PEBL: Positive Example Based Learning for Web Page Classification Using SVM

Hwanjo Yu
Department of Computer Science
University of Illinois
Urbana-Champaign, IL 61801
USA
hwanjoyu@uiuc.edu

Jiawei Han
Department of Computer Science
University of Illinois
Urbana-Champaign, IL 61801
USA
hanj@cs.uiuc.edu

Kevin Chen-Chuan Chang
Department of Computer Science
University of Illinois
Urbana-Champaign, IL 61801
USA
kcchang@cs.uiuc.edu

ABSTRACT

Web page classification is one of the essential techniques for Web mining. Specifically, classifying Web pages of a user-interesting class is the first step of mining interesting information from the Web. However, constructing a classifier for an interesting class requires laborious pre-processing 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 non-homepages (negative examples). In particular, *collecting negative training examples* requires arduous work and special caution to avoid biasing them. We introduce in this paper the *Positive Example Based Learning (PEBL)* framework for Web page classification which eliminates the need for manually collecting negative training examples in pre-processing. We present an algorithm called *Mapping-Convergence (M-C)* that achieves classification accuracy (with positive and unlabeled data) as high as that of traditional SVM (with positive and negative data). Our experiments show that when the M-C algorithm uses the same amount of positive examples as that of traditional SVM, the M-C algorithm performs as well as traditional SVM.

Categories and Subject Descriptors

H.m [**Information Systems**]: Miscellaneous; I.5.2 [**Computing Methodologies**]: Design Methodologies—*Classifier design and evaluation*; I.7.m [**Computing Methodologies**]: Document and Text Processing—*Miscellaneous*; I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

General Terms

Algorithms, Performance, Experimentation

SIGKDD '02 Edmonton, Alberta, Canada

Keywords

Mapping-Convergence (M-C) algorithm, labeled data, unlabeled data, 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 database of information, in terms of both volume and variance. Given the fact that Web pages are based on loosely structured text, many variances of statistical text learning algorithms have been applied to Web page classification.

Previous approaches for multi-class classification problems [2, 15] 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 not a very realistic assumption because of the fact that a single page usually falls into several categories. Also, the pre-defined classes usually do not fit end-users' search purposes because it is hard for a pre-defined set of classes to satisfy the diverse and changing interests of users. For example, a company wants to find "XML expert" pages from the Internet. They may start by searching for the keywords "XML" or "expert" on any search engine, then trying to refine the search results repeatedly until they collect a fair amount of "XML expert" pages. However, if they specify a search class or domain information such as "resume" or "personal homepage", the refining process could be eliminated by applying a query of "XML" upon the classes of resume or personal homepage.

Researchers have realized these problems, and proposed the classifications of user-interesting classes such as "calls for paper", "personal homepage" [9]. This involves binary classification techniques that distinguish Web pages of a user-interesting 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 of building a domain specific engine [12] as well as a multiclass classification system [16, 1]. When binary classifiers are considered independently

[1]Classification is distinguished from clustering in terms that classification requires a learning phase (training phase) before actual classification (testing phase).

within a multiclass classification system, an item may fall into none, one, or more than one class, which relaxes the mutual-exclusion assumption between classes [8].

Constructing a binary classifier for Web pages requires laborious pre-processing. For instance, in order to construct a "homepage" classifier, one needs to collect a sample of homepages (positive training examples) and a sample of non-homepages (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 non-homepage 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.

We present here the *Positive Example Based Learning (PEBL)* framework for Web page classification, which eliminates the need of manually collecting negative training examples in pre-processing. The PEBL framework uses a sample of the universal set as unlabeled data, and learns from additional *positive* data and the given *unlabeled data* without requiring *labeled data*. *Labeled data* indicates both positive and negative examples manually classified to train a classifier, which our framework does not require. *Unlabeled data* indicates random samples of the universal set for which the class of each sample is unknown and irrelevant. (e.g. samples of homepages and non-homepages are *labeled* data because we know the class of the samples from manual classification, and random sampling of the Internet provides *unlabeled* data because the classes of the samples are of no concern.) In many real-world learning problems including Web page classification problem, unlabeled and positive data are widely available while negative data sets are rare and expensive [13, 5]. 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), but negative data are expensive if detection tests for the disease are expensive since all patients in the database cannot be assumed to be negative samples if they have never been tested.

Our goal is to achieve classification accuracy from positive and unlabeled data as high as that from labeled (positive and negative) data. (We only assume that the unlabeled data is unbiased.) There are two main challenges in this approach: (1) collecting unbiased unlabeled data from universal set (e.g. the Internet), and (2) achieving classification accuracy from the positive and unlabeled data as high as that from the labeled (positive and negative) data. To address the first issue, we assume it sufficient to use random sampling to collect unbiased unlabeled data. Random sampling is supported in most databases and warehouses, including search engine databases, or 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. We introduce an algorithm called *Mapping-Convergence (M-C)* that learns from positive and unlabeled data as accurately as a traditional SVM (Support Vector Machine) that learns from labeled data. The M-C algorithm uses SVM technology [3], especially the marginal property of SVMs, which ensures classification accuracy from positive and unlabeled data converges to the accuracy received from labeled data. We present the details of the SVM properties in Section 3.

