Using Labeled and Unlabeled Data to Learn Drifting Concepts

Ralf Klinkenberg
University of Dortmund, LS Informatik VIII
44221 Dortmund, Germany
E-Mail: klinkenberg@ls8.cs.uni-dortmund.de
WWW: http://www-ai.cs.uni-dortmund.de/

Abstract

For many learning tasks, where data is collected over an extended period of time, one has to cope two problems. The distribution underlying the data is likely to change and only little labeled training data is available at each point in time. A typical example is information filtering, i. e. the adaptive classification of documents with respect to a particular user interest. Both the interest of the user and the document content change over time. A filtering system should be able to adapt to such concept changes. Since users often give little feedback, a filtering system should also be able to achieve a good performance, even if only few labeled training examples are provided. This paper proposes a method to recognize and handle concept changes with support vector machines and to use unlabeled data to reduce the need for labeled data. The method maintains windows on the training data, whose size is automatically adjusted so that the estimated generalization error is minimized. The approach is both theoretically well-founded as well as effective and efficient in practice. Since it does not require complicated parameterization, it is simpler to use and more robust than comparable heuristics. Experiments with simulated concept drift scenarios based on real-world text data compare the new method with other window management approaches and show that it can effectively select an appropriate window size in a robust way. In order to achieve an acceptable performance with fewer labeled training examples, the proposed method exploits unlabeled examples in a transductive way.

1 Introduction

Machine learning methods are often applied to problems, where data is collected over an extended period of time. In many real-world applications this introduces the problem that the distribution underlying the data is likely to change over time. For example, companies collect an increasing amount of data like sales figures and customer data to find patterns in the customer behaviour and to predict future sales. As the customer behaviour tends to change over time, the model underlying successful predictions should be adapted accordingly.

The same problem occurs in information filtering, i.e. the adaptive classification of documents with respect to a particular user interest. Information filtering techniques are used, for example, to build personalized news filters, which learn about the news-reading preferences of a user or to filter e-mail. Both the interest of the user and the document content change over time. A filtering system should be able to adapt to such concept changes. A second problem in many real world applications is that only little labeled training data is available. This also applies to the information filtering domain. Since users often give only partial feedback, a filtering system should also be able to achieve a good performance, even if only few labeled training examples are provided.

This paper proposes a method for detecting and handling concept changes with support vector machines extending the approach described in [Klinkenberg and Joachims, 2000] by using unlabeled data to reduce the need for labeled data. The approach has a clear theoretical motivation and does not require complicated parameter tuning. After reviewing other work on adaptation to changing concepts and shortly describing inductive and transductive support vector machines, this paper explains the new window adjustment approach and evaluates it in three simulated concept drift scenarios on real-world text data. The experiments show that the approach effectively selects an appropriate window size and results in a low predictive error rate when given complete feedback. When the amount of feedback is reduced and no unlabeled data is used, the performance of the system decreases. Since empirical results (see e.g. [Nigam *et al.*, 2000], [Joachims, 1999b], [Lanquillon, 2000]) show that unlabeled data can help to significantly improve the performance of text classifiers, especially in case of few labeled examples, the use of unlabeled data can also be expected to improve the performance of the proposed information filtering approach and to let its performance drop more gracefully for reduced amounts of feedback. As pointed out in [Joachims, 1999b], it is well known in information retrieval that words in natural language occur in strong co-occurence patterns (see [van Rijsbergen, 1977]). While some words are likely to occur in one document, others are not. This type of information is independent of the document labels and can be exploited, if unlabeled data is used.

2 Concept Drift

Throughout this paper, we study the problem of concept drift for the pattern recognition problem in the following framework. Each example $\vec{z} = (\vec{x}, y)$ consists of a feature vector $\vec{x} \in \mathbf{R}^N$ and a label $y \in \{-1, +1\}$ indicating its classification. Data arrives over time in batches. Without loss of generality these batches are assumed to be of equal size, each containing m examples.

$$\vec{z}_{(1,1)}, \ldots, \vec{z}_{(1,m)}, \vec{z}_{(2,1)}, \ldots, \vec{z}_{(2,m)}, \cdots, \vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)}, \vec{z}_{(t+1,1)}, \ldots, \vec{z}_{(t+1,m)}$$

$\vec{z}_{(i,j)}$ denotes the j-th example of batch i. For each batch i the data is independently identically distributed (i.i.d.) with respect to a distribution $\Pr_i(\vec{x}, y)$. Depending on the amount and type of concept drift, the example distribution $\Pr_i(\vec{x}, y)$ and $\Pr_{i+1}(\vec{x}, y)$ between batches will differ. The goal of the learner $\mathcal{L}$ is to sequentially predict the labels of the next batch. For example, after batch t the learner can use any subset of the training examples from batches 1 to t to predict the labels of batch $t + 1$. The learner aims to minimize the cumulated number of prediction errors.

If the user gives only partial feedback, i.e. assigns labels y_j only to some of the $\vec{x}_j$, some of the labels y_j may be unknown. While supervised learning algorithms that are not able to use unlabeled data (in addition to labeled data) have to disregard the corrsponding (training) examples $\vec{z}_j = (\vec{x}_j, y_j)$, other (semi-)supervised learners are able to exploit the information captured in this unlabeled examples (see sections 3.2 and 3.3 for exampeles), which is basically information about $\Pr_i(\vec{x})$. The algorithms described in this section, however, are not able to use unlabeled data.

In machine learning, changing concepts are often handled by time windows of fixed or adaptive size on the training data [Mitchell *et al.*, 1994; Widmer and Kubat, 1996; Klinkenberg and Renz, 1998] or by weighting data or parts of the hypothesis according to their age and/or utility for the classification task [Kunisch, 1996; Taylor *et al.*, 1997]. The latter approach of weighting examples has already been used for information filtering in the incremental relevance feedback approaches of [Allan, 1996] and [Balabanovic, 1997]. In this paper, the earlier approach maintaining a window of adaptive size is explored.

