PEBL: Web Page Classification without Negative Examples

Hwanjo Yu, Jiawei Han, and Kevin Chen-Chuan Chang, *Member*, *IEEE*

Abstract—Web page classification is one of the essential techniques for Web mining because classifying Web pages of an interesting class is often the first step of mining the Web. However, constructing a classifier for an interesting class requires laborious preprocessing such as collecting positive and negative training examples. For instance, in order to construct a "homepage" classifier, one needs to collect a sample of homepages (positive examples) and a sample of nonhomepages (negative examples). In particular, *collecting negative training examples* requires arduous work and caution to avoid bias. This paper presents a framework, called *Positive Example Based Learning (PEBL)*, for Web page classification which eliminates the need for manually collecting negative training examples in preprocessing. The PEBL framework applies an algorithm, called *Mapping-Convergence (M-C)*, to achieve high classification accuracy (with positive and unlabeled data) as high as that of a traditional SVM (with positive and negative data). M-C runs in two stages: the *mapping stage* and *convergence stage*. In the mapping stage, the algorithm uses a weak classifier that draws an initial approximation of "strong" negative data. Based on the initial approximation, the convergence stage iteratively runs an internal classifier (e.g., SVM) which maximizes margins to progressively improve the approximation of negative data. Thus, the class boundary eventually converges to the true boundary of the positive class in the feature space. We present the M-C algorithm with supporting theoretical and experimental justifications. Our experiments show that, given the same set of positive examples, the M-C algorithm outperforms one-class SVMs, and it is almost as accurate as the traditional SVMs.

Index Terms—Web page classification, Web mining, document classification, single-class classification, Mapping-Convergence (M-C) algorithm, SVM (Support Vector Machine).

1 INTRODUCTION

AUTOMATIC categorization or classification[1] of Web pages have been studied extensively since the Internet has become a huge repository of information, in terms of both volume and variance. Given the fact that Web pages are based on loosely structured text, various statistical text learning algorithms have been applied to Web page classification.

While Web page classification has been actively studied, most previous approaches assume a multiclass framework, in contrast to the one-class binary classification problem that we focus on. These multiclass schemes (e.g., [1], [2]) define mutually exclusive classes a priori, train each class from training examples, and choose one best matching class for each testing data. However, mutual-exclusion between classes is often not a realistic assumption because a single page can usually fall into several categories. Moreover, such predefined classes usually do not match users' diverse and changing search targets.

Researchers have realized these problems and proposed the classifications of user-interesting classes such as "call for papers," "personal homepages," etc. [3]. This approach involves binary classification techniques that distinguish Web pages of a desired class from all others. This binary classifier is an essential component for Web mining because identifying Web pages of a particular class from the Internet is the first step of mining interesting data from the Web. A binary classifier is a basic component for building a type-specific engine [4] or a multiclass classification system [5], [6]. When binary classifiers are considered independently in a multiclass classification system, an item may fall into none, one, or more than one class, which relaxes the mutual-exclusion assumption between classes [7].

However, traditional binary classifiers for text or Web pages require laborious preprocessing to collect positive and negative training examples. For instance, in order to construct a "homepage" classifier, one needs to collect a sample of homepages (positive training examples) and a sample of nonhomepages (negative training examples). *Collecting negative training examples* is especially delicate and arduous because 1) negative training examples must uniformly represent the universal set excluding the positive class (e.g., sample of a nonhomepage should represent the Internet uniformly excluding the homepages), and 2) manually collected negative training examples could be biased because of human's unintentional prejudice, which could be detrimental to classification accuracy.

To eliminate the need for manually collecting negative training examples in the preprocessing, we proposed a framework, called *Positive Example Based Learning (PEBL)* [8]. Using a sample of the universal set as unlabeled data, PEBL learns from a set of *positive* data as well as a collection of *unlabeled data*. A traditional learning framework learns from *labeled data* which contains *manually classified*, both positive and negative examples. *Unlabeled data* indicates

1. Classification is distinguished from clustering in terms that classification requires a learning phase (training, and possibly, plus accuracy testing) before actual classification.

• *The authors are with the Department of Computer Science, University of Illinois, Urbana-Champaign, IL 61801.*
E-mail: {hwanjoyu, hanj, kcchang}@uiuc.edu.

Manuscript received 1 Sept. 2002; revised 1 Apr. 2003; accepted 10 Apr. 2003.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number 118553.

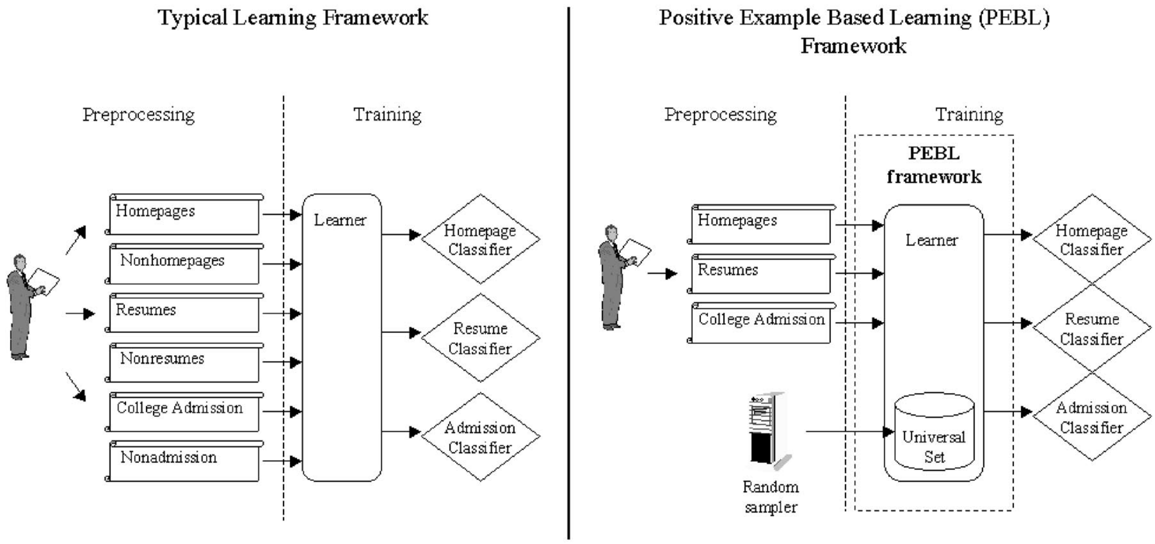

Fig. 1. A typical learning framework versus the Positive Example Based Learning (PEBL) framework. Once a sample of the universal set is collected in PEBL, the same sample is reused as unlabeled data for every class.

random samples of the universal set for which the class of each sample is arbitrary and uncorrelated. For example, samples of homepages and nonhomepages are *labeled* data because we know the class of the samples from manual classification, whereas random sampling of the Internet provides *unlabeled* data because the classes of the samples are unknown. In many real-world learning problems including Web page classification, unlabeled and positive, data are widely available whereas acquiring a reasonable sampling of the negative is impossible or expensive because the negative data set is just the complement of the positive one, and, thus, its probability distribution can hardly be approximated [8], [9], [10]. For example, consider the automatic diagnosis of diseases: Unlabeled data are easy to collect (all patients in the database), and positive data are also readily available (the patients who have the disease). However, negative data can be expensive to acquire; not all patients in the database can be assumed to be negative if they have not been tested for the disease, and such tests can be expensive.

Our goal is to achieve classification accuracy from positive and unlabeled data as high as that from fully labeled (positive and negative) data. Here, we only assume that the unlabeled data is unbiased. There are two main challenges in this approach: 1) collecting unbiased unlabeled data from a universal set which can be the entire Internet or any logical or physical domain of Web pages, and 2) achieving classification accuracy from positive and unlabeled data as high as that from labeled data. To address the first issue, we assume it is sufficient to use random sampling to collect unbiased unlabeled data. Random sampling can be done in most databases, warehouses, and search engine databases (e.g., DMOZ) or it can be done independently directly from the Internet.