Our experiments (Section 5) fall into two different domains of universal sets: one is the Internet (*Experiment 1*), and the other is computer science department sites (*Experiment 2*). Both experiments show that the PEBL framework using the M-C algorithm achieves classification accuracy from positive and unlabeled data as high as that from labeled data.

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

The new contributions of our PEBL framework are the following.

- Pre-processing 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. Figure 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.

- PEBL achieves accuracy as high as that of a typical framework without loss of efficiency in testing. PEBL runs the M-C algorithm in 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 equivalent to that of a typical SVM in terms of both accuracy and efficiency.

- Although we address Web page classification in this paper because of its high demand for this application, our PEBL framework can be easily applied to other classification problems within different domains, such as diagnosis databases, text databases, or electronic commercial databases.

Note that this paper concentrates on the classification algorithms, and 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 [20, 22, 18]. Additionally, 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 (see Section 5 for more details).

The remainder of 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

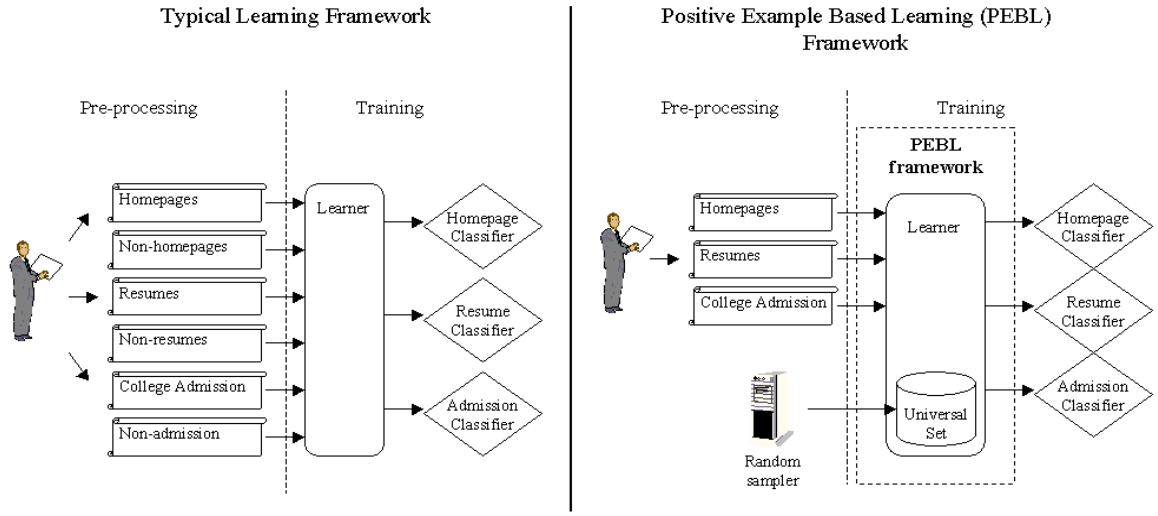


Figure 1: A typical learning framework versus the Positive Example Based Learning (PEBL) framework. Once a sample of the universal set is collected in PEBL, the same sample is reused as unlabeled data for every class.

algorithm and provide justification of why it works. 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 learning algorithm, and the PEBL framework. Finally, Section 7 reviews and concludes our discussion of the PEBL framework.

2. RELATED WORK

Previous approaches have used unlabeled data, which present issues that need discussion.

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. This is called *semi-supervised learning*. The EM algorithm is a representative algorithm which can be used for either semi-supervised learning or unsupervised learning [6]. 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 [17]. Another semi-supervised learning occurs when it is combined with SVMs, to form transductive SVM [11]. With careful parameter settings, both of those works show good results within certain environments, such as environments with an extremely low amount of labeled data. When the number of labeled data grows or the generative assumptions are violated, semi-supervised learning schemes suffer significant degradation of classification accuracy.

Another stream of research which uses unlabeled data in classification is termed *learning from positive and unlabeled data*. In 1998, F. 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 [7]. Since then, some experimental attempts to learn using positive and unlabeled data have been tried using k-DNF or decision trees [13, 5]. However, those methods are not very useful for Web page classification problems because; (1) k-DNF or decision trees are not very tolerant with high dimensionality and sparse instance space, which are the properties of Web page classification, (2) their algorithms require knowledge of the proportion of positive instances within the universal set, which is not available in many classification problems, and (3) they perform poorer than traditional learning schemes given sufficient labeled data. The M-C (Mapping-Convergence) algorithm introduced in this paper relaxes the above three limitations in Web page classification from positive and unlabeled examples.