For windows of fixed size, the choice of a *"good" window size* is a compromise between fast adaptivity (small window) and good generalization in phases without concept change (large window). The basic idea of *adaptive window management* is to adjust the window size to the current extent of concept drift.

The task of learning drifting or time-varying concepts has also been studied in computational learning theory. Learning a changing concept is infeasible, if no restrictions are imposed on the type of admissible concept changes,[1] but drifting concepts are provably efficiently learnable (at least for certain concept classes), if the rate or the extent of drift is limited in particular ways.

[Helmbold and Long, 1994] assume a possibly permanent but slow concept drift and define the *extent of drift* as the probability that two subsequent concepts disagree on a randomly drawn example. Their results include an upper bound for the extend of drift maximally tolerable by any learner and algorithms that can learn concepts that do not drift more than a certain constant extent of drift. Furthermore they show that it is sufficient for a learner to see a fixed number of the most recent examples. Hence a window of a certain minimal fixed size allows to learn concepts for which the extent of drift is appropriately limited.

While Helmbold and Long restrict the extend of drift, [Kuh *et al.*, 1991] determine a maximal *rate of drift* that is acceptable by any learner, i. e. a maximally acceptable frequency of concept changes, which implies a lower bound for the size of a fixed window for a time-varying concept to be learnable, which is similar to the lower bound of Helmbold and Long.

In practice, however, it usually cannot be guaranteed that the application at hand obeys these restrictions, e.g. a reader of electronic news may change his interests (almost) arbitrarily often and radically. Furthermore the large time window sizes, for which the theoretical results hold, would be impractical. Hence more application oriented approaches rely on far smaller windows of fixed size or on window adjustment heuristics that allow far smaller window sizes and usually perform better than fixed and/or larger windows (see e.g. [Widmer and Kubat, 1996] or [Klinkenberg and Renz, 1998]). While these heuristics are intuitive and work well in their particular application domain, they usually require tuning their parameters, are often not transferable to other domains, and lack a proper theoretical foundation.

3 Support Vector Machines

3.1 (Inductive) Support Vector Machines (SVMs)

The window adjustment approach described in this paper uses support vector machines [Vapnik, 1998] as their core learning algorithm. Support vector machines are based on the *structural risk minimization* principle [Vapnik, 1998] from statistical learning theory. In their basic form, SVMs learn linear decision rules

$$h(\vec{x}) = sign\{\vec{w} \cdot \vec{x} + b\} = \begin{cases} +1, & \text{if } \vec{w} \cdot \vec{x} + b > 0 \\ -1, & \text{else} \end{cases} \quad (1)$$

described by a weight vector $\vec{w}$ and a threshold b. The idea of structural risk minimization is to find a hypothesis h for which one can guarantee the lowest probability of error. For SVMs, [Vapnik, 1998] shows that this goal can be translated into finding the hyperplane with maximum soft-margin.[2] Computing this hyperplane is equivalent to solving the following optimization problem.

Optimization Problem 1 (SVM (primal))

$$\textit{Minimize:} \quad V(\vec{w}, b, \vec{\xi}) = \frac{1}{2}\vec{w} \cdot \vec{w} + C \sum_{i=1}^{n} \xi_i \quad (2)$$

$$\textit{subject to:} \quad \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i \quad (3)$$

$$\forall_{i=1}^{n} : \xi_i > 0 \quad (4)$$

[1] E.g. a function randomly jumping between the values one and zero cannot be predicted by any learner with more than 50% accuracy.

[2] See [Burges, 1998] for an introduction to SVMs.

In this optimization problem, the Euclidean length $||\vec{w}||$ of the weight vector is inversely proportional to the soft-margin of the decision rule. The constraints (3) require that all training examples are classified correctly up to some slack ξ_i. If a training example lies on the "wrong" side of the hyperplane, the corresponding ξ_i is greater or equal to 1. Therefore $\sum_{i=1}^{n} \xi_i$ is an upper bound on the number of training errors. The factor C in (2) is a parameter that allows trading-off training error vs. model complexity.

For computational reasons it is useful to solve the Wolfe dual [Fletcher, 1987] of optimization problem 1 instead of solving optimization problem 1 directly [Vapnik, 1998].

Optimization Problem 2 (SVM (dual))

$$\textit{Minimize: } W(\vec{\alpha}) = -\sum_{i=1}^{n} \alpha_i + \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j (\vec{x}_i \cdot \vec{x}_j) \qquad (5)$$

$$\textit{subject to: } \sum_{i=1}^{n} y_i \alpha_i = 0 \qquad (6)$$

$$\forall_{i=1}^{n} : 0 \leq \alpha_i \leq C \qquad (7)$$

In this paper, SVM^{light} [Joachims, 1999a] is used for computing the solution of this optimization problem.[3] Support vectors are those training examples $\vec{x}_i$ with $\alpha_i > 0$ at the solution. From the solution of optimization problem 2 the decision rule can be computed as

$$\vec{w} \cdot \vec{x} = \sum_{i=1}^{n} \alpha_i y_i (\vec{x}_i \cdot \vec{x}) \quad \text{and} \quad b = y_{usv} - \vec{w} \cdot \vec{x}_{usv} \qquad (8)$$

The training example $(\vec{x}_{usv}, y_{usv})$ for calculating b must be a support vector with $\alpha_{usv} < C$. Finally, the training losses ξ_i can be computed as $\xi_i = max(1 - y_i \left[\vec{w} \cdot \vec{x}_i + b\right], 0)$.

