M_i(neg)$ into NEG (unbiased negatives) regardless of the quality of the initial mapping, $M_1(neg)$. The details of constructing the DL and reasoning for each heuristic in the mapping algorithm will be presented in a technical report.

5. Experimental Results

In this section, we provide empirical evidence that our learning framework using positive and unlabeled examples performs as well as the traditional SVM using manually labeled (positive and unbiased negative) examples. We present experimental results with two different domains of universal sets: the Internet (Experiment 1), and university computer science department (Experiment 2).

5.1 Data sets and Experimental Methodology

**Experiment 1.** (The Internet) The first universal set in our experiments is the Internet. To collect random samples of internet pages, we used DMOZ, which is a free open directory of the Web containing hundreds of millions of web pages. Random sampling of a search engine database such as DMOZ is sufficient (we assume) to construct an unbiased sample of the Internet. We randomly selected 2388 pages from DMOZ to collect unbiased unlabeled examples. We also manually collected 368 personal homepages, 192 college admission pages, and 188 resume pages to classify the three

---

3Open Directory Project, [http://dmoz.org](http://dmoz.org)
interesting classes: personal homepages, college admission pages, and resume pages. (Each class is classified independently.) We used around half of the pages of each class for training and another half for testing. For testing negative data (for evaluating the classifier), we manually collected 449 non-homepages, 450 non-admission pages, and 533 non-resume pages. (We collected negative data just for evaluating the classifier we construct. Our framework does not require collecting negative data to construct classifiers.) For instance, for personal homepage class, we used 183 positive and 2388 unlabeled examples for training, and used 185 positive and 449 negative examples for testing.

Experiment 2. (University computer science department) The WebKB data set (Craven et al., 1998) contains 8282 web pages gathered from university computer science departments. The collection includes the entirety of computer science departments from various universities. The pages are divided into seven categories: student, project, faculty, course, staff, department and other. In our experiments, we classify the three most popular categories (from all independently): student, project, faculty. The number of the pages in each category is 1641, 504, and 1124 respectively. We randomly selected 1052 and 589 student pages, 339 and 165 project pages, and 741 and 383 faculty pages for training and testing respectively. For testing negative examples, we also randomly selected 662 non-student pages, 753 non-project pages, and 729 non-faculty pages. We picked up randomly 4093 pages from all categories to make a sample universal set, and same sample is used for four classes as unlabeled data. For instance, for faculty page classification, we used 741 positive and 4093 unlabeled examples for training, and used 383 positive and 729 negative example for testing.

We extracted features from different parts of a page – URL, title, headings, link, anchor-text, normal text, and meta tags. Each feature is a predicate indicating whether each term or special character appeared in each part. (e.g. ’~’ in URL, a word 'homepage' in title) We did not use stemming or a stoplist because it could hurt performance in web page classification. For example, a common stopword, "I" or "my", is a good indicator of a student homepage.

For SVM implementation, we used SVM light (Joachims, 1999a). As we discussed in Section 3, we used linear kernel method because it is efficient, and we found that advanced kernel methods (e.g. polynomial kernel) easily overfitted the concept and degraded the accuracy in web page classification. For the parameter, C (the penalty imposed on training examples that fall on the wrong side of the decision boundary), we used the default parameter, \[\text{C} = \frac{\text{argmax}}{x \in X} \cdot \text{argmin} \cdot \frac{1}{x \cdot x} \]. We didn’t rigorously try to find out the optimal C, because the default setting showed good performance in all cases. In many other learning algorithms, finding best parameters is usually critical to the performance. It is necessary for them to perform cross-validation to determine many problem-specific parameters, which is a time consuming and laborious manual process. Without it, they perform extremely poorly (sometimes, poorer than random). The strong theory of SVM (i.e. maximization of margin) makes it possible to run the M-C algorithm fully automatically without human interruption to determine best parameter setting for each iteration or each specific problem. We used the same parameter, C, for all our experiments, so the whole process can be done automatically and generally (not dependent on specific problem).