One-class SVMs have been recently developed, which distinguish one class of data from the rest of the feature space given only positive data set [19, 14]. One-class SVMs draw the class boundary of the positive data set in the feature space. However, due to lack of the information about negative data distribution, they require much larger amount of positive training data to induce accurate boundary, and their performance is dependent on the user parameters indicating how strictly the boundary should fit around the data. From our experiments of the one-class SVM using the LIBSVM[2], the one-class SVM performs initially very poor with the standard parameter setting, and even with careful parameter setting, the performance is much worse than that of our M-C algorithm because the one-class SVM does not utilize the distribution of unlabeled data.

Our approach is fundamentally different from previous approaches. The first stage of the M-C algorithm (called the *mapping stage*) is based on *1*-DNF, which was previously proven learnable from positive and unlabeled data [7]. The

[2]http://www.csie.ntu.edu.tw/ cjlin/libsvm

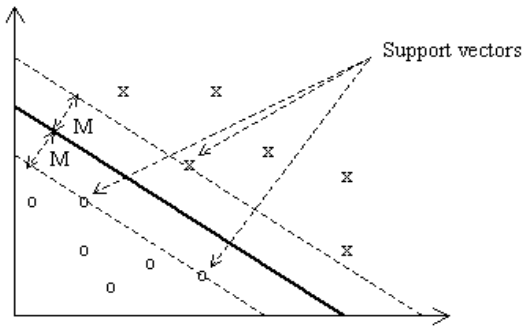


Figure 2: A graphical representation of a linear SVM in a two-dimensional case. (i.e. Only two features are considered.) M is the distance from the separator to the support vectors in feature space.

second stage of the M-C algorithm (called the *convergence stage*) uses SVM (Support Vector Machine) technology [3], especially the marginal property of SVMs.

3. MARGINAL PROPERTY OF SVM

As a binary classification algorithm, SVM gains increasing popularity because it has shown outstanding performance in many domains of classification problems [8, 10, 21]. 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 that other learning algorithms do not have, such as maximization of margin and nonlinear transformation of the input space to the feature space using kernel methods [3]. 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 of the positive and negative data in the feature space. Figure 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 non-resume 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 (the penalty imposed on training data that fall on the wrong side of the decision boundary). 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 a non-linear input space to a linear feature space. We used the linear kernel of SVM in our experiments because of it's speed and relatively high

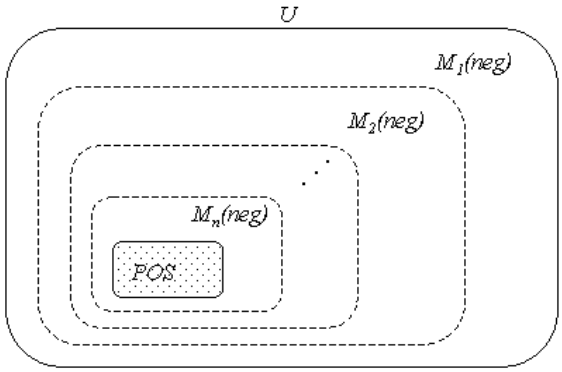


Figure 3: Strength of negative

classification accuracy. We discuss the usage of advanced kernel methods within our framework in Section 6.

4. THE 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.

What can we learn 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% of positive data but only in 10% 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 every possibly positive data point from the unlabeled data set, which leaves only *strongly negative data* – We call these *strong negatives*. For instance, we say *a strong negative* is a data point not having any of the positive features in the list. (In this case, the list is considered *1*-DNF.) In this way, we can extract strong negatives from the unlabeled data. This is what the *mapping stage* of the M-C algorithm accomplishes. However, using the list, we 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.

What can we do with the strong negatives to construct an accurate class boundary? We were given samples of *positive* and *unlabeled data*, and now we have *strong negatives* extracted from the unlabeled data through the mapping stage. If we construct a SVM from the positives and only the strong negatives, the class boundary would be far from accurate due to the insufficient negative training data. We use the marginal property of SVMs to refine the inaccurate boundary into the accurate one. This process is the *convergence stage* of the M-C algorithm. In this section, we explain the details of the M-C algorithm.

4.1 Definitions

The distance from the hyperplane (the class boundary) to a negative data point in the feature space is proportional to

its relative strength. For instance, consider a resume classifier. Assume that there are two negative data points (non-resume 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. The following definition quantizes the level of the negative strength into n discrete levels. (This is conceptual division of the strength. In real cases, the levels of the strength may be continuous.)

Definition 1. (*Map of Negative:* $M_i(neg)$) 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 3 visualizes the strength of negative.

$\cup_{i=1}^{n} M_i(neg)$ is equivalent to the set of the negative data points, everything that excludes positives in the universal set.

