Intelligent Data Analysis 8 (2004) 281–300
IOS Press


# Learning drifting concepts: Example selection vs. example weighting

Ralf Klinkenberg
*University of Dortmund, Computer Science Department, Artificial Intelligence Unit (LS VIII), 44221 Dortmund, Germany*
*E-mail: Ralf.Klinkenberg@cs.uni-dortmund.de*
*URL: http://www-ai.cs.uni-dortmund.de/*

Received 1 February 2003
Revised 15 July 2003
Accepted 15 September 2003

**Abstract.** For many learning tasks where data is collected over an extended period of time, its underlying distribution is likely to change. 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. This paper proposes several methods to handle such concept drifts with support vector machines. The methods either maintain an adaptive time window on the training data [13], select representative training examples, or weight the training examples [15]. The key idea is to automatically adjust the window size, the example selection, and the example weighting, respectively, so that the estimated generalization error is minimized. The approaches are both theoretically well-founded as well as effective and efficient in practice. Since they do not require complicated parameterization, they are simpler to use and more robust than comparable heuristics.

Experiments with simulated concept drift scenarios based on real-world text data compare the new methods with other window management approaches. We show that they can effectively select an appropriate window size, example selection, and example weighting, respectively, in a robust way. We also explain how the proposed example selection and weighting approaches can be turned into incremental approaches. Since most evaluation methods for machine learning, like e.g. cross-validation, assume that the examples are independent and identically distributed, which is clearly unrealistic in the case of concept drift, alternative evaluation schemes are used to estimate and optimize the performance of each learning step within the concept drift handling frameworks as well as to evaluate and compare the different frameworks.

## 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 behavior and to predict future sales. As the customer behavior 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. With the amount of online information and communication growing rapidly, there is an increasing need for automatic information filtering. Information filtering techniques are used, for example, to build personalized news filters, which learn about the news-reading preferences

of a user [18,33], to filter e-mail [4], or to guide a user's search on the World Wide Web [10]. Both the interest of the user, i.e. the concept underlying the classification of the texts, and the document content change over time. A filtering system should be able to adapt to such concept changes.

This paper discusses several approaches to deal with the problem of concept drift. The central question is, how representative or important older examples are for predicting new instances of the possibly changed concept. In case of a drifting concept, the importance of an example obviously depends on its age. We extend the approach of [13] by using some special properties of support vector machines that will prove useful in handling concept drift, for example effective and efficient performance estimation [8] and example weighting [12,15]. A main goal is to keep the learning algorithm as effective, efficient, and with as little parameterization as possible.

Experiments with simulated concept drift scenarios based on real-world text data compare the new methods with other window management approaches. We show that they can effectively select an appropriate window size, example selection, and example weighting, respectively, in a robust way. We also explain how the proposed example selection and weighting approaches can be turned into incremental approaches. Since most evaluation methods for machine learning, like e.g. cross-validation, assume that the examples are independent and identically distributed, which is clearly unrealistic in the case of concept drift, alternative evaluation schemes are used to estimate and optimize the performance of each learning step within the concept drift handling frameworks (internal evaluation and parameter optimization) as well as to evaluate and compare the different frameworks (external evaluation and comparison).

## 2. Concept drift

### *2.1. Problem definition*

Throughout this paper, we study the problem of concept drift for the pattern recognition problem in the following framework. Each example $z = (x, y)$ consists of a feature vector $x \in \mathbf{R}^N$ 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.

$$z_{(1,1)}, \ldots, z_{(1,m)}, z_{(2,1)}, \ldots, z_{(2,m)}, \cdots, z_{(t,1)}, \ldots, z_{(t,m)}, z_{(t+1,1)}, \ldots, z_{(t+1,m)}$$

$z_{(i,j)}$ denotes the $j$-th example of batch $i$. For each batch $i$ the data is independently identically distributed with respect to a distribution $\Pr_i(x, y)$. Depending on the amount and type of concept drift, the example distribution $\Pr_i(x, y)$ and $\Pr_{i+1}(x, y)$ 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.