For both solving optimization problem 2 as well as applying the learned decision rule, it is sufficient to be able to calculate inner products between feature vectors. Exploiting this property, Boser et al. introduced the use of kernels $\mathcal{K}(\vec{x}_1, \vec{x}_2)$ for learning non-linear decision rules. Depending on the type of kernel function, SVMs learn polynomial classifiers, radial basis function (RBF) classifiers, or two layer sigmoid neural nets. Such kernels calculate an inner-product in some feature space and replace the inner-product in the formulas above.

3.2 Transductive Support Vector Machines (TSVMs)

The setting of transductive inference was introduced by Vapnik (see for example [Vapnik, 1998]). The description here follows the description in [Joachims, 1999b]. For a learning task $P(\vec{x}, y) = P(y|\vec{x})P(\vec{x})$ the learner $\mathcal{L}$ is given a hypothesis space H of functions $h : X \longrightarrow \{-1, 1\}$ and an i.i.d. sample S_{train} of n training examples

$$(\vec{x}_1, y_1), (\vec{x}_2, y_2), ..., (\vec{x}_n, y_n) \qquad (9)$$

Each training example consists of a document vector $\vec{x} \in X$ and a binary label $y \in \{-1, +1\}$. In contrast to the inductive setting, the learner is also given an i.i.d. sample S_{test} of k test examples

$$\vec{x}^*_1, \vec{x}^*_2, ..., \vec{x}^*_k \qquad (10)$$

from the same distribution. The transductive learner $\mathcal{L}$ aims to select a function $h_{\mathcal{L}} = \mathcal{L}(S_{train}, S_{test})$ from H using S_{train} and S_{test} so that the expected number of erroneous predictions

$$R(\mathcal{L}) = \int \frac{1}{k}\sum_{i=1}^{k} \Theta(h_{\mathcal{L}}(\vec{x}^*_i), y^*_i) dP(\vec{x}_1, y_1) \cdots dP(\vec{x}^*_k, y^*_k)$$

on the test examples is minimized. $\Theta(a, b)$ is zero if $a = b$, otherwise it is one. As shown in [Vapnik, 1998], for linearly separable problems this leads to the following optimization problem.

Optimization Problem 3 (Transductive SVM (lin. sep. case))
Minimize over $(y^*_1, ..., y^*_k, \vec{w}, b)$*:*

$$\frac{1}{2}||\vec{w}||^2$$

$$\textit{subject to: } \quad \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1$$

$$\forall_{j=1}^{k} : y^*_j[\vec{w} \cdot \vec{x}^*_j + b] \geq 1$$

Solving this problem means finding a labelling $y^*_1, ..., y^*_k$ of the test data and a hyperplane $< \vec{w}, b >$, so that this hyperplane separates both training and test data with maximum margin. To be able to handle non-separable data, we can introduce slack variables ξ_i similarly to the way we do with inductive SVMs.

Optimization Problem 4 (Transductive SVM (non-sep. case))
Minimize over $(y^*_1, ..., y^*_k, \vec{w}, b, \xi_1, ..., \xi_n, \xi^*_1, ..., \xi^*_k)$*:*

$$\frac{1}{2}||\vec{w}||^2 + C\sum_{i=0}^{n} \xi_i + C^* \sum_{j=0}^{k} \xi^*_j$$

$$\textit{subject to: } \quad \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i$$

$$\forall_{j=1}^{k} : y^*_j[\vec{w} \cdot \vec{x}^*_j + b] \geq 1 - \xi^*_j$$

$$\forall_{i=1}^{n} : \xi_i > 0$$

$$\forall_{j=1}^{k} : \xi^*_j > 0$$

C and C^* are parameters set by the user. They allow trading off margin size against misclassifying training examples or excluding test examples. [Joachims, 1999b] proposes an algorithm to solve this optimization problem efficiently. In this paper, SVM^{light} is also used for computing the solution of this optimization problem.[3]

3.3 Other Approaches for Using Unlabeled Data

Besides of the TSVMs [Joachims, 1999b] described above, there are also other (semi-)supervised methods for exploiting unlabeled data. Early empirical results using transduction can be found in [Vapnik and Sterin, 1977]. More recently, Bennett [Bennett, 1999] showed small improvements for some of the standard UCI datasets. For ease of computation, she conducted the experiments only for a linear-programming approach which minimizes the L_1 norm instead of L_2 and prohibits the use of kernels. Connecting to concepts of algorithmic randomness, [Gammerman *et al.*, 1998] presented an

[3] SVM^{Light} is available at http://www-ai.informatik.uni-dortmund.de/svm_light

approach to estimating the confidence of a prediction based on a transductive setting.

Nigam et al. [Nigam *et al.*, 1998; 2000] proposed another approach to using unlabeled data for text classification. They use a multinomial Naive Bayes classifier and incorporate unlabeled data using the EM-algorithm. One problem with using Naive Bayes is that its independence assumption is clearly violated for text. Nevertheless, using EM showed substantial improvements over the performance of a regular Naive Bayes classifier. [Lanquillon, 2000] describes an extension of this EM-based framework to an EM-style framwork for arbitrary (text) classifiers.

Blum and Mitchell's work on co-training [Blum and Mitchell, 1998] uses unlabeled data in a particular setting. They exploit the fact that, for some problems, each example can be described by multiple representations. WWW-pages, for example, can be represented as the text on the page and/or the anchor texts on the hyperlinks pointing to this page. Blum and Mitchell develop a boosting scheme which exploits a conditional independence between these representations.