In this paper, we focus on the second challenge, achieving classification accuracy as high as that from labeled data. The PEBL framework applies an algorithm, called *Mapping-Convergence (M-C)*, which uses the SVM (Support Vector Machine) techniques [11]. In particular, it leverages the marginal property of SVMs to ensure that the classification accuracy from positive and unlabeled data will converge to that from labeled data. We present the details of the SVM properties in Section 3.

Our experiments (Section 5) explore the classes within two different universal sets: The first universal set is the entire Internet (*Experiment 1*), and the second is computer science department sites (*Experiment 2*). Both experiments show that the PEBL framework is able to achieve classification accuracy as high as using fully labeled data.

One might argue that using a sample of universal set itself as an approximation for negative training data is sufficient since the portion of positive class in the universal set $(P(C))$ is usually much smaller than its complement $(P(\overline{C}))$. However, when training a SVM, a small number of false positive training data could be detrimental. Experiment 2 (i.e., CS department sites) shows that using samples of the universal set as a substitute for negative examples degrades accuracy significantly.

In summary, the contributions of our PEBL framework are the following:

1. Preprocessing for classifier construction requires collecting only positive examples, which speeds up the entire process of constructing classifiers and also opens a way to support example-based query on the Internet. Fig. 1 shows the difference between a typical learning framework and the PEBL framework for Web page classification. Once a sample of the universal set is collected in PEBL, the sample is reused as unlabeled data for every class, therefore, users would not need to resample the universal set each time they construct a new classifier.
2. PEBL achieves accuracy as high as that of a typical framework without loss of efficiency in testing. PEBL runs the M-C algorithm in the training phase to construct an accurate SVM from positive and unlabeled data. Once the SVM is constructed, classification performance in the testing phase will be the same as that of a typical SVM in terms of both accuracy and efficiency.

Note that this paper concentrates on the classification algorithms, but not on other related important problems for Web page classification, such as feature modeling and extraction. For instance, taking advantage of the structures of the documents or hyperlinks is also an important issue for Web page classification [12], [13], [14]. Moreover, selection of good features is critical to the classification accuracy regardless of the algorithms. However, these issues are beyond the scope of this paper. We consider a set of commonly used and clearly defined Web-based features of Web pages in our experiments, such as URL, head text, all text, hyperlink, and anchor text (Section 5).

While this paper focuses on algorithmic issues rather than applications, we note that the PEBL framework can enable many Web search and mining tasks that are essentially built on page classification. The PEBL framework, as a classifier that does not rely on negative labeled data, will be easily deployable, which makes it applicable in many practical applications. For instance, a *focused crawler* can traverse and mine the Web to discover pages of specific types (e.g., job announcements on the Web, in order to build a job database). Similarly, a *type-specific search engine* can use a page classifier to limit the search scope to only some target class (e.g., finding "C++" among only the job announcements). Further, our technique is also critical for *query by examples*, in which users give a few (positive) example pages to retrieval more in the same class. While such search methods will enable powerful queries beyond the current limitations of keyword queries, it is impractical and nonintuitive if users have to specify negative examples as well. Finally, although we specifically address Web page classification in this paper, our PEBL framework is applicable to classification problems in diverse domains (such as diagnosis of diseases or pattern recognition) with minor revisions. We discuss this further in Section 6.

The paper is organized as follows: Section 2 describes related work including the review of using unlabeled data in classification. Section 3 reviews the marginal properties of SVMs. Section 4 presents the M-C algorithm and provides theoretical justification. Section 5 reports the result of a systematic experimental comparison using two classification domains: the Internet and CS department sites. Section 6 outlines several important issues to consider regarding the learning algorithm and the PEBL framework. Finally, Section 7 reviews and concludes our discussion of the PEBL framework.

2 Related Work

Although traditional classification approaches use both fully-labeled positive and negative examples in classification, there are also approaches that use unlabeled data, which we discuss and contrast below.

How are unlabeled data useful when learning classification? Unlabeled data contains 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 clustering of data. When unlabeled data are used with a sample of labeled data, it increases classification accuracy in certain problem settings. Such techniques are called *semisupervised learning*. The EM algorithm is a representative algorithm which can be used for either semisupervised learning or unsupervised learning [15]. However, the result depends on the critical assumption that the data sets are generated using the same parametric model used in classification. Kamal Nigam inserted two parameters into EM (to relax the generative assumptions): one for controlling the contributions of labeled data and unlabeled data and the other for controlling the quantity of mixture components corresponding to one class [16]. Another semisupervised learning occurs when it is combined with SVMs to form transductive SVM [17]. With careful parameter setting, both of these works show good results in certain environments, e.g., with an extremely low amount of labeled data. When the number of labeled data grows or when the generative assumptions are violated, semisupervised learning schemes suffer significant degradation of classification accuracy.

Another line of research for using unlabeled data in classification is *learning from positive and unlabeled data*, often referred to as *single-class learning or classification*. Many works attempt rule-based or probability-based learning from positive or positive and unlabeled data [18], [19], [10], [9]. In 1998, Denis defined the PAC learning model for positive and unlabeled examples and showed that k-DNF (Disjunctive Normal Form) is learnable from positive and unlabeled examples [19]. After that, some experimental attempts [9], [10] have pursued using k-DNF or C4.5. However, these rule-based learning methods are often not applicable to Web page classification because:

1. they are not very tolerant with high dimensionality and sparse instance space, which are essential issues for Web page classification,
2. their algorithms require knowledge of the proportion of positive instances within the universal set, which is not available in many problem settings, and
3. they perform poorer than traditional learning schemes given sufficient labeled data.

Recently, a probabilistic method built upon the EM algorithm has been proposed for the text domain [18]. The method has several fundamental assumptions: the generative model assumption, the attribute independence assumption which results in linear separation, and the availability of prior probabilities. Our method does not require the prior probability of each class, and it can draw nonlinear boundaries using advanced SVM kernels.

The pattern recognition and verification fields have also explored various single-class classification methods, including neural network models [20], [21] and the SVMs [22], [23] (with increasing popularity). Some of these techniques tend to be domain specific: For instance, [23] uses SVM with only positive examples for face detection, however, it relies on using the face features of other nontarget classes as negative examples and, thus, is not generally applicable to other domains.

For document classification, Manevitz and Yousef [22] compared various single-class classification methods including neural network method, one-class SVM, nearest neighbor, naive Bayes, and Rocchio and concluded that one-class SVM and neural network methods were superior to all the other methods, and the two are comparable.

One-class SVMs (OSVMs), based on the strong mathematical foundation of SVM, distinguish one class of data

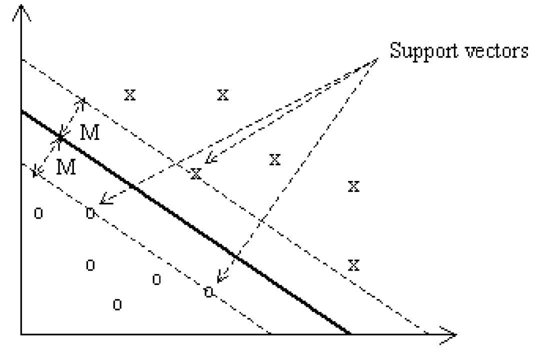

Fig. 2. A graphical representation of a linear SVM in a two-dimensional case. (i.e., Only two features are considered.) M is the distance from the separator to the support vectors in feature space.

from the rest of the feature space given only positive data sets [24], [22]. OSVMs draw the class boundary of the positive data set in the feature space. They have the same advantages of SVM such as scalability on the number of dimensions or nonlinear transformation of the input space to the feature space. However, due to lack of information about negative data distribution, they require a much larger amount of positive training data to induce an accurate boundary, and they also tend to easily overfit or underfit. We discuss the theoretical aspect of OSVM and the justification on the empirical results in Sections 4.3 and 5.