Definition 2. (*Subsumption of Positive:* $S_i(pos)$) Let $S_i(pos)$ be a set subsuming POS. Namely,

$$S_i(pos) = \bigcup_{k=i+1}^{n} M_k(neg) \cup POS$$

for i=0 to n-1 where $U = S_0(pos)$. Note that $S_i(pos) = S_{i+1}(pos) \cup M_{i+1}(neg)$, and $U = S_1(pos) \cup M_1(neg)$

4.2 Mapping-Convergence (M-C) algorithm

As alluded to earlier, the Mapping-Convergence (M-C) algorithm consists of two stages – (1) the *mapping stage* and (2) the *convergence stage*.

4.2.1 Mapping Stage

The mapping stage maps the strongest negative, $M_1(neg)$, from U using *1*-DNF learning without great concern for the quality of the mapping. The remaining data points, excluding the strongest negative ($M_1(neg)$) from U, would be $S_1(pos)$ which subsumes the positive as defined previously. k-DNF has been proven learnable from positive and unlabeled data in previous work [7], which supports the feasibility of the mapping stage execution when not overly concerned with the quality of the mapping. A condition for the mapping stage is that $M_1(neg)$ excludes true positive. The quality of the mapping is not critical to the performance of the M-C algorithm if this condition is not violated. We discuss this more in Section 4.3.

4.2.2 Convergence Stage

The *convergence stage* trains the SVM repeatedly to aggregate mapped negatives ($M_i(neg)$) as close as possible to the unbiased negatives (NEG). We illustrate this through the following example.

Example 1. (*Convergence*) 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 a set of the other pages in the university site excluding faculty pages (unbiased negative pages). POS and U are assumed to be given, and NEG is initially set to *null* because negative examples are not given. We assume that a university site has the following four different levels (n=4) of the negative strength.

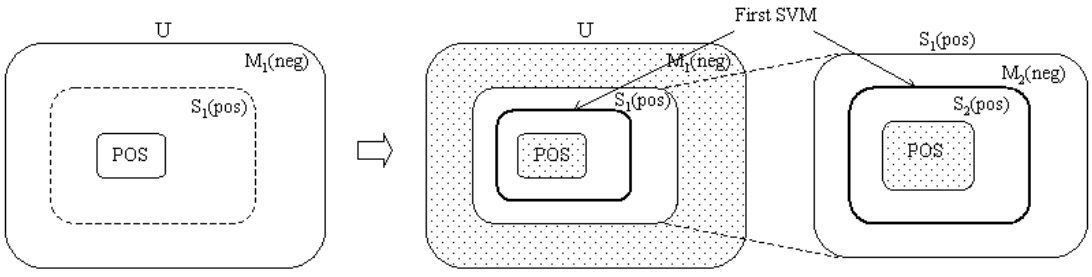
$M_4(neg)$: the weakest negative pages considered most similar to the faculty pages (e.g. staff pages)
$M_3(neg)$: the secondly weakest negative pages which are less similar to the faculty pages (e.g. student pages)
$M_2(neg)$: the negative pages not very similar to the faculty pages (e.g. project or course pages)
$M_1(neg)$: the strongest negative pages considered most disparate from the faculty pages (e.g. information or facility pages)

Next, consider how we fill the NEG with unbiased negative data extracted from U. Let's say the *1*-DNF identifies only the strongest negative, $M_1(neg)$ (e.g. information or facility pages). We save the $M_1(neg)$ into NEG, and the rest of U will be $S_1(pos)$. ($S_1(pos) = \cup_{k=2}^{4} M_k(neg) \cup POS$.) We train a SVM with the NEG (currently containing $M_1(neg)$) and the given POS. The SVM generates a hyperplane between NEG and POS, which *keeps a maximal margin between them* in the feature space. (See Figure 4.(a).) When, using the SVM, we test $S_1(pos)$, the SVM divides the $S_1(pos)$ into $M_2(neg)$ (e.g. project or course pages) and $S_2(pos)$. ($S_2(pos) = POS \cup M_4(neg) \cup M_3(neg)$.) We now accumulate the $M_2(neg)$ into NEG, and then we re-train the SVM with the NEG (currently containing $M_1(neg) \cup M_2(neg)$) and the given POS. The SVM generates another hyperplane between NEG and POS also keeping maximal margin between them in the feature space. (See Figure 4.(b).) When we test $S_2(pos)$ using the SVM, the SVM divides the $S_2(pos)$ into $M_3(neg)$ (e.g. student pages) and $S_3(pos)$. ($S_3(pos) = POS \cup M_4(neg)$.) We accumulate the $M_3(neg)$ into NEG again, and re-train the SVM with the NEG (currently containing $\cup_{k=1}^{3} M_k(neg)$) and the given POS. We iterate these processes until the $M_i(neg)$ becomes empty set. The SVM constructed at the end of the process will be close to the SVM constructed from positive and unbiased negative data because NEG will converge into the unbiased negative data in the universal set, U. (U was assume to be an unbiased sample of the universal set.) Figure 5 shows the outline of the M-C algorithm, and Figure 6 shows the corresponding data flow of each part of the algorithm.