4 Window Adjustment by Optimizing Performance

The method proposed in this paper is an extension of the approach described in [Klinkenberg and Joachims, 2000]. This approach to handling drift in the distribution of examples uses a window on the training data. This window should include only those (labeled) examples which are sufficiently "close" to the current target concept. Assuming the amount of drift increases with time, the window includes the last n training examples. Previous approaches used similar windowing strategies. Their shortcomings are that they either fix the window size [Mitchell *et al.*, 1994] or involve complicated heuristics [Widmer and Kubat, 1996; Klinkenberg and Renz, 1998]. A fixed window size makes strong assumptions about how quickly the concept changes. While heuristics can adapt to different speed and amount of drift, they involve many parameters that are difficult to tune. Here, we present an approach to selecting an appropriate window size that does not involve complicated parameterization. They key idea is to select the window size so that the estimated generalization error on new examples is minimized. To get an estimate of the generalization error we use a special form of $\xi\alpha$-estimates [Joachims, 2000]. $\xi\alpha$-estimates are a particularly efficient method for estimating the performance of an SVM.

4.1 $\xi\alpha$-Estimators

$\xi\alpha$-estimators are based on the idea of leave-one-out estimation [Lunts and Brailovskiy, 1967]. The leave-one-out estimator of the error rate proceeds as follows. From the training sample $S_{train} = ((\vec{x}_1, y_1), \cdots, (\vec{x}_n, y_n))$ the first example $(\vec{x}_1, y_1)$ is removed. The resulting sample $S^{\backslash 1}_{train} = ((\vec{x}_2, y_2), \cdots, (\vec{x}_n, y_n))$ is used for training, leading to a classification rule $h^{\backslash 1}_{\mathcal{L}}$. This classification rule is tested on the held out example $(\vec{x}_1, y_1)$. If the example is classified incorrectly it is said to produce a leave-one-out error. This process is repeated for all training examples. The number of leave-one-out errors divided by n is the leave-one-out estimate of the generalization error.

While the leave-one-out estimate is usually very accurate, it is very expensive to compute. With a training sample of size n, one must run the learner n times. $\xi\alpha$-estimators overcome this problem using an upper bound on the number of leave-one-out errors instead of calculating them brute force. They owe their name to the two arguments they are computed from. $\vec{\xi}$ is the vector of training losses at the solution of the primal SVM training problem. $\vec{\alpha}$ is the solution of the dual SVM training problem. Based on these two vectors — both are available after training the SVM at no extra cost — the $\xi\alpha$-estimators are defined using the following two counts. With R^2_Δ being the maximum difference of any two elements of the Hessian (i.e. $R^2_\Delta \geq \max_{\vec{x},\vec{x}'}(\mathcal{K}(\vec{x},\vec{x}) - \mathcal{K}(\vec{x},\vec{x}'))$),

$$d = |\{i : (\alpha_i R^2_\Delta + \xi_i) \geq 1\}| \qquad (11)$$

counts the number of training examples, for which the quantity $\alpha_i R^2_\Delta + \xi_i$ exceeds one. Since the document vectors are normalized to unit length in the experiments described in this paper, here $R^2_\Delta = 1$. It is proven in [Joachims, 2000] that d is an approximate upper bound on the number of leave-one-out errors in the training set. With n as the total number of training examples, the $\xi\alpha$-estimators of the error rate is

$$Err^n_{\xi\alpha}(h_{\mathcal{L}}) = \frac{|\{i : (\alpha_i R^2_\Delta + \xi_i) \geq 1\}|}{n} \qquad (12)$$

The theoretical properties of this $\xi\alpha$-estimator are discussed in [Joachims, 2000]. It can be shown that the estimator is pessimistically biased, overestimating the true error rate on average. Experiments show that the bias is acceptably small for text classification problems and that the variance of the $\xi\alpha$-estimator is essentially as low as that of a holdout estimate using twice as much data. It is also possible to design similar estimators for precision and recall, as well as combined measures like $F1$ [Joachims, 2000].

4.2 Window Adjustment Algorithm

The method described here is an extension of the approach described in [Klinkenberg and Joachims, 2000] in that it exploits unlabeled data. Its basic idea is to first use the algorithm described in [Klinkenberg and Joachims, 2000] to find the right window size on the labeled training data, $win_{labeled}$, using $\xi\alpha$-estimates for an inductive SVM, and to then use an almost identical algorithm to determine a good window size on the unlabeled data, $win_{unlabeled}$, on the same stream of documents using $\xi\alpha$-estimates for a transductive SVM to estimate the prediciton error on the test set, leaving the window size $win_{labeled}$ unchanged. The following two subsections describe these two parts of the algorithm.

Why are separate window sizes $win_{labeled}$ and $win_{unlabeled}$ maintained for labeled and unlabeled data respectively? The probability $P(y|\vec{x})$, which describes the user interest, i.e. the drifting concept, and which is captured by the labeled data, may change at an other rate than the probability $P(\vec{x})$, which describes the distribution of documents identically underlying both the labeled and unlabeled examples independent of their labels. Hence it is sensible to use separate windows to obtain the best information from both probability distributions.

Window on the Labeled Data

This subsections describes the window adjustment approach of [Klinkenberg and Joachims, 2000], which works on the labeled examples in the training set only and uses an inductive SVM. This window adjustment algorithm ignores any unlabeled example in the window and uses the labeled examples only. For the sake of a simpler description, lets assume for this subsection that all examples are labeled.

A window adjustment algorithm has to solve the following trade-off. A large window provides the learner with much training data, allowing it to generalize well given that the concept did not change. On the other hand, a large window can contain old data that is no longer relevant (or even confusing) for the current target concept. Finding the right size means trading-off the quality against the number of training examples.