Our approach is also based on SVM to use unlabeled data for the single-class classification problem. As a major difference from other SVM-based approaches, we first draw a rough class boundary using a rule-based learner that is proven to be PAC learnable from positive and unlabeled examples. After that, we induce an accurate boundary from the rough boundary by using SVM iteratively.

3 SVM Overview

As a binary classification algorithm, SVM gains increasing popularity because it has shown outstanding performance in many domains of classification problems [7], [25], [26]. Especially, it tolerates the problem of high dimensions and sparse instance spaces. There has been a recent surge of interest in SVM classifiers in the learning community.

SVM provides several salient properties, such as maximization of margin and nonlinear transformation of the input space to the feature space using kernel methods [11]. To illustrate, consider its simplest form, a linear SVM. A linear SVM is a hyperplane that separates a set of positive data from a set of negative data with *maximum margin* in the feature space. The *margin* (M) indicates the distance from the hyperplane (class boundary) to the nearest positive and negative data in the feature space. Fig. 2 shows an example of a simple two-dimensional problem that is linearly separable. Each feature corresponds to one dimension in the feature space. The distance from the hyperplane to a data point is determined by the strength of each feature of the data. For instance, consider a resume page classifier. If a page has many strong features related to the concept of "resume" (e.g., words "resume" or "objective" in headings), the page would belong to positive (resume class) in the feature space, and the location of the data point should be far from the class boundary on the positive side. Likewise, another page not having any resume related features, but having many nonresume related features should be located far from the class boundary on the negative side.

In cases where the points are not linearly separable, the SVM has a parameter C (ν in ν-SVM) which controls the noise in training data. The SVM 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 initial feature space to another high-dimensional feature space. Linear kernels are fast, but generally Gaussian kernels perform better in terms of accuracy, especially for single-class classification problems [27]. We used Gaussian kernels in our experiments for the best results and for the fair comparison with one-class SVMs. We discuss the choice of kernels within our framework in Section 6.

4 Mapping-Convergence (M-C) Algorithm

The main thrust of this paper is how to achieve classification accuracy (from positive and unlabeled data) as high as that from labeled (positive and unbiased negative) data. The M-C algorithm achieves this goal.

M-C runs in two stages: the *mapping stage* and *convergence stage*. In the mapping stage, the algorithm uses a weak classifier (e.g., a rule-based learner) that draws an initial approximation of "strong" negative data. (The definition of *strength of negative instances* is provided in Section 4.1.) Based on the initial approximation, the convergence stage runs in iteration using a second base classifier (e.g., SVM) that maximizes margin to make progressively better approximation of negative data. Thus, the class boundary eventually converges to the true boundary of the positive class in the feature space.

How can we draw the approximation of "strong" negative data from positive and unlabeled data? We can identify *strong positive features* from positive and unlabeled data by checking the frequency of those features within positive and unlabeled training data. For instance, a feature that occurs in 90 percent of positive data but only in 10 percent of unlabeled data would be a strong positive feature. Suppose we build a list of every positive feature that occurs in the positive training data more often than in the unlabeled data. By using this list of the positive features, we can filter out any possibly positive data point from the unlabeled data set, which leaves only *strongly negative data* (which we call *strong negatives*). A data point not having any of the positive features in the list is regarded as *a strong negative*. In this case, the list is considered a monotone disjunction list (or *1*-DNF). The *1*-DNF construction algorithm is described in Fig. 4. In this way, one can extract strong negatives from the unlabeled data. This extraction is what the *mapping stage* of the M-C algorithm accomplishes. However, using the list, one can only identify *strong negatives* that are located far from the class boundary. In other words, although *1*-DNF is potentially learnable from positive and unlabeled data, its resulting quality of learning is not good enough.

Input: positive training examples POS, unlabeled data set U
Output: hypothesis at each iteration $h'_1, ..., h'_k$

Ψ_1: an algorithm identifying "strong negatives" from U
Ψ_2: a supervised learning algorithm that maximizes margin (e.g. SVM)

Algorithm:
1. Use Ψ_1 to construct a classifier h from POS and U which classifies only "strong negatives" as negative and the others as positive
2. Classify U by h
   * N_1:= examples from U classified as negative by h
   * P_1:= examples from U classified as positive by h
3. Set $NEG := \emptyset$ and $i := 1$
4. Do loop until $N_i = \emptyset$
   4.1. $NEG := NEG \cup N_i$
   4.2. Use Ψ_2 to construct h'_i from POS and NEG
   4.3. Classify P_i by h'_i
      * N_{i+1}:= examples from P_i classified as negative by h'_i
      * P_{i+1}:= examples from P_i classified as positive by h'_i
   4.4. $i := i + 1$
5. Return $(h'_1, ..., h'_k)$

Fig. 3. Mapping-Convergence algorithm (M-C).

How can we progressively construct better approximation of negative data from the strong negatives and the given positives and unlabeled data? If an SVM is constructed from the positives and the strong negatives only, the class boundary would be far from accurate due to the insufficient negative training data. More concretely, if an SVM is trained with insufficient negative training data, the boundary will be located toward the negative side too much because the boundary of SVM will have equal margins to the positive and negative training data. However, this "biased" boundary can be useful because it still maximizes the margin. The biased boundary divides the space between the strong negatives and the positives into half within the feature space. Thus, one would get a little less strong negative data from the unlabeled data if the rest of the unlabeled data (the unlabeled data excluding the strong negatives) is further classified using the biased boundary. In general, the less strong data would be the data within the area between the strong negatives and the biased boundary in the feature space. By iterating this process, one can continuously extract negative data from the unlabeled data until there exists no negative data in the unlabeled data. The boundary will also converge into the true boundary. This belongs to the *convergence stage* of the M-C algorithm.

We note that our framework can essentially work with any base classifier that satisfies the specific requirements, namely, that Ψ_1 (e.g., *1*-DNF) is a classifier that does not generate false negatives, and that Ψ_2 (e.g., SVM) maximizes margin. To make our discussion concrete, our framework uses *1*-DNF for Ψ_1 and SVM for Ψ_2. We will provide the theoretical proofs of M-C and the requirements of the base classifiers (Section 4.2) after a detailed presentation of the M-C algorithm.

4.1 Algorithm Description

Let us define some basic concepts first. Assume, as usual in classification or pattern recognition problems, that the dissimilarity between two objects is proportional to the distance between them in feature space. Let $\mathcal{F}$ be the feature space and $f(x)$ be the boundary function of the positive class, which computes the distance of x to the boundary in $\mathcal{F}$ such that

$$\begin{array}{ll} f(x) > 0 & \textit{if } x \textit{ is a positive instance,} \\ f(x) < 0 & \textit{if } x \textit{ is a negative instance,} \\ |f(x)| > |f(x')| & \textit{if } x \textit{ is located farther than} \\ & x' \textit{ from the boundary in } \mathcal{F}. \end{array}$$

For example, in SVMs, the boundary function $f(x)$ ($f(x) = w * \Phi(x)$, where w is a weight vector, x is an input vector, and Φ is a nonlinear transformation function) behaves exactly this way. (Section 6 further discusses the usage of nonlinear kernels for the M-C algorithm.) An instance x is *stronger* than x' when x is located farther than x' from the boundary of the positive class in feature space. ($f(x) = 0$ when x is on the boundary.)