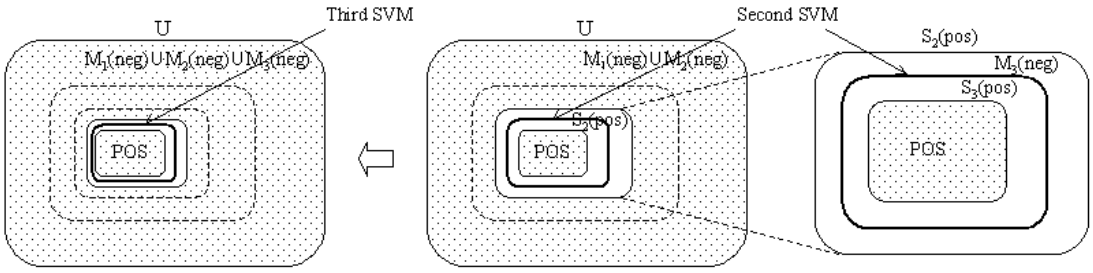
Example 1 may discretize too much the boundary of each level of the negative strength. However, in real cases with continuous levels, as you see in Figure 4, the marginal property of SVMs urges the hyperplane to converge into the real boundary of the two classes (positive and negative classes) in the feature space. Our experiments in Section 5 show that the M-C algorithm with positive and unlabeled data actually performs as well as the traditional SVM with labeled data.

4.3 *1*-DNF Mapping

The goal of the mapping stage is to subsume positive without great concern for the quality of the mapping as we discussed in Section 4. We do this by building a disjunction list of positive features, which is equivalent to *1*-DNF. After we construct the *1*-DNF of positive features, we map the subsuming positive ($S_1(pos)$) and the strongest negative ($M_1(neg)$) by filtering the universal set (U) through the *1*-



(a) Training first SVM (from M_1(neg) and POS) that divides S_1(pos) into M_2(neg) and S_2(pos)



(b) Training second SVM (from M_1(neg)$\cup M_2$(neg) and POS) that divides S_2(pos) into M_3(neg) and S_3(pos)

Figure 4: Marginal convergence of SVM

DNF. (The beginning of Section 4 discuss the intuition of the mapping stage.) The choice of k-DNF, however, not critical to the final accuracy of the M-C algorithm as long as $S_1(pos)$ subsumes POS, because $\cup M_i(neg)$ converges into the unbiased negatives through the iterations regardless of the quality of the initial mapping. The poor quality of the initial mapping would increase the number of the iterations in the algorithm, which ends up longer training time, but the final accuracy would be the same. Our experiments show that classification accuracy of the M-C algorithm converges into that of the traditional SVM trained from labeled data in every class no matter how bad the initial mapping is. (See Figure 7 and 8 in Section 5.2.)

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 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 page, we used DMOZ[3], 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 data. We also manually collected 368 personal homepages, 192 college admission pages, and 188 resume pages to classify the three 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. The PEBL does not require collecting negative data to construct classifiers.) For instance, for personal homepage class, we used 183 positive and 2388 unlabeled data for training, and used 185 positive and 449 negative data for testing.

Experiment 2. (University computer science department) The WebKB data set [4] 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 others. 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 data, 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 data for training, and used 383 positive and 729 negative data for testing.

[3]Open Directory Project, http://dmoz.org

Input:

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

Output:

- a SVM

Algorithm:

- *1*-DNF := construct_*1*-DNF (POS, U); // ...①
  $M_1(neg)$, $S_1(pos)$:= *1*-DNF.classify (U); // ...②
  $NEG := \varnothing$; i := 1;
  do {
  - $NEG := NEG \cup M_i(neg)$;
    SVM := construct_SVM (POS, NEG); // ...③
    $M_{i+1}(neg)$, $S_{i+1}(pos)$:= SVM.classify ($S_i(pos)$);
    //...④
    i := i + 1;

  } while ($M_i(neg) \neq \varnothing$);
  return SVM;

Rationale:

- *1*-DNF learns from POS and U...①, and extracts the the strongest negative ($M_1(neg)$) from U...②. (The remainder is $S_1(pos)$.) Save the $M_1(neg)$ into NEG, and construct a SVM from POS and the NEG...③. The SVM classifies $S_1(pos)$ into the secondly strong negative ($M_2(neg)$) and the remainder ($S_2(pos)$)...④. Accumulate the $M_2(neg)$ into NEG, and construct a SVM again from POS and NEG...③. The SVM classifies $S_2(pos)$ into $M_3(neg)$ and $S_3(pos)$...④. We iterate these processes until $M_i(neg)$ becomes empty set.

Analysis:

- Training time is proportional to the time of training a SVM multiplied by i (# of iterations).
- i (# of iterations) is logarithmic to the size of negative space because ③ and ④ include half of the rest of space at each iteration.