To answer this question the window adjustment algorithm proposed in the following uses $\xi\alpha$-estimates in a particular way. At batch t, it essentially tries various window sizes, training a SVM for each resulting training set.

$$\vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)} \quad (13)$$

$$\vec{z}_{(t-1,1)}, \ldots, \vec{z}_{(t-1,m)}, \vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)} \quad (14)$$

$$\vec{z}_{(t-2,1)}, \ldots, \vec{z}_{(t-2,m)}, \vec{z}_{(t-1,1)}, \ldots, \vec{z}_{(t-1,m)}, \vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)} \quad (15)$$

$$\vdots$$

For each window size it computes a $\xi\alpha$-estimate based on the result of training. In contrast to the previous section, the $\xi\alpha$-estimator used here considers only the last batch, that is the m most recent training examples $\vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)}$

$$Err^m_{\xi\alpha}(h_\mathcal{L}) = \frac{|\{i : 1 \le i \le m \wedge (\alpha_{(t,i)} R^2_\Delta + \xi_{(t,i)}) \ge 1\}|}{m} \quad (16)$$

This reflects the assumption that the most recent examples are most similar to the new examples in batch $t+1$. The window size minimizing the $\xi\alpha$-estimate of the error rate is selected by the algorithm and returned as $win_{labeled}$.

The algorithm can be summarized as follows:

```
• input: $S_{train}$ training sample consisting of $t$ batches containing $m$
  (labeled) examples each
• for $h \in \{0, ..., t-1\}$
    – train SVM on examples $\vec{z}_{(t-h,1)}, \ldots, \vec{z}_{(t,m)}$
    – compute $\xi\alpha$-estimate on examples $\vec{z}_{(t,1)}, \ldots, \vec{z}_{(t,m)}$
• output: window size which minimizes $\xi\alpha$-estimate ($win_{labeled}$)
```

Window on the Unlabeled Data

According to the transductive setting, the test set, i.e. the examples in the batch $t+1$ are used as unlabeled examples in the optimization for learning a TSVM. For text classification tasks with stable, i.e. non-drifting concepts, the performance improvement of a TSVM as compared to an inductive SVM is maximal for small sets of labeled training examples and large sets of unlabeled examples used in addition [Joachims, 1999b]. Therefore, here not only the relatively small test set is considered useful unlabeled data, but also the unlabeled data in the current time window of size $win_{labeled}$, i.e. in the currently used part of the training set, and all training examples outside this time window, which are all treated as unlabeled examples, are considered potentially useful.

The batch size is usually imposed by the application, e.g. by the number of news texts arriving per day, and hence it is the same for the this part of the algorithm and the part described in the previous section. While only the labeled examples were used by the algorithm described in the previous section, both labeled and unlabeled examples are used by the algorithm described in this section. Since the earlier uses only the labeled examples, its virtual batch size m seems to be smaller than the real batch size m' of the latter containing all examples.

The algorithm to find the window for the unlabeled data (and the final hypothesis) can be summarized as follows:

```
• input: $S_{train}$ training sample consisting of $t$ batches containing $m'$
  examples each and $S_{test}$ test sample
• for $h \in \{0, ..., t-1\}$
    – train TSVM on examples $\vec{z}_{(t-h,1)}, \ldots, \vec{z}_{(t,m')}$,
      considering all training examples outside the window of size
      $win_{labeled}$ as unlabeled, and on the test examples
      $\vec{z}_{(t+1,1)}, \ldots, \vec{z}_{(t+1,m')}$
    – compute $\xi\alpha$-estimate on examples $\vec{z}_{(t+1,1)}, \ldots, \vec{z}_{(t+1,m')}$
• output: window size which minimizes $\xi\alpha$-estimate ($win_{unlabeled}$)
```

5 Experiments

5.1 Experimental Setup

In order to evaluate the first part of the window adjustment approach proposed in section 4, which corrsponds to the methods proposed in [Klinkenberg and Joachims, 2000], and which does not make any use of unlabeled data, each of the following *data management approaches* is evaluated in combination with the inductive SVM:

- *"Full Memory"*: The learner generates its classification model from all previously seen examples, i.e. it cannot "forget" old examples.
- *"No Memory"*: The learner always induces its hypothesis only from the most recent batch. This corresponds to using a window of the fixed size of one batch.
- Window of *"Fixed Size"*: A window of the fixed size of three batches is used.
- Window of *"Adaptive Size"*: The window adjustment algorithm proposed in [Klinkenberg and Joachims, 2000] adapts the window size to the current concept drift situation.

Table 1: Relevance of the categories in the concept change scenarios A, B, and C.

Scenario	Category	Probability of being relevant for a document of the specified category at the specified time step (batch)																			
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
A	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
B	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.6	0.4	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.4	0.6	0.8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
C	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.0	0.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
	3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

The experiments are performed in an information filtering domain, a typical application area for learning drifting concept. Text documents are represented as attribute-value vectors (*bag of words* model), where each distinct word corresponds to a feature whose value is the "ltc"-TF/IDF-weight [Salton and Buckley, 1988] of that word in that document. Words occurring less than three times in the training data or occurring in a given list of stop words are not considered. Each document feature vector is normalized to unit length to abstract from different document lengths.

The performance of a classifier is measured by the three metrics prediction error, recall, and precision. *Recall* is the probability, that the classifier recognizes a relevant document as relevant. *Precision* is the probability, that a document classified as relevant actually is relevant. All reported results are estimates averaged over ten runs.

The experiments use a subset of 2608 documents of the data set of the *Text REtrieval Conference (TREC)* consisting of English business news texts. Each text is assigned to one or several categories. The categories considered here are 1 (Antitrust Cases Pending), 3 (Joint Ventures), 4 (Debt Rescheduling), 5 (Dumping Charges), and 6 (Third World Debt Relief). For the experiments, three concept change scenarios are simulated. The texts are randomly split into 20 batches of equal size containing 130 documents each.[4] The texts of each category are distributed as equally as possible over the 20 batches.

Table 1 describes the relevance of the categories in the three concept change scenarios A, B, and C. For each time step (batch), the probability of being relevant (interesting to the user) is specified for documents of categories 1 and 3, respectively. Documents of the classes 4, 5, and 6 are never relevant in any of these scenarios. In the first scenario (*scenario A*), first documents of category 1 are considered relevant for the user interest and all other documents irrelevant. This changes abruptly (concept shift) in batch 10, where documents of category 3 are relevant and all others irrelevant. In the second scenario (*scenario B*), again first documents of category 1 are considered relevant for the user interest and all other documents irrelevant. This changes slowly (concept drift) from batch 8 to batch 12, where documents of category 3 are relevant and all others irrelevant. The third scenario (*scenario C*) simulates an abrupt concept shift in the user interest from category 1 to category 3 in batch 9 and back to category 1 in batch 11.