Definition 1: Strength of Negative Instances. *For two negative instances x and x' such that $f(x) < 0$ and $f(x') < 0$, if $|f(x)| > |f(x')|$, then x is **stronger** than x'.*

Example 1. Consider a resume classifier. Assume that there are two negative data points (nonresume pages) in the feature space: one is "how to write a resume" page, and the other is "how to write an article" page. In the feature space, the article writing page is considered to be more distant from the resume class because the resume writing page has more features related to resumes (e.g., the word "resume" in text) though it is not an actual resume page.

We present the M-C algorithm in Fig. 3 and the conceptual data flow in Fig. 5. To illustrate, consider classifying "faculty pages" in a university site. POS is a given sample of faculty (positive) pages. U is a sample of the university site (an unbiased sample of the universal set). NEG is the set of the other pages in the university site excluding faculty pages. Given POS and U, our goal is to construct NEG, which is initially set to null. The algorithm Ψ_1 is to extract *strong negatives* from U. Fig. 4 shows a simple monotone disjunction (*1*-DNF) algorithm for Ψ_1. Steps 1

Notation: a feature $x_i \in \{0, 1\}$, v is a subset of $\{x_1, ..., x_n\}$
f_p^i: frequency of feature x_i in positive class
f_u^i: frequency of feature x_i in unlabeled data set
θ: decision threshold (normally set to 1 but can be adjusted)
Input: POS, U
Output: output $h = \sum_{x_i \in v} x_i$
Algorithm: $v = null$
for $i = 0$ to n do
if $\frac{f_p^i}{f_u^i} > \theta$ then $v = v \cup x_i$
return $h = \sum_{x_i \in v} x_i$

Fig. 4. Monotone-disjunction learning for the algorithm Ψ_1.

and 2 in Fig. 3 correspond to the mapping stage which extracts strong negatives from U. The mapping stage fills N_1 with the strong negatives and P_1 with the others. Step 3 initiates a set and a variable for the convergence stage. The convergence loop starts at Step 4. Then, the strong negatives are put into NEG, and a class boundary is constructed from the NEG and POS using the algorithm Ψ_2 (e.g., SVM). Ψ_2 generates a boundary between NEG and POS, which *keeps a maximal margin between them* in the feature space (Fig. 6a). We then use the boundary to divide P_1 into N_2 and P_2. Now, N_2 is accumulated into NEG, and then Ψ_2 is retrained with NEG (currently containing $N_1 \cup N_2$) and the given POS. Ψ_2 generates another boundary between NEG and POS with a maximal margin between them in the feature space as Fig. 6b shows. Then, this new boundary is used to divide P_2 into N_3 and P_3. This process iterates until N_i becomes empty. The final class boundary will be close to the real boundary of the positive data set. We provide the convergence proof in Section 4.2 and empirical evidence in Section 5.

The performance of the algorithm Ψ_1 does not affect the performance of M-C. Poor performance of Ψ_1 only increases the training time by increasing the number of iterations in M-C. (We will discuss this in Section 4.2.) Our experiments also show that regardless of the poor performance of the mapping stage, classification accuracy of M-C converges into that of the traditional SVM trained from labeled data in every class. (See Figs. 7 and 8 mentioned in Section 5.2.)

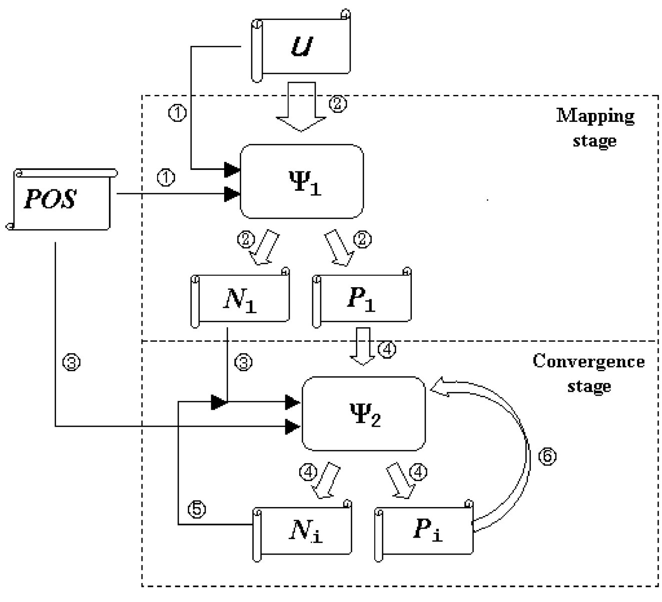

Fig. 5. Data flow diagram of the Mapping-Convergence (M-C) algorithm.

4.2 Analysis

This section provides theoretical justification of the M-C algorithm.

Let $\mathcal{U}$ be the space of an iid (identically independently distributed) sample of the universal set U, and $\mathcal{POS}$ be the space of the positive data set POS. In Fig. 3, let $\mathcal{N}_1$ be the negative space and $\mathcal{P}_1$ be the positive space within $\mathcal{U}$ divided by h (a boundary drawn by Ψ_1), and let $\mathcal{N}_i$ be the negative space and $\mathcal{P}_i$ be the positive space within $\mathcal{P}_{i-1}$ divided by h'_i (a boundary drawn by Ψ_2). Then, we can induce the following formulae from the M-C algorithm of Fig. 3.

$$\mathcal{U} = \mathcal{P}_i + \bigcup_{k=1}^{i} \mathcal{N}_k, \tag{1}$$

$$\mathcal{P}_i = \mathcal{POS} + \bigcup_{k=i+1}^{n} \mathcal{N}_k, \tag{2}$$

where n is the number of iterations in the M-C algorithm.

Theorem 1: Fast Convergence of Boundary. *Suppose U and POS are uniformly distributed. If the algorithm Ψ_1 does not generate false negatives and the algorithm Ψ_2 maximizes margin, then the class boundary of M-C converges into the boundary that maximally separates POS and U outside $\mathcal{POS}$. The number of iterations for the convergence in M-C is logarithmic to the size of $\mathcal{U} - (\mathcal{N}_1 + \mathcal{POS})$.*

Proof. $\mathcal{N}_1 \cap \mathcal{POS} = \emptyset$ because a classifier h constructed by the algorithm Ψ_1 does not generate false negative. A classifier h'_1 constructed by the algorithm Ψ_2, trained from the separated space $\mathcal{N}_1$ and $\mathcal{POS}$, divides the rest of the space ($\mathcal{U} - (\mathcal{N}_1 + \mathcal{POS})$ which is equal to $\cup_{k=2}^{n} \mathcal{N}_k$) into two classes with a boundary that maximizes the margin between $\mathcal{N}_1$ and $\mathcal{POS}$. The first half becomes $\mathcal{N}_2$ and the other half becomes $\cup_{k=3}^{n} \mathcal{N}_k$. Repeatedly, a classifier h'_i constructed by the same algorithm Ψ_2, trained from the separated space $\cup_{k=1}^{i} \mathcal{N}_k$ and $\mathcal{POS}$, evenly divides the rest of the space $\cup_{k=i+1}^{n} \mathcal{N}_k$ into $\mathcal{N}_{i+1}$ and $\cup_{k=i+2}^{n} \mathcal{N}_k$ with equal margins. Thus, $\mathcal{N}_{i+1}$ is always half of $\mathcal{N}_i$. Therefore, the number of iterations n will be logarithmic to the size of $\mathcal{U} - (\mathcal{N}_1 + \mathcal{POS})$.