### *2.2. Heuristic approaches to handle concept drift*

In machine learning, changing concepts are often handled by time windows of fixed or adaptive size on the training data (e.g., see [14,19,22,34]) or by weighting data or parts of the hypothesis according to their age and/or utility for the classification task [17,31]. The latter approach of weighting examples has already been used for information filtering in the incremental relevance feedback approaches of [1]

and [2]. In this paper, the earlier approach maintaining a window of adaptive size is explored. More detailed descriptions of the methods described above and further approaches can be found in [11].

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.

### *2.3. Theoretical approaches to handle 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 [6] 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, Petsche, and Rivest [16] 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 [14,19,34]. 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

We use support vector machines (SVMs) as our learning algorithm, because SVMs have some very useful properties that can be exploited in our approach. The theory of SVMs itself is well known (see [32]). Therefore we will give only a short introduction to the concepts that are important with respect to our work.

Support vector machines are based on the structural risk minimization principle [32] of statistical learning theory. Statistical learning theory deals with the question, how a function $f$ from a class of functions $(f_\alpha)_{\alpha\in\Lambda}$ can be found, that minimizes the expected risk

$$R[f] = \int\int L(y, f(x))dP(y|x)dP(x) \tag{1}$$

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

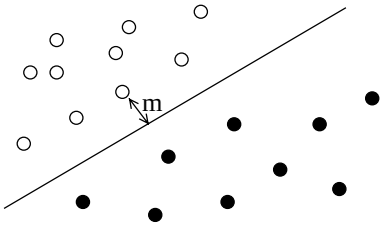


Fig. 1. Maximum margin hyperplane and margin m.

with respect to a loss function $L$, when the distributions of the examples $P(x)$ and their classifications $P(y|x)$ are unknown and have to be estimated from finitely many examples $(x_i, y_i)_{i \in I}$.

The SVM algorithm solves this problem by minimizing the regularized risk $R_{\text{reg}}[f]$, which is the weighted sum of the empirical risk $R_{\text{emp}}[f]$ with respect to the data $(x_i, y_i)_{i=1\ldots n}$ and a complexity term $||w||^2$

$$R_{\text{reg}}[f] = ||w||^2 + C \cdot R_{\text{emp}}[f]. \tag{2}$$

In their basic formulation, support vector machines find a linear decision function $y = f(x) = \text{sign}(w \cdot x + b)$ that both minimizes the prediction error on the training set and promises the best generalization performance. Geometrically, the principle of the SVM can be interpreted as finding a hyperplane in the example space, that separates the positive and negative examples (minimizes the error on the training set) with maximum margin (best expected generalization performance), see Fig. 1.

Given the examples $(x_1, y_1), \ldots, (x_n, y_n)$, the SVM solution is found by solving the following optimization problem:

$$\Psi(w, \xi) = \frac{1}{2}(w^T w) + C \sum_{i=1}^{n} \xi_i \tag{3}$$
$$\rightarrow \min$$

subject to

$$y_i(w^T x_i + b) \leqslant 1 - \xi_i, i = 1, \ldots, n \tag{4}$$

$$\xi_i \geqslant 0, i = 1, \ldots, n \tag{5}$$

The decision hyperplane is given by the normal vector $w$ and the additive constant $b$, such that $f(x) = w^T x + b$. The variables $\xi_i$ are slack variables that allow for a certain amount of misclassification in Eq. (4). In practice, this optimization problem can be efficiently solved in its dual form

$$\Phi(\alpha) = -\frac{1}{2} \sum_{i,j=1}^{n} y_i y_j x_i \cdot x_j + \sum_{i=1}^{n} \alpha_i \tag{6}$$
$$\rightarrow \min$$

subject to

$$0 \leqslant \alpha_i \leqslant C \ \forall i = 1 \ldots n \tag{7}$$

$$\sum_{i=1}^{n} \alpha_i = 0 \tag{8}$$

Here, the hyperplane is given by $w = \sum_{i=1}^{n} \alpha_i y_i x_i$. The vectors $x_i$ that have non-zero variables $\alpha_i$ are called the support vectors. The so-called capacity constant $C$ limits the maximum impact an individual example may have on the hypothesis to be learned and thereby allows to trade off model complexity versus generalization.