Result reports are based on precision-recall break even point (P-R), a standard measure for binary classification. Accuracy is not a good performance metric because very high accuracy can be achieved by always predicting the negative class. Precision and recall are defined as:

\[
\text{Precision} = \frac{\text{# of correct positive predictions}}{\text{# of positive predictions}}
\]

\[
\text{Recall} = \frac{\text{# of correct positive predictions}}{\text{# of positive examples}}
\]

The precision-recall break even point (P-R) is defined as the precision and recall value at which the two are equal. We adjusted the decision threshold b of the SVM at the end of each experiment to find P-R.

5.2 Results

Table 1. Precision-recall break even points (P-R) showing performance of M-C and TSVM (the traditional SVM trained from manually labeled data) in the two universal sets (U). The number of iterations to the convergence in M-C is shown in parentheses.

<table>
<thead>
<tr>
<th>U</th>
<th>CLASS</th>
<th>TSVM</th>
<th>M-C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>THE INTERNET</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>student</td>
<td>HOMEPAGE 88.11</td>
<td>86.49 (7)</td>
<td></td>
</tr>
<tr>
<td>project</td>
<td>ADMISSION 92.00</td>
<td>92.00 (4)</td>
<td></td>
</tr>
<tr>
<td>faculty</td>
<td>RESUME 96.2</td>
<td>96.2 (4)</td>
<td></td>
</tr>
<tr>
<td><strong>CS DEPARTMENT</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>student</td>
<td>PROJECT 94.74</td>
<td>94.23 (14)</td>
<td></td>
</tr>
<tr>
<td>project</td>
<td>FACULTY 86.67</td>
<td>86.06 (12)</td>
<td></td>
</tr>
<tr>
<td>faculty</td>
<td>92.95</td>
<td>91.12 (11)</td>
<td></td>
</tr>
</tbody>
</table>

We first show the performance comparison between
Figure 5. Convergence of negatives (NEG) and performance (P-R, precision-recall breakeven point) when the universal set is the Internet. TSVM indicates the traditional SVM constructed from manually labeled (positive and unbiased negative) data.

M-C and traditional SVM (trained from manually labeled data) on the six classes of the two universal sets – the Internet and CS department sites. We first constructed a SVM from positive (POS) and unlabeled data (U) using M-C. On the other hand, we manually classified the unlabeled data (U) to extract unbiased negatives from them, and then we built a traditional SVM from POS and those unbiased negatives. We tested same testing documents using those two SVMs – M-C and TSVM (Traditional SVM). Table 1 shows the P-R (precision-recall breakeven points) of each SVM, and it also shows the number of iterations to converge in the case of the M-C. In most cases, M-C without negative training data performs almost as well as the traditional SVM with manually labeled training data. For example, when we manually classify 109 resume pages and 2388 unbiased non-resume pages to train a SVM in a traditional way, it gives 96.2% P-R (precision-recall breakeven point). When we use M-C with only the 109 resume pages without non-resume pages, it gives also 96.2% P-R.

Figure 5 and 6 show the details of convergence (of the induced negative training data and corresponding P-R) at each iteration in the experiment of the universal set, the Internet and CS department sites respectively. The performance (P-R: precision-recall breakeven point) of M-C is converging to that of TSVM in all our experiments.

The P-R convergence graphs in Figure 6 show one more line (P-R of UN), which is the P-R when using the sample of universal set (U) as a substitute for negative training data. One might argue to use a sample of universal set itself as a substitute for negative training data since the portion of positive class in universal set (P(C)) is usually much smaller than the portion of its complement (P(-C)). However, when training SVM, small number of false positive training data affects significantly the set of support vectors which is critical to classification accuracy. Experiment 2 (CS department sites) shows that UN (using sample of universal set as a substitute for negative training data) degrades the performance significantly in classifying every class in the domain.

6. Conclusions

In many learning problems, positive examples are available with unlabeled data whereas labeled examples are rare or expensive. We present the Mapping-Convergence (M-C) algorithm that learns from positive and unlabeled examples as accurately as the traditional SVM that learns from labeled examples. Our experiments show that when the M-C uses same amount of positive as that of traditional SVM (TSVM), the M-C achieves classification accuracy as high as that of the TSVM, without needing negative examples that are hard to acquire in many practical problems. The
M-C algorithm adopting the SVM technology provides a very reliable solution for classification using positive and unlabeled examples.

References