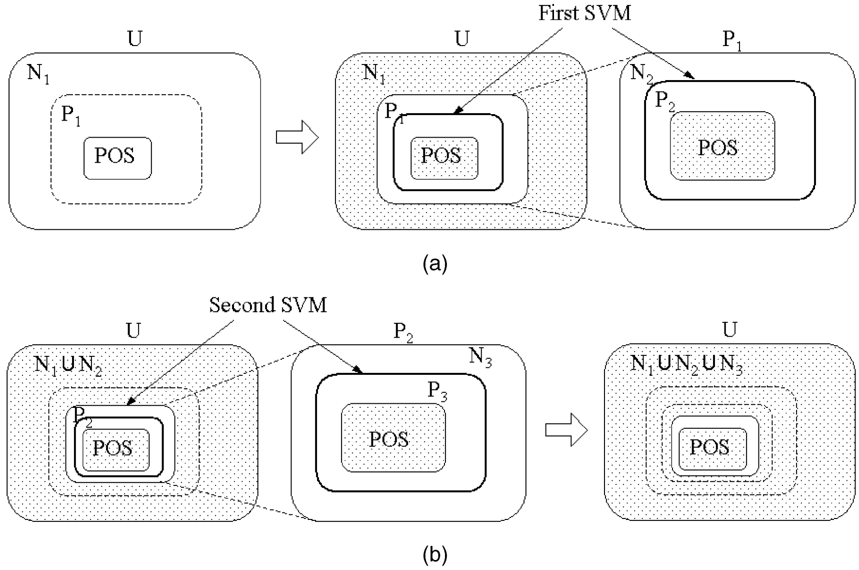

Fig. 6. Marginal convergence of SVM: (a) Training first SVM (from N_1 and POS) that divides P_1 into N_2 and P_2. (b) Training second SVM (from $N_1 \cup N_2$ and POS) that divides P_2 into N_3 and P_3.

M-C stops when $N_{i+1} = \emptyset$, i.e., there exists no sample of U outside $\mathcal{POS}$. Therefore, the final boundary will be located between POS and U outside $\mathcal{POS}$. □

Theorem 2: Convergence Safety. *If Ψ_1 does not generate false negatives and Ψ_2 does not misclassify separable data set in the feature space, the final boundary h'_n of M-C does not trespass the positive space $\mathcal{POS}$ regardless of the number of iterations in the algorithm.*

Proof. $\mathcal{N}_1 \cap \mathcal{POS} = \emptyset$ because h does not generate false negatives. When $\mathcal{N}_i \cap \mathcal{POS} = \emptyset$, $\mathcal{N}_{i+1} \cap \mathcal{POS} = \emptyset$ because h'_i, trained from $\cup_{k=1}^{i} \mathcal{N}_k$ and $\mathcal{POS}$, classifies $\mathcal{P}_i$ into $\mathcal{N}_{i+1}$ as negative and $\mathcal{P}_{i+1}$ as positive, and h'_i does not misclassify separable data set, so $\mathcal{N}_{i+1} \cap \mathcal{POS} = \emptyset$. Therefore, h'_n, trained from $\cup_{k=1}^{n} \mathcal{N}_k$ and $\mathcal{POS}$, does not trespass $\mathcal{POS}$ regardless of the number of iterations n because $(\cup_{k=1}^{n} \mathcal{N}_k) \cap \mathcal{POS} = \emptyset$. □

By Theorem 1, in order to guarantee the fast convergence of the boundary, the learning algorithm Ψ_1 must not generate false negatives, and Ψ_2 must maximize margin. First, we can adjust the threshold θ of the monotone disjunction learning algorithm (Fig. 4) so that it makes 100 percent recall by sacrificing precision. (Note that false positives determine precision and false negatives determine recall.) Second, SVM maximizes margin. Thus, M-C guarantees that the boundary converges fast with the *1*-DNF and SVM.

Theorem 2 shows that the boundary does not trespass the area of positive data set even if we do the iteration an infinite number of times in M-C. SVM also does not misclassify separable data set. Thus, M-C also guarantees that the convergence is safe as well as fast.

We can induce a fact from Theorem 1 that the performance of the mapping algorithm Ψ_1 does not affect the accuracy of the final boundary but does affect the training time. The training time of M-C is the training time of SVM multiplying the number of iterations. By Theorem 1, the number of iterations is logarithmic to the size of the negative space (more precisely, the size of $\mathcal{U} - (\mathcal{N}_1 + \mathcal{P})$ which is the negative space subtracted by the "strong negatives"). The "strong negatives" are determined by the mapping algorithm Ψ_1. Therefore, the performance of the mapping stage affects only the training time, but not the accuracy of the final boundary because the boundary will converge. Our experiments in Section 5.2 also show that the final boundary becomes very accurate although the initial boundary of the mapping stage is very rough. The classification time of M-C depends on the algorithm Ψ_2 because the final boundary of M-C is a boundary function of Ψ_2.

Theorem 1 also shows that the final boundary will be located between the positive data set (POS) and the unlabeled data (U) outside the positive area ($\mathcal{POS}$) in the feature space, which implies that the boundary accuracy depends on the quality of the samples: POS and U. Specifically, we can observe that the final boundary will overfit the true positive concept space if the positive data is "undersampled". If the positive data set does not cover major directions of the positive space, the unlabeled data outside the area of the positive data set but inside the real positive concept space will be considered negative by the final boundary. In this case, an intermediate boundary may actually be closer to the true boundary because the final boundary may be overfitting. In practice, positive data set tends to be undersampled especially when the concept is highly complicated or the data is manually sampled because it is often hard to sample the positive data set that completes the true concept space. In our experiments, the homepage class and college admission class in Fig. 7, and the project class and faculty class in Fig. 8 show the peak performance in an intermediate iterations because their final boundaries overfit. We discuss this phenomenon more in Section 5.2.

4.3 One-Class SVM (OSVM)

OSVM also draws a boundary around positive data set in the feature space [27], [28]. However, its boundary cannot

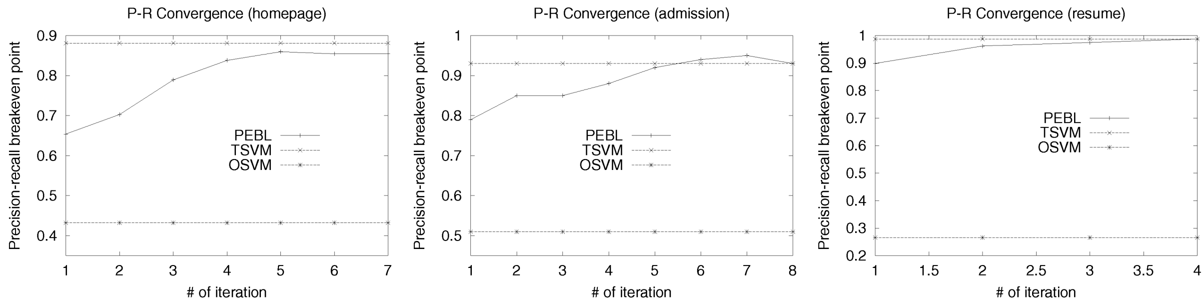

Fig. 7. Convergence of performance (P-R: precision-recall breakeven point) when the universal set is the Internet. TSVM indicates the traditional SVM constructed from manually labeled (positive and unbiased negative) data.

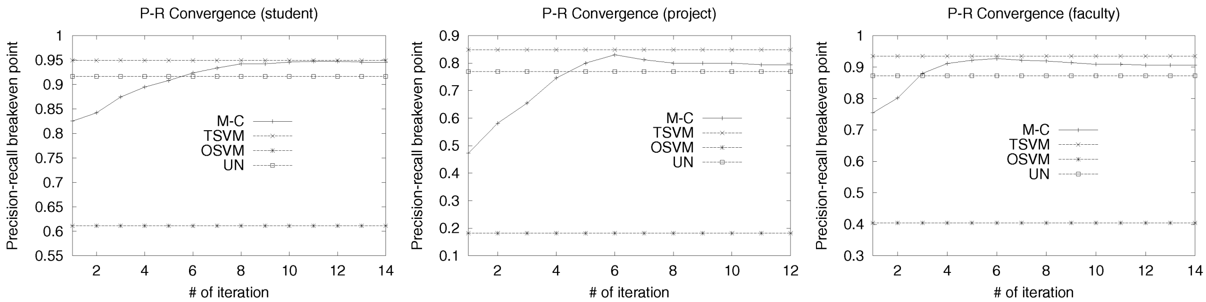

Fig. 8. Convergence of performance (P-R: precision-recall breakeven point) when the universal set is computer science department sites. TSVM indicates the traditional SVM constructed from manually labeled (positive and unbiased negative) data. UN indicates the SVM constructed from positive and sample of universal set as a substitute for unbiased negative training data.

be as accurate as that of M-C. In this section, we supply the theoretical justification of why M-C outperforms OSVM. We also show the empirical evidence of it in Section 5.