### *3.1. SVM loss functions and weighted examples*

In Eq. (3), the empirical risk of the SVM solution is measured with respect to a linear loss function, but the support vector algorithm is not restricted to the case of linear loss functions, but can be extended to broader classes of loss functions (e.g. see [28]). In particular, one can set an individual weight $c_i$ to each example by replacing definition (3) by

$$\Psi(w, \xi) = \frac{1}{2}(w^T w) + C \sum_{i=1}^{n} c_i \xi_i. \tag{9}$$

This leads to the same dual formulation as before, except that Eq. (7) is replaced by

$$0 \leqslant \alpha_i \leqslant c_i C \quad \forall i = 1 \ldots n \tag{10}$$

The resulting decision function is biased towards examples with a higher weight. The effect of this manipulation can be viewed as changing the probability distribution $P(x)$ of the examples, placing more probability mass to the examples with higher weight.

### *3.2. Incremental SVMs*

Support vector machines have been successfully used for learning with large and high dimensional data sets. One reason for this is the fact that the generalization property of an SVM does not depend on all training examples, but only on a subset of them, the so-called support vectors [32]. Since the training examples that are no support vectors do not influence the learning result, an SVM learns the same model from the support vectors only as from all training examples.

Syed et al. [29,30] describe an approach to incrementally learning support vector machines by compressing the training data from past batches to their support vectors and discarding the examples that are no support vectors. At each point of time, i.e. for each new batch of data, a new SVM is trained on the support vectors from the previous learning step and on all new training examples. For stable, i.e. non-drifting concepts, SVMs trained by this approach compare well to a non-incrementally trained equivalents [30]. The approach handles *virtual* concept drift implied by incrementally learning from several subsamples of a large training set [29], but it does not address the problem of (*real*) concept drift addressed here (see Section 2). Another disadvantage of this approach to incrementally train an SVM on new data by discarding all previous data except their support vectors is that it gives only approximate results.

As stated above, the incremental learning result is equal to the non-incremental result, if the last training set contains all examples that are support vectors in the non-incremental case. But how to decide whether an example will end up as a support vector in the non-incremental case without actually training

on all data? The problem is that support vectors are a sufficient description of the decision boundary between examples ($P(y|x)$), but not of the examples themselves ($P(x)$). Since there are typically only very few support vectors, their influence on the decision function in the next incremental learning step may be very small, if the data is distributed differently. The robustness of SVMs against outliers, usually a desired property, may now lead to not taking the old support vectors sufficiently into account for the new decision function.

Rüping [25,26] extends the approach described in [29] to make up for this by making an error on an old support vector more costly than an error on a new example. This extended approach considers weights for the training examples as described in the previous section. It assigns a weight of one to each new example and computes the weight for the old support vectors as the number of examples in the previous batch divided by the number of support vectors of the previous learning step on this previous batch, in order to approximate the average error of the new decision function over all old examples by the average error over just the old support vectors. The results of this extended approach come closer to the corresponding non-incremental results and hence yield a better performance [25,26], but of course the results are still just an approximation of the non-incremental results.

In contrast to the approaches above, Cauwenbergh and Poggio [3] propose an exact online method to incrementally train an SVM that recursively constructs the solution, incorporating one example at at time and retaining the Kuhn-Tucker (KT) condition on *all* previously seen data. But like the approaches above, this method does not take the problem of concept drift into consideration. Therefore neither of these incremental approaches was used for the work described in this paper. However, Section 6 describes how they could be used in the concept drift frameworks proposed in this paper.

### *3.3. Estimating the performance of SVMs by $\xi\alpha$-Estimators*

The performance of an SVM can be effectively and efficiently estimated by a special form of $\xi\alpha$-estimates [8]. $\xi\alpha$-estimators are based on the idea of leave-one-out estimation [20]. The leave-one-out estimator of the error rate proceeds as follows. From the training sample $S = ((x_1, y_1), \cdots, (x_n, y_n))$ the first example $(x_1, y_1)$ is removed. The resulting sample $S^{\setminus 1} = ((x_2, y_2), \cdots, (x_n, y_n))$ is used for training, leading to a classification rule $h_{\mathcal{L}}^{\setminus 1}$. This classification rule is tested on the held out example $(x_1, y_1)$. 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 $R_\Delta^2$ being the maximum difference of any two elements of the Hessian (i.e. $R_\Delta^2 \geqslant \max_{x,x'}(\mathcal{K}(x,x) - \mathcal{K}(x,x'))$),

$$d = |\{i : (\alpha_i R_\Delta^2 + \xi_i) \geqslant 1\}| \tag{11}$$

counts the number of training examples, for which the quantity $\alpha_i R_\Delta^2 + \xi_i$ exceeds one. Since the document vectors are normalized to unit length in the experiments described in this paper, here $R_\Delta^2 = 1$.