Figure 5: Mapping-Convergence (M-C) algorithm

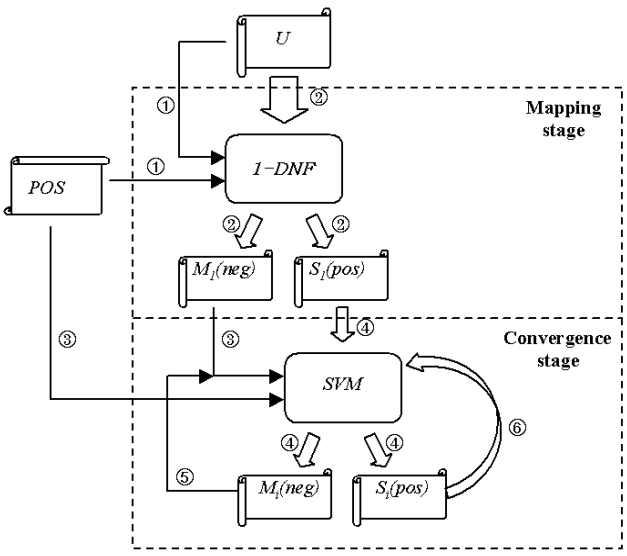


Figure 6: Data flow diagram of the Mapping-Convergence (M-C) algorithm

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 appears 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[4]. As we discussed in Section 3, we used linear kernel method because it is simple and efficient. We discuss more about the usage of advanced kernel methods in Section 6. For the parameter, C (the penalty imposed on training data that fall on the wrong side of the decision boundary), we used the default parameter, $[ave./x*x]^{-1}$ of the SVM light. 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 mathematical foundation of SVM 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 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 first show the performance comparison between PEBL 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 PEBL. 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 the same testing documents using those two SVMs – PEBL 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 PEBL. In most cases, PEBL without negative training data performs almost as well as the traditional SVM with manually labeled training data. For example, when we

[4]http://svmlight.joachims.org

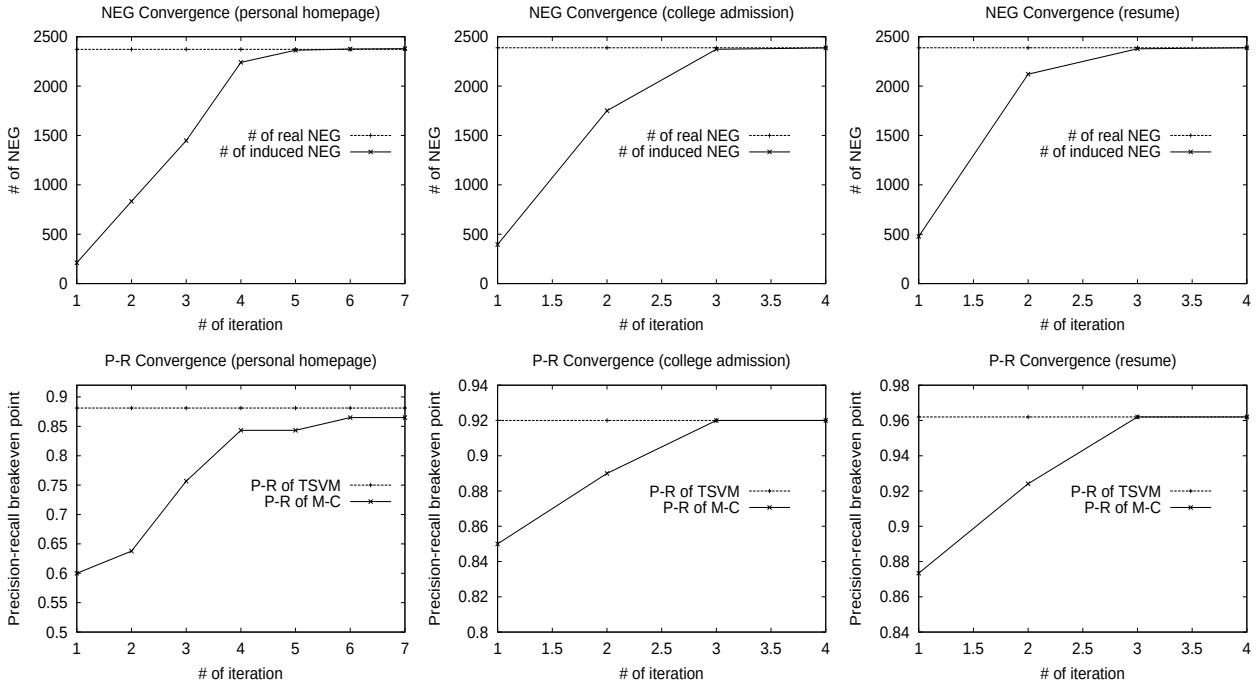


Figure 7: Convergence of negatives (NEG) and performance (P-R: precision-recall breakeven point) when the universal set is the Internet. TSVM indicates the traditional SVM constructed from manually labeled (positive and unbiased negative) data.