Let $\{x_i\} \subseteq \chi$ be a data set of N points, with $\chi \subseteq \Re^d$, the input space. Using a nonlinear transformation Φ from χ to some high-dimensional feature space, OSVMs look for the smallest enclosing sphere of radius R, described by the constraints:

$$||\Phi(x_i) - a||^2 \leq R^2 + \xi_i, \forall i, \xi_i \geq 0, \tag{3}$$

where $||\cdot||$ is the Euclidean norm, a is the center of the sphere, and ξ_i are slack variables to incorporate soft constraints. After applying the Lagragian method to solve this problem, the Lagrangian W is written as:

$$W = \sum_i \alpha_i K(x_i, x_i) - \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j). \tag{4}$$

We use the Gaussian kernel for K because the Gaussian kernel performs the best for single-class classification as noted in [27]. That is,

$$K(x_i, x_j) = e^{-\gamma||x_i - x_j||^2}, \tag{5}$$

where γ is a parameter that controls the number of support vectors.

OSVMs require much more positive training data in order to draw an accurate boundary because support vectors (SVs) in OSVMs are only supported by positive data whereas M-C utilizes unlabeled data in addition. SVs determine the boundary shape. As the number of SVs is decreasing, the boundary shape is becoming a hypersphere which will underfit the true concept. When the number of SVs is increasing, the boundary shape is getting more flexible and it starts overfitting at some point. Since the SVs in OSVM are only supported by positive data, the OSVM overfits and underfits more easily before it draws an accurate boundary because the small number of positive support vectors can hardly cover the major directions of the positive area in the high-dimensional feature space. In our experiments, OSVM performs poorly even with careful parameter settings.

5 Experimental Results

In this section, we provide empirical evidence that our PEBL framework using positive and unlabeled data performs as well as the traditional SVM using manually labeled (positive and unbiased negative) data. We present experimental results with two different universal sets: the Internet (*Experiment 1*) and university computer science departments (*Experiment 2*). The Internet (*Experiment 1*) is the largest possible universal set in the Web, and CS department sites (*Experiment 2*) is a conventional small universal set. We design these two experiments with the two totally different sizes of universal sets so that we verify the applicability of our method on various domains of universal sets.

We first consider three different classes for each universal set and then merge the three into one positive class of a mixture model. The experiment on the merged class is to verify the ability of our method on dealing with the positive classes of mixture models.

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,[2] which is a free open directory of the Web containing millions of Web pages. A 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 2,388 pages from DMOZ to collect unbiased unlabeled data. We also manually collected 368 personal homepages, 192 college admission pages, and 188 resume pages to classify the three corresponding classes. (Each class is classified independently.) We used about half of the pages of each class for training and the other half for testing. For testing negative data (for evaluating the classifier), we manually collected 449 nonhomepages, 450 nonadmission pages, and 533 nonresume pages. (We collected negative data just for evaluating the classifier we construct. The PEBL does not require collecting negative data to construct classifiers.) For instance, for personal homepage class, we used 183 positive and 2,388 unlabeled data for training, and used 185 positive and 449 negative data for testing.

Experiment 2: University computer science department. The WebKB data set [29] contains 8,282 Web pages gathered from university computer science departments. The collection includes the entire computer science department Web sites from various universities. The pages are divided into seven categories: student, project, faculty, course, staff, department, and others. In our experiments, we independently classify the three most common categories: *student*, *project*, and *faculty*, which contain 1,641, 504, and 1,124 pages, respectively. We randomly selected 1,052 and 589 student pages, 339 and 165 project pages, and 741 and 383 faculty pages for training and testing, respectively. For testing negative data, we also randomly selected 662 nonstudent pages, 753 nonproject pages, and 729 nonfaculty pages. We randomly 4,093 picked up pages from all categories to make a sample universal set and the same sample is used for all the three classes as unlabeled data. For instance, for faculty page classification, we used 741 positive and 4,093 unlabeled data for training, and used 383 positive and 729 negative data for testing.

We extracted features from different parts of a page—URL, title, headings, link, anchor-text, normal text, and metatags. Each feature is a predicate indicating whether each term or special character appears in each part, e.g., "~" in URL, or a word "homepage" in title. In Web page classification, normal text is often a small portion of a Web page and structural information usually embodies crucial features for Web pages, thus the standard feature representation for text classification such as TFIDF is not often used in the Web domain because it is tricky to incorporate such representations for structural features. For instance, occurrence of "~" in URL is more important information than how many times it occurs. For the same reason, we did not perform stopwording and stemming because the common words in text classification may not be common in Web pages. For instance, a common stopword, "I" or "my," can be a good indicator of a student homepage. However, our feature extraction method may not be the best way for SVM for Web page classification. Using more sophisticated techniques for preprocessing, the features could improve the performance further.

2. Open Directory Project, http://dmoz.org.

TABLE 1
Precision-Recall Breakeven Points (P-R) Showing Performance of PEBL (Positive Example-Based Framework), TSVM (Traditional SVM Trained from Manually Labeled Data), and OSVM (One-Class SVM) in the Two Universal Sets (U)

U	Class	TSVM	PEBL	OSVM
The Internet	homepage	88.11	85.95 (7)	43.24
	admission	93.0	95.0 (8)	51.0
	resume	98.73	98.73 (4)	26.58
CS Department	student	94.91	94.74 (14)	61.12
	project	84.85	83.03 (12)	18.18
	faculty	93.47	92.69 (14)	40.47

The number of iterations to the convergence in PEBL is shown in parentheses.

For SVM implementation, we used LIBSVM.[3] As we discussed in Section 3, we used Gaussian kernels because of its high accuracy. For a positive example-based learning problem, Gaussian kernels perform the best because of its flexible boundary shape that fits complicated positive concept [27]. Table 2 shows the performances of Gaussian kernels and linear kernels on a mixture model of positive class. Both TSVM and M-C show better performance with Gaussian kernels. We discuss more about the choice of kernels within our framework in Section 6.

We report the result with *precision-recall breakeven 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:

$$Precision = \frac{\#\ of\ correct\ positive\ predictions}{\#\ of\ positive\ predictions},$$

$$Recall = \frac{\#\ of\ correct\ positive\ predictions}{\#\ of\ positive\ data}.$$

The *precision-recall breakeven 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

We compare three different methods: TSVM, PEBL, and OSVM. (See Table 1 for the full names.) We show the performance comparison on the six classes of the two universal sets: the Internet and CS department sites. We first constructed an SVM from positive (POS) and unlabeled data (U) using PEBL. On the other hand, we manually classified the unlabeled data (U) to extract unbiased negatives from them, and then built a TSVM (Traditional SVM) trained from POS and those unbiased

3. http://www.csie.ntu.edu.tw/cjlin/libsvm/.