In order to evaluate the second part of the window adjustment approach proposed in section 4, the advantage achieved by using unlabeled examples and by automatically chosing the amount of unlabeled data to use, the following experiments are planned based on the same three scenarios with different rates of user feedback, but not yet completed:[5]

- Inductive SVM using only labeled data and the first part of the window adjustment method proposed in section 4.2.
- TSVM on the labeled training set used by the inductive SVM above plus the test set.
- TSVM as above, but also using the unlabeled examples in the window of size $win_{labeled}$ of the inductive SVM.
- TSVM as above, but also using all training data outside the window of size $win_{labeled}$ of the inductive SVM as unlabeled data.
- TSVM using both parts of the window adjustment method proposed in section 4.2, automatically chosing the amount of unlabeled data to use.

5.2 Results

Figure 1 compares the prediction error rates of the adaptive window size algorithm with the non-adaptive methods. The graphs show the prediction error on the following batch. In all three scenarios, the full memory strategy and the adaptive window size algorithm essentially coincide as long as there is no concept drift. During this stable phase, both show lower prediction error than the fixed size and the no memory approach. At the point of concept drift, the performance of all methods deteriorates. While the performance of no memory and adaptive size recovers quickly after the concept drift, the error rate full memory approach remains high especially in scenarios A and B. Like before the concept drift, the no memory and the fixed size strategies exhibit higher error rates than the adaptive window algorithm in the stable phase after the concept drift. This shows that the no memory, the fixed size, and the full memory approaches all perform suboptimally in some situation. Only the adaptive window size algorithm can achieve a relatively low error rate over all phases in all scenarios. This is also reflected in the average error rates over

[4] Hence, in each trial, out of the 2608 documents, eight randomly selected texts are not considered.

[5] The results of these experiments will be presented at the workshop.

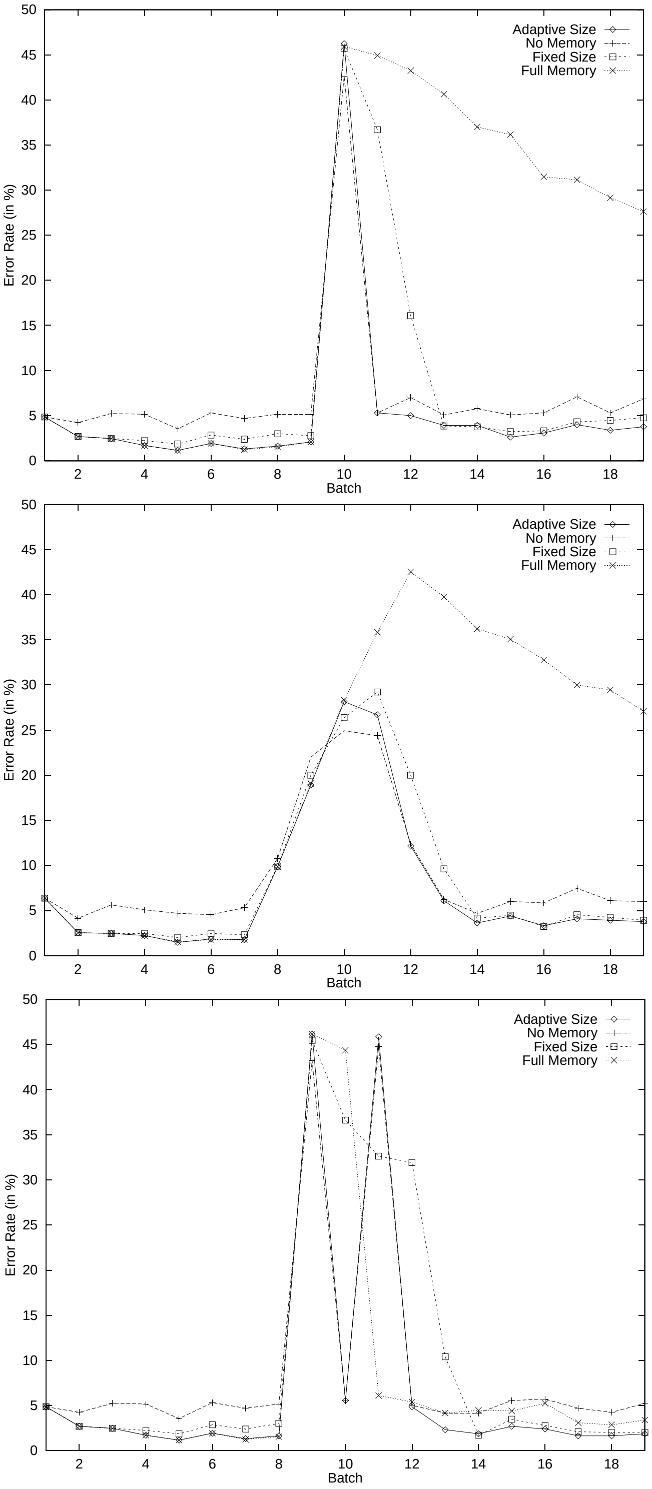

Figure 1: Comparison of the prediction error rates for scenario A (top), B (middle), and C (bottom). The x-axis denotes the batch number and the y-axis the average prediction error.