Table 1: Precision-recall breakeven points (P-R) showing performance of PEBL (Positive Example Based Framework) and TSVM (the Traditional SVM trained from manually labeled data) in the two universal sets (U). The number of iterations to the convergence in PEBL is shown in parentheses.

U	Class	TSVM	PEBL
The Internet	homepage	88.11	86.49 (7)
	admission	92.0	92.0 (4)
	resume	96.2	96.2 (4)
CS Department	student	94.74	94.23 (14)
	project	86.67	86.06 (12)
	faculty	92.95	91.12 (11)

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 PEBL with only the 109 resume pages without non-resume pages, it gives also 96.2% P-R.

Figure 7 and 8 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. For instance, consider the graphs of the first column (personal homepage class) in Figure 7. The number of induced negatives at the first iteration is around 250 (shown on the first row of the graph), and the P-R of the SVM trained from positive and those 250 negatives is 0.60 (shown on the second row of the graph). At the second iteration, the number of induced negatives is around 800, and the SVM trained from positive and those 800 negatives gives 0.64 P-R. Likewise, at the seventh iteration, the number of induced negatives is almost the same as the number of real unbiased negatives, and also the P-R at the point is as high as the P-R of the traditional SVM (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 Figure 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.

6. DISCUSSION

In this section, we discuss several important issues to consider regarding learning algorithm and PEBL framework.

`1. Possible extension of 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 real boundary of the two (positive and negative) class. 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.

`2. Choice of kernel functions.`

SVMs provide nonlinear transformation of input space to

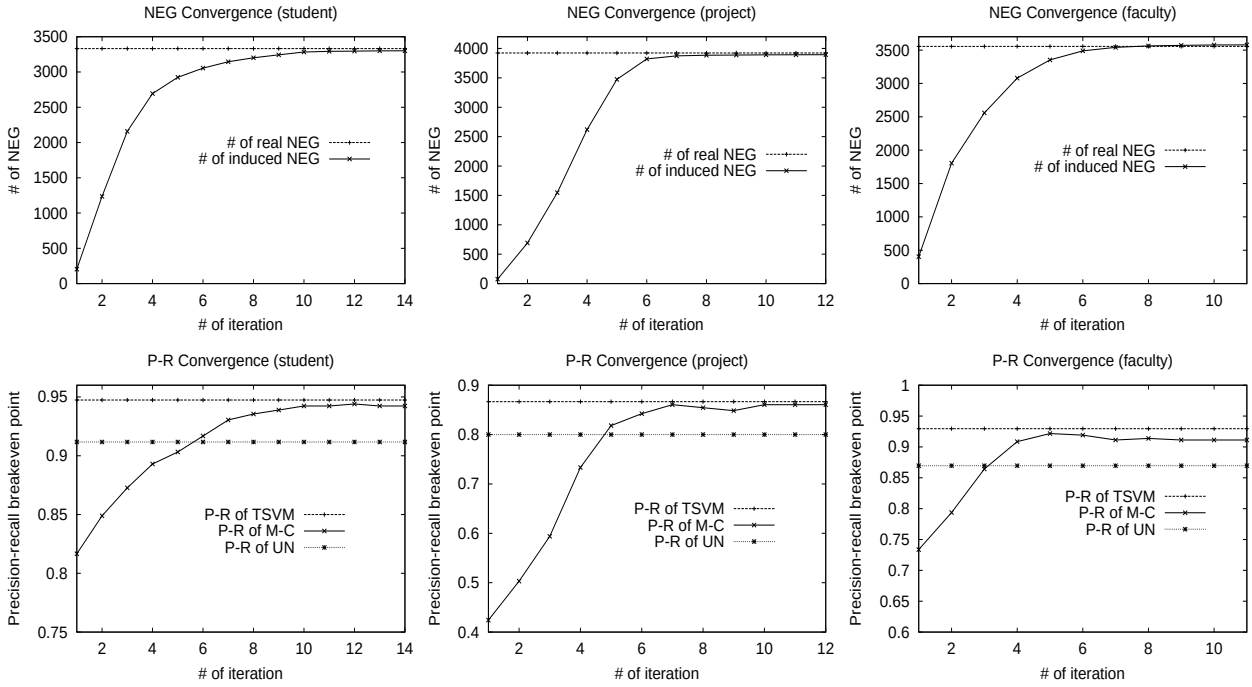


Figure 8: Convergence of negatives (NEG) and performance (P-R: precision-recall breakeven point) when the universal set is computer science department sites. TSVM indicates the traditional SVM constructed from manually labeled (positive and unbiased negative) data. UN indicates the SVM constructed from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

feature space using advanced kernels (i.e. polynomial or gaussian kernels) [3]. *Polynomial kernel* combines input features to create high dimensional features in feature space. As the degree of polynomial kernel grows, higher dimensional function is used to try to fit more training data, which ends up overfitting at some point. In our experiments with Web pages, even the second degree of polynomial kernel degraded the accucary due to the overfitting. For Web page classification, polynomial kernel may not be a good choice because of its inefficiency and easy overfitting. From our experiments with *Gaussian kernel*, it shows $1 \sim 3\%$ higher accuracy than linear kernel overall but with careful setting of the parameters, γ and C. Default setting of γ and C usually gives poor accuracy. The usage of gaussian kernel in the M-C algorithm would increase the accuracy but also increase the training time multiply because searching the best parameters requires time-consuming process such as cross-validation. We used linear kernel in the experiments because it is efficient and shows also good accuracy. Identifying or building proper kernel functions for specific problems is still ongoing research.