TABLE 2
Precision-Recall Breakeven Points (P-R) Showing the Performance of PEBL and TSVM on Mixture Models in the Two Universal Sets

U	Class	Kernel	TSVM	PEBL
The Internet	$homepage + admission + resume$	Linear	85.99	85.16 (7)
		Gaussian	86.54	86.26 (7)
CS Department	student + project + faculty	Linear	89.18	89.36 (23)
		Gaussian	91.12	90.24 (27)

The number of iterations to the convergence in PEBL is shown in parentheses.

negatives. We also constructed OSVM from POS. We tested the same testing documents using those three methods.

Table 1 shows the P-R (precision-recall breakeven points) of each method, and also the number of iterations to converge in the case of the PEBL. In most cases, PEBL without negative training data performs almost as well as TSVM with manually labeled training data. For example, when we collect 1,052 student pages and manually classify 4,093 unlabeled data to extract nonstudent pages to train TSVM, it gives 94.91 percent P-R (precision-recall breakeven point). When we use PEBL without doing the manual classification, it gives 94.74 percent P-R. However, OSVM without the manual classification performs very poorly (61.12 percent P-R).

Figs. 7 and 8 show the details of performance convergence at each iteration in the experiment of the universal set, the Internet, and CS department sites, respectively. For instance, consider the graphs of the first column (personal homepage class) in Fig. 7. The P-R of M-C is 0.65 at the first iteration, and 0.7 at the second iteration. At the seventh iteration, the P-R of M-C is very close to that of TSVM. The performance (P-R: precision-recall breakeven point) of M-C is converging rapidly into that of TSVM in all our experiments.

The P-R convergence graphs in Fig. 8 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. As we discussed at the end of Section 1, they obviously show the performance decrement when using U as a substitute for negative training data because a small number of false positive training data affects significantly the set of support vectors which is critical to classification accuracy.

As we discussed in Section 4.2, for some classes, the performance starts decreasing at some point of the iterations in M-C. For instance, the convergence graph of the project class in Fig. 8 shows the peak performance at the sixth iteration and the performance starts decreasing from that point. This happens when the positive data set is undersampled so that it does not cover major directions of positive area in the feature space. In this case, the final boundary of M-C, which fits around the positive data set tightly, tends to end up overfitting the true positive concept space. Finding the best number of iterations in M-C requires a validation process which is used to optimize parameters in conventional classifications. However, M-C is assumed to have no negative examples available, which makes impossible to use the conventional validation methods to find the best number of iterations. Determining the stopping criteria for M-C without negative examples is an interesting further work for the cases that the positive data is seriously undersampled.

In order to experiment with the ability of M-C on dealing with the positive classes of mixture models, we combined the three classes of each universal set to compose one big composite positive class. In other words, for the first universal set, the Internet, we made the positive class from personal homepage, resume page, and college admission page. For the second universal set, CS department sites, we compose the positive class of student, project, and faculty pages. Table 2 shows that both TSVM and M-C perform well with mixture models. As expected, Gaussian kernels performs better than linear kernels for this positive example-based learning problem.

6 Discussion

In this section, we discuss several important issues pertinent to the PEBL framework.

6.1 Possible Extensions of the PEBL Methodology to Non-SVM Learning Method?

Other supervised learning methods such as probabilistic (e.g., naive Bayes) or mistake-driven learning methods (e.g., perceptron, winnow) do not maximize the margin. As we discussed in Section 3, SVM maximizes the margin, which ensures that the initial class boundary converges into the true boundary of the two (positive and negative) classes. Other learning methods do not guarantee the convergence of the class boundary, and even if they converge the boundary occasionally, the rate of convergence would be slower.

6.2 Choice of Kernel Functions

SVMs provide nonlinear transformation of input space to another feature space using advanced kernels (e.g., polynomial or Gaussian kernels) [11]. *Polynomial kernels* combine the input features to create high-dimensional features in the feature space. As the degree of polynomial kernels grows, higher-dimensional function is used to try to fit more training data, which may end up overfitting at some point. In our experiments with Web pages, even the second degree of polynomial kernels degraded the accuracy due to the overfitting. Gaussian kernels perform the best for positive example-based learning because they draw flexible boundaries to fit a mixture model by implicit transformation of the feature space. Gaussian kernels have an additional parameter γ which needs to be optimized. In our experiments, we used Gaussian kernels of a fixed γ, the same value as used for TSVM, for fair comparisons with TSVM and OSVM (which also performs the best with Gaussian kernels [27]). However, linear kernels also showed very comparable performance as Gaussian kernels in our experiments because Web page classification is often considered linearly separable problem due to its high dimensionality as in text classification [25]. For pattern recognition problems in other domains where linear kernels do not work well, optimizing the parameters of the advanced kernels is necessary at each iteration of M-C, which could be further work for applying M-C to other domains. Identifying or building proper kernel functions for specific problems is still ongoing research.

6.3 Further Improvements of Algorithm Performance

The M-C algorithm takes multiple times longer to train one class than traditional SVM since the iteration of training SVM has to be serialized due to data dependency between adjacent iterations. However, only part of the NEG (the negatives accumulated at each iteration) is dependent between two iterations, which leaves some room for speeding up by parallel processing, such as pipelining techniques. Using pipeline architecture to speed up this training process and using advanced kernel methods to increase classification accuracy for specific problems could be good research directions. Once training is done (a classifier is constructed), however, the speed of testing (classifying) is the same as traditional SVMs, which makes our framework practical in many real-world problems.

6.4 PEBL for Diverse Domains

As based on SVM, PEBL can be applied to other problem domains where SVM also works well, e.g., pattern recognition or bioinformatics. The major adaptation would be for the mapping algorithm to handle numerical attributes which other domains often use (unlike our current binary features). Once the mapping stage initializes the strong negatives from unlabeled data, the convergence stage will have no difficulties in handling numerical features since SVMs work well with numerical attribute values.

7 Summary and Conclusions

Web page classification is one of the essential techniques for Web mining because classifying Web pages of an interesting class is often the first step of mining the Web. However, constructing a classifier for an interesting class requires laborious preprocessing, such as collecting positive and negative training examples. In particular, *collecting negative training examples* requires arduous work and caution to avoid bias. The *Positive Example Based Learning (PEBL)* framework for Web page classification eliminates the need for manually collecting negative training examples in preprocessing. The *Mapping-Convergence (M-C)* algorithm in the PEBL framework achieves classification accuracy (with positive and unlabeled data) as high as that of traditional SVM (with positive and negative data). Our experiments show that, given the same set of positive examples, the M-C algorithm outperforms one-class SVMs, and it is almost as accurate as traditional SVM. Currently, PEBL needs an additional multiplicative logarithmic factor in the training time on top of that of a SVM. However, as training is typically an offline process, its increased time may not be critical in practice. For the more critical online testing, PEBL requires the same time as SVMs. PEBL contributes to automating the preprocessing of Web page classification without much loss of classification performance. We believe that further improvements of PEBL will be interesting, e.g., for applications that require fast training or numerical attributes.