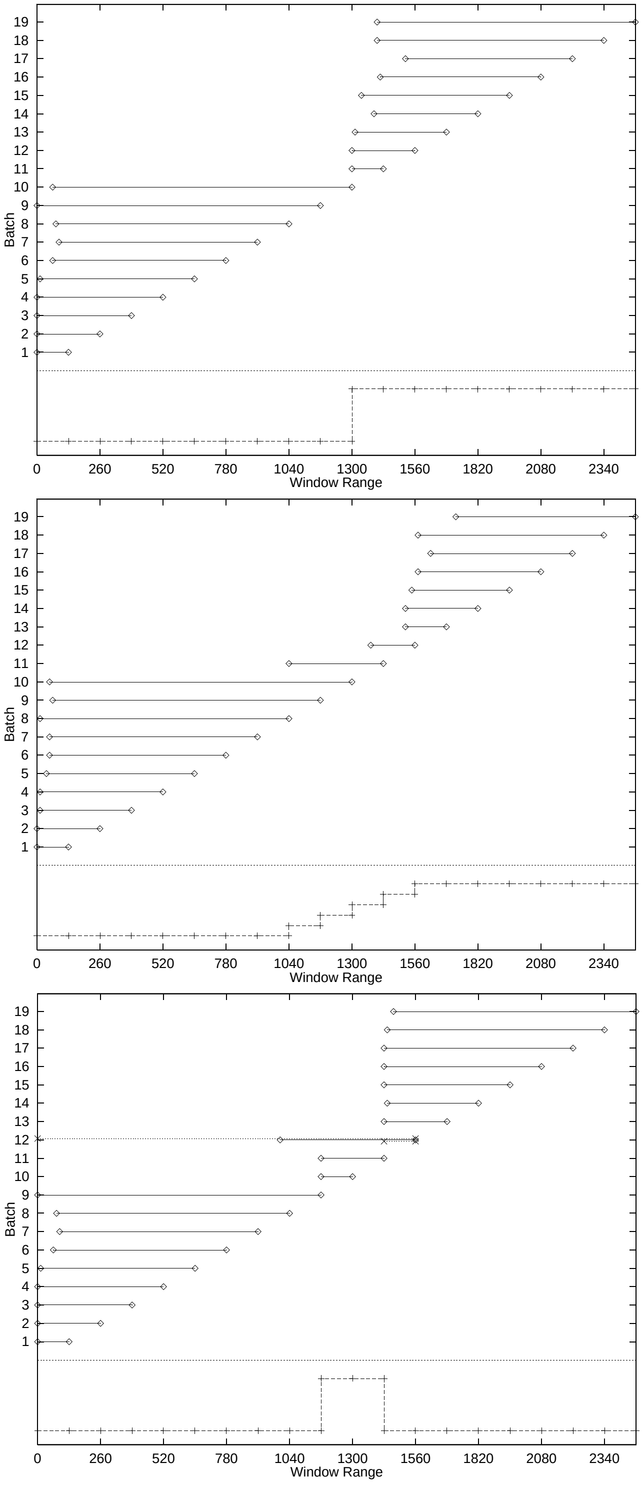

Figure 2: Window size and range for scenario A (top), B (middle), and C (bottom). The y-axis denotes the batch number. Each horizontal line marks the average training window range selected at that batch number. The bottom part of each graph depicts the location and type of the concept shift.

all batches given in Table 2. The adaptive window size algorithm achieves a low average error rate on all three scenarios. Similarly, precision and recall are consistently high.

The behavior of the adaptive window algorithm is best explained by looking at the window sizes it selects. Figure 2 shows the average training window ranges. The bottom of each graph depicts the time and extent of concept drift in the corresponding scenario. For scenario A the training window increases up to the abrupt concept change after batch 10, covering almost all examples available for the current concept. Only in batches 5 to 10 the average training set size is slightly smaller than maximally possible. Our explanation is that for large training sets a relatively small number of additional examples does not always make a “noticeable” difference. After the concept change in batch 10 the adaptive window size algorithm now picks training windows covering only those examples from after the drift as desired. A similar behavior is found for scenario B (Figure 2, middle). Since the drift is less abrupt, the adaptive window size algorithm intermediately selects training examples from both concepts in batch 11. After sufficiently many training examples from the new distribution are available, those earlier examples are discarded. The behavior of the adaptive window size algorithm in scenario C is reasonable as well (Figure 2, bottom). A particular situation occurs in batch 12. Here the window size exhibits a large variance. For 8 of the 10 runs the algorithm selects a small training set size of one batch, while for the remaining 2 runs it selects all available training examples starting with batch 1. Here there appears to be a borderline decision between accepting 2 (out of 12) batches of “bad” examples or just training on a single batch.

6 Summary and Conclusions

In this paper, we proposed a method for handling concept drift with support vector machines The method uses a window adjustment approach directly implementing the goal of discarding irrelevant data with the aim of minimizing generalization error. Exploiting the special properties of SVMs, we adapted $\xi\alpha$-estimates to the window size selection problem. Unlike for the conventional heuristic approaches, this gives the new method a clear and simple theoretical motivation. Furthermore, the new method is easier to use in practical applications, since it involves less parameters than complicated heuristics. Experiments in an information filtering domain show that the new algorithm achieves a low error rate and selects appropriate window sizes over very different concept drift scenarios.

Furthermore an extension of the method was proposed exploiting unlabeled data to reduce the amount of labeled data needed to learn drifting concepts. This extension based on transductive SVMs (TSVMs) can also react on changes in $P(\vec{x})$ and can be expected to need less labeled examples than previous approaches.

Table 2: Error, accuracy, recall, and precision of all window management approaches for all scenarios averaged over 10 trials with 20 batches each (standard sample error in parentheses).