`3. Further improvement of algorithm performance.` The M-C algorithm takes several times longer to train one class than traditional SVM does since the iteration of training SVM in the M-C algorithm has to be serialized due to data dependency between adjacent iterations. However, only part of the NEG (the negatives accumulated each iteration) is dependent between each iteration, which leaves a 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 with the enhanced architecture to increase classification accuracy for specific problems could be good future works. Once training is done (a classifier is constructed), however, the speed of testing (classifying) is equivalent to traditional SVM, which makes our framework practical in many real-world problems.

7. SUMMARY AND CONCLUSIONS

Web page classification is one of the essential techniques for Web mining. Specifically, classifying Web pages of a user-interesting class is the first step of mining interesting information from the Web. However, constructing a classifier for an interesting class requires laborious pre-processing such as collecting positive and negative training examples. In particular, *collecting negative training examples* requires arduous work and special caution to avoid biasing them. The *Positive Example Based Learning (PEBL)* framework for Web page classification eliminates the need for manually collecting negative training examples in pre-processing. 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 when the M-C algorithm uses the same amount of positive examples as that of traditional SVM, the M-C algorithm achieves classification accuracy as high as that of traditional SVM. PEBL framework using the M-C algorithm contributes to automating pre-processing of Web page classification without much loss of classification accuracy.

8. REFERENCES

[1] E. L. Allwein, R. E. Schapire, and Y. Singer. Reducing multiclass to binary: a unifying approach for margin classifiers. *Journal of Machine Learning Research*, 1(2000):113–141, 2000.

[2] H. Chen, C. Schuffels, and R. Orwig. Internet categorization and search: a machine learning approach. *Journal of Visual Communications and Image Representation*, 7(1):88–102, 1996.

[3] C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, (20):273–297, 1995.

[4] M. Craven, D. Dipasquo, and D. Freitag. Learning to extract symbolic knowledge from the world wide web. In *AAAI*, 1998.

[5] F. DeComite, F. Denis, and R. Gilleron. Positive and unlabeled examples help learning. In *Algorithmic Learning Theory (ALT)*, 1999.

[6] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, Series B*, 39:1–38, 1977.

[7] F. Denis. Pac learning from positive statistical queries. In *ALT*, 1998.

[8] S. Dumais and H. Chen. Hierarchical classification of web content. In *SIGIR*, 2000.

[9] E. J. Glover, G. W. Flake, and S. Lawrence. Improving category specific web search by learning query modifications. In *Symposium on Applications and the Internet (SAINT 2001)*, 2001.

[10] T. Joachims. Text categorization with support vector machines. In *ECML*, 1998.

[11] T. Joachims. Transductive inference for text classification using support vector machines. In *ICML*, 1999.

[12] A. Kruger, C. L. Giles, and E. Glover. Deadliner: Building a new niche search engine. In *Conference on Information and Knowledge Management (CIKM)*, 2000.

[13] F. Letouzey, F. Denis, and R. Gilleron. Learning from positive and unlabeled examples. In *ALT*, 2000.

[14] L. M. Manevitz and M. Yousef. One-class svms for document classification. *Journal of Machine Learning Research*, 2(Dec):139–154, 2001.

[15] H. Mase. Experiments on automatic web page categorization for ir system. Technical report, Stanford University, http://citeseer.nj.nec.com/164846.html, 1998.

[16] E. N. Mayoraz. Multiclass classification with pairwise coupled neural networks or support vector machines. In *ICANN*, 2001.

[17] K. Nigam. Text classification from labeled and unlabeled documents using em. *Machine Learning*, 39(2/3):103–134, 2000.

[18] H.-J. Oh, S.-H. Myaeng, and M.-H. Lee. A practical hypertext categorization method using links and incrementally available class information. In *SIGIR*, 2000.

[19] D. M. J. Tax and R. P. W. Duin. Uniform object generation for optimizating one-class classifiers. *Journal of Machine Learning Research*, 2(Dec):155–173, 2001.

[20] W.-C. Wong and A. W.-C. Fu. Finding structure and characteristics of web documents for classification. In *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000.

[21] Y. Yang and Y. Lui. A re-examination of text categorization methods. In *SIGIR*, 1999.

[22] J. Yi and N. Sundaresan. A classifier for semi-structured documents. In *KDD*, 2000.