Acknowledgements

This paper is based on the paper by H. Yu, J. Han, and K.C. Chang, "PEBL: Positive Example Based Learning for Web Page Classification Using SVM," *Proc. 2002 Int'l Conf. Knowledge Discovery in Databases (KDD '02)*, pp. 229-238, July 2002. However, this submission is substantially enhanced in technical contents and contains new and major-value added technical contribution in comparison with our conference publication. This work was supported in part by the US National Science Foundation under Grant No. IIS-02-09199 and IIS-01-33199, the University of Illinois, and an IBM Faculty Award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

References

[1] H. Chen, C. Schuffels, and R. Orwig, "Internet Categorization and Search: A Machine Learning Approach," *J. Visual Comm. and Image Representation*, vol. 7, pp. 88-102, 1996.
[2] H. Mase, "Experiments on Automatic Web Page Categorization for IR System," technical report, Stanford Univ., Stanford, Calif., 1998.
[3] E.J. Glover, G.W. Flake, and S. Lawrence, "Improving Category Specific Web Search by Learning Query Modifications," *Proc. 2001 Symp. Applications and the Internet (SAINT '01)*, pp. 23-31, 2001.
[4] A. Kruger, C.L. Giles, and E. Glover, "Deadliner: Building a New Niche Search Engine," *Proc. Ninth Int'l Conf. Information and Knowledge Management (CIKM '00)*, pp. 272-281, 2000.
[5] E.N. Mayoraz, "Multiclass Classification with Pairwise Coupled Neural Networks or Support Vector Machines," *Proc. Int'l Conf. Artificial Neural Network (ICANN '01)*, pp. 314-321, 2001.
[6] E.L. Allwein, R.E. Schapire, and Y. Singer, "Reducing Multiclass to Binary: A Unifying Approach for Margin Classifiers," *J. Machine Learning Research*, vol. 1, pp. 113-141, 2000.
[7] S. Dumais and H. Chen, "Hierarchical Classification of Web Content," *Proc. 23rd ACM Int'l Conf. Research and Development in Information Retrieval (SIGIR '00)*, pp. 256-263, 2000.
[8] H. Yu, J. Han, and K.C.-C. Chang, "PEBL: Positive-Example Based Learning for Web Page Classification Using SVM," *Proc. Eighth Int'l Conf. Knowledge Discovery and Data Mining (KDD '02)*, pp. 239-248, 2002.
[9] F. Letouzey, F. Denis, and R. Gilleron, "Learning from Positive and Unlabeled Examples," *Proc. 11th Int'l Conf. Algorithmic Learning Theory (ALT '00)*, 2000.
[10] F. DeComite, F. Denis, and R. Gilleron, "Positive and Unlabeled Examples Help Learning," *Proc. 11th Int'l Conf. Algorithmic Learning Theory (ALT '99)*, pp. 219-230, 1999.
[11] C. Cortes and V. Vapnik, "Support Vector Networks," *Machine Learning*, vol. 30, no. 3, pp. 273-297, 1995.

[12] W. Wong and A.W. Fu, "Finding Structure and Characteristics of Web Documents for Classification," *Proc. 2000 ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery (DMKD '00)*, pp. 96-105, 2000.
[13] J. Yi and N. Sundaresan, "A Classifier for Semi-Structured Documents," *Proc. Sixth Int'l Conf. Knowledge Discovery and Data Mining (KDD '00),* pp. 340-344, 2000.
[14] H. Oh, S. Myaeng, and M. Lee, "A Practical Hypertext Categorization Method Using Links and Incrementally Available Class Information," *Proc. 23rd ACM Int'l Conf. Research and Development in Information Retrieval (SIGIR '00),* pp. 264-271, 2000.
[15] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum Likelihood from Incomplete Data Via the EM Algorithm," *J. Royal Statistical Soc., Series B,* vol. 39, pp. 1-38, 1977.
[16] K. Nigam, "Text Classification from Labeled and Unlabeled Documents Using EM," *Machine Learning,* vol. 39, pp. 103-134, 2000.
[17] T. Joachims, "Transductive Inference for Text Classification Using Support Vector Machines," *Proc. 16th Int'l Conf. Machine Learning (ICML '00),* pp. 200-209, 1999.
[18] B. Liu, W.S. Lee, P.S. Yu, and X. Li, "Partially Supervised Classification of Text Documents," *Proc. 19th Int'l Conf. Machine Learning (ICML '02),* pp. 8-12, 2002.
[19] F. Denis, "PAC Learning from Positive Statistical Queries," *Proc. 10th Int'l Conf. Algorithmic Learning Theory (ALT '99),* pp. 112-126, 1998.
[20] A. Frosini, M. Gori, and P. Priami, "A Neural Network-Based Model for Paper Currency Recognition and Verification," *IEEE Trans. Neural Networks,* vol. 7, no. 6, pp. 1482-1490, Nov. 1996.
[21] M. Gori, L. Lastrucci, and G. Soda, "Autoassociator-Based Models for Speaker Verification," *Pattern Recognition Letters,* vol. 17, pp. 241-250, 1995.
[22] L.M. Manevitz and M. Yousef, "One-Class SVMs for Document Classification," *J. Machine Learning Research,* vol. 2, pp. 139-154, 2001.
[23] S.M. Bileschi and B. Heisele, "Advances in Component-Based Face Detection," *Proc. 2002 First Int'l Workshop Pattern Recognition with Support Vector Machines,* pp. 135-143, 2002.
[24] D.M.J. Tax and R.P.W. Duin, "Uniform Object Generation for Optimizating One-Class Classifiers," *J. Machine Learning Research,* vol. 2, pp. 155-173, 2001.
[25] T. Joachims, "Text Categorization with Support Vector Machines," *Proc. 10th European Conf. Machine Learning (ECML '98),* pp. 137-142, 1998.
[26] Y. Yang and Y. Lui, "A Re-Examination of Text Categorization Methods," *Proc. 22nd ACM Int'l Conf. Research and Development in Information Retrieval (SIGIR '99),* pp. 42-49, 1999.
[27] D.M.J. Tax and R.P.W. Duin, "Support Vector Domain Description," *Pattern Recognition Letters.* vol. 20, pp. 1991-1999, 1999.
[28] B. Scholkopf, R.C. Williamson, A.J. Smola, and J. Shawe-Taylor, "SV Estimation of a Distribution's Support," *Proc. 14th Neural Information Processing Systems (NIPS '00),* pp. 582-588, 2000.
[29] M. Craven, D. Dipasquo, and D. Freitag, "Learning to Extract Symbolic Knowledge from the World Wide Web," *Proc. 15th Conf. Am. Assoc. for Artificial Intelligence (AAAI '98),* pp. 509-516, 1998.

Hwanjo Yu received the BS degree in computer science and engineering from Chung-Ang University, Seoul, in 1997. He is currently a PhD candidate in the Computer Science Department at the University of Illinois at Urbana-Champaign. He has been working on knowledge discovery, data mining, Web mining, machine learning, information retrieval, and bioinformatics, with various publications from related conferences: SIGKDD, SIGIR, ICDM, WWW, etc. He received a student award in 2002 ACM SIGKDD conference.

Jiawei Han received the PhD degree in computer science from the University of Wisconsin at Madison. He is a professor in the Computer Science Department at the University of Illinois at Urbana-Champaign. Previously, he was an Endowed University Professor at Simon Fraser University, Canada. He has been working on research into data mining, data warehousing, database systems, spatial databases, deductive and object-oriented databases, Web databases, biomedical databases, etc., with more than 200 journal and conference publications. He has chaired or served in many program committees of international conferences and workshops, including 2001 and 2002 SIAM-Data Mining Conference (PC cochair), 2004 and 2002 International Conference on Data Engineering (PC vice-chair), ACM SIGKDD conferences (2001 best paper award chair, 2002 student award chair), 2002 and 2003 ACM SIGMOD conference (2000 demo/exhibit program chair), etc. He also served or is serving on the editorial boards for *Data Mining and Knowledge Discovery: An International Journal*, *IEEE Transactions on Knowledge and Data Engineering*, and *Journal of Intelligent Information Systems*. He has also been serving on the board of directors for the executive committee of the ACM Special Interest Group on Knowledge Discovery and Data Mining (SIGKDD). His book *Data Mining: Concepts and Techniques* (Morgan Kaufmann, 2001) has been popularly adopted as a textbook in many universities.

Kevin Chen-Chuan Chang received the MS degree in computer science in 1995 and the PhD degree in electrical engineering in 2001 from Stanford University. He is an assistant professor in the Department of Computer Science, University of Illinois at Urbana-Champaign. His research interests include database systems, Internet information access, and information integration. He is a member of the IEEE and the ACM.