	Full Memory	No Memory	Fixed Size	Adaptive Size
Scenario A:				
Error	20.36%	7.30%	7.96%	5.32%
	(4.21%)	(1.97%)	(2.80%)	(2.29%)
Recall	51.69%	74.42%	77.64%	85.35%
	(8.37%)	(4.61%)	(6.07%)	(4.93%)
Precision	64.67%	91.29%	87.73%	91.61%
	(8.38%)	(5.10%)	(5.93%)	(5.11%)
Scenario B:				
Error	20.25%	9.08%	8.44%	7.56%
	(3.56%)	(1.57%)	(2.00%)	(1.89%)
Recall	49.35%	67.22%	73.85%	76.70%
	(7.01%)	(5.04%)	(5.51%)	(5.42%)
Precision	65.09%	88.86%	87.19%	88.48%
	(6.80%)	(3.67%)	(4.18%)	(3.89%)
Scenario C:				
Error	7.74%	8.97%	10.17%	7.07%
	(3.05%)	(2.84%)	(3.30%)	(3.16%)
Recall	76.54%	63.68%	68.18%	78.17%
	(6.26%)	(5.27%)	(7.05%)	(6.34%)
Precision	83.15%	87.67%	79.00%	87.38%
	(6.69%)	(7.06%)	(8.09%)	(6.99%)

Acknowledgements

This work was supported by the DFG Collaborative Research Center on Computational Intelligence (SFB 531). We would also like to thank Thorsten Joachims for making his SVM light implementation publicly available [Joachims, 1999a].[3]

References

[Allan, 1996] James Allan. Incremental relevance feedback for information filtering. In H. P. Frei, editor, *Proceedings of the Nineteenth ACM Conference on Research and Development in Information Retrieval*, pages 270–278, New York, 1996. ACM Press.

[Balabanovic, 1997] Marko Balabanovic. An adaptive web page recommendation service. In W. L. Johnson, editor, *Proceedings of the First International Conference on Autonomous Agents*, pages 378–385, New York, 1997. ACM Press.

[Bennett, 1999] K. Bennett. Combining support vector and mathematical programming methods for classification. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT-Press, 1999.

[Blum and Mitchell, 1998] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Annual Conference on Computational Learning Theory (COLT-98)*, 1998.

[Burges, 1998] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[Fletcher, 1987] R. Fletcher. *Practical Methods of Optimization* (2nd edition). Wiley, New York, 1987.

[Gammerman *et al.*, 1998] A. Gammerman, V. Vapnik, and V. Vowk. Learning by transduction. In *Conference on Uncertainty in Artificial Intelligence*, pages 148–156, 1998.

[Helmbold and Long, 1994] David P. Helmbold and Philip M. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14:27–45, 1994.

[Joachims, 1999a] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, MA, USA, 1999.

[Joachims, 1999b] Thorsten Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning (ICML)*, Bled, Slowenien, 1999.

[Joachims, 2000] T. Joachims. Estimating the generalization performance of a SVM efficiently. In *Proceedings of the Seventeenth International Conference on Machine Learning*, San Francisco, 2000. Morgan Kaufman.

[Klinkenberg and Joachims, 2000] Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, San Francisco, 2000. Morgan Kaufmann.

[Klinkenberg and Renz, 1998] Ralf Klinkenberg and Ingrid Renz. Adaptive information filtering: Learning in the presence of concept drifts. In M. Sahami, M. Craven, T. Joachims, and A. McCallum, editors, *Workshop Notes of the ICML-98 Workshop on Learning for Text Categorization*, pages 33–40, Menlo Park, CA, USA, 1998. AAAI Press.

[Kuh *et al.*, 1991] A. Kuh, T. Petsche, and R.L. Rivest. Learning time-varying concepts. In *Advances in Neural Information Processing Systems*, volume 3, pages 183–189, San Mateo, CA, USA, 1991. Morgan Kaufmann.

[Kunisch, 1996] Gerhard Kunisch. *Anpassung und Evaluierung statistischer Lernverfahren zur Behandlung dynamischer Aspekte in Data Mining*. Masters thesis, Fachbereich Informatik, Universität Ulm, Germany, jun 1996.

[Lanquillon, 2000] Carsten Lanquillon. Partially Supervised text Classification: Combining Labeled and Unlabeled Documents Using an EM-like Scheme. In Ramon López de Mántaras and Enric Plaza, editors, *Proceedings of the 11th Conference on Machine Learning (ECML 2000)*, volume 1810 of *LNCS*, pages 229 – 237. Springer Verlag Berlin, Barcelona, Spain, 2000.

[Lunts and Brailovskiy, 1967] A. Lunts and V. Brailovskiy. Evaluation of attributes obtained in statistical decision rules. *Engineering Cybernetics*, 3:98–109, 1967.

[Mitchell *et al.*, 1994] Tom Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski. Experience with a learning personal assistant. *Communications of the ACM*, 37(7):81–91, jul 1994.

[Nigam *et al.*, 1998] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Learning to classify text from labeled and unlabeled documents. In *Proceedings of the AAAI-98*, 1998.

[Nigam *et al.*, 2000] Kamal Nigam, Andrew McCallum, Sebastian Thrun, and Tom Mitchell. Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning*, 39(2/3):103 – 134, 2000.

[Salton and Buckley, 1988] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[Taylor *et al.*, 1997] Charles Taylor, Gholamreza Nakhaeizadeh, and Carsten Lanquillon. Structural change and classification. In G. Nakhaeizadeh, I. Bruha, and C. Taylor, editors, *Workshop Notes of the ECML-97 Workshop on Dynamically Changing Domains: Theory Revision and Context Dependence Issues*, pages 67–78, apr 1997.

[van Rijsbergen, 1977] C. van Rijsbergen. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of Documentation*, 33(2):106–119, June 1977.

[Vapnik and Sterin, 1977] V. Vapnik and A. Sterin. On structural risk minimization or overall risk in a problem of pattern recognition. *Automation and Remote Control*, 10(3):1495–1503, 1977.

[Vapnik, 1998] V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.

[Widmer and Kubat, 1996] Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(2):69–101, 1996.