It is proven in [8] 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 $\xi\alpha$-estimator of the error rate is

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

The theoretical properties of this $\xi\alpha$-estimator are discussed in [8]. 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 hold-out estimate using twice as much data. It is also possible to design similar estimators for precision and recall, as well as for combined measures like $F1$ [8].

### *3.4. SVMs for text classification*

It was first noted by Joachims in [7], that SVMs are especially well suited for text classification, because the complexity of a SVM hyperplane depends on the margin it separates the data with and not on the dimensionality of the input space. Text data typically has a very high dimension (more than 10000) and very few irrelevant features. SVMs can efficiently learn with all features in the data set, so they are much better suited than other algorithm that demand complicated feature selection. Experience shows that support vector machines are currently the most successful tool for text classification [9].

## 4. Learning drifting concepts by example selection or example weighting

In a learning problem with drifting concepts as introduced in Section 2, we face the problem to decide, how much information from past examples may be used to find a hypothesis that is adequate to predict the class information of future data. Since we do not know, if and when a concept drift happens, there are two opposing effects: On the one hand, the older the data is, the more likely it is that its probability distribution differs from the current distribution that underlies the process, so that the data may be misleading. On the other hand, the more data is used in the learning process, the better the results are if no concept drift occurred since the data arrived.

In this section we present different approaches for learning drifting concepts. They differ in the way previous examples are used to construct a new hypothesis. All of our approaches share the assumption, that concept drifts do not reverse, i.e. newer examples are always more important than older ones. This assumption was implemented by a common scheme for estimating the performance of a learner: In all experiments, the performance was only calculated on the last batch of data, regardless of how many batches were used in training. To get a good estimation of the performance but still be efficient, we used the so-called $\xi\alpha$-estimator of [8] (see also Section 3.3), which estimates the leave-one-out-error of a SVM based solely on the one SVM solution learned with all examples.

### *4.1. Example selection using an adaptive time window*

One of the simplest scenarios for detecting concept drift are concept drifts that happen very quickly between relatively stable single concepts. For example, imagine a user of an information filtering system, who wants to buy a new car: at first, he is interested in information about all sorts of cars, but after he made his decision and bought the car, he is only interested in information about this special type of car. This may be more accurately called “concept change” or “concept shift” rather than “concept drift”.

In this scenario, the problem of learning drifting concepts can be approached as the problem of finding the time point $t$ at which the last concept change happened. After that, a standard learning algorithm for fixed concepts can be used to learn from the data since $t$. Similarly, other concept drift scenarios can be handled by using a time window on the training data, assuming that the amount of drift increases with time and hence focusing on the last $n$ training examples.

The shortcomings of previous windowing approaches are that they either fix the window size [22] or involve complicated heuristics [14,19,34]. 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.

In [13], Klinkenberg and Joachims presented an approach to selecting an appropriate window size that does not involve complicated parameterization. The 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, a special form of $\xi\alpha$-estimates [8] (see also Section 3.3) is used.

The adaptive window approach employs these estimates in the following way. At batch $t$, it essentially tries various window sizes, training a SVM for each resulting training set.

$$z_{(t,1)}, \ldots, z_{(t,m)} \tag{13}$$

$$z_{(t-1,1)}, \ldots, z_{(t-1,m)}, z_{(t,1)}, \ldots, z_{(t,m)} \tag{14}$$

$$z_{(t-2,1)}, \ldots, z_{(t-2,m)}, z_{(t-1,1)}, \ldots, z_{(t-1,m)}, z_{(t,1)}, \ldots, z_{(t,m)} \tag{15}$$

$$\vdots$$

For each window size it computes a $\xi\alpha$-estimate based on the result of training, considering only the last batch for the estimation, that is the $m$ most recent training examples $z_{(t,1)}, \ldots, z_{(t,m)}$

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

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 used to train a classifier for the current batch.

The window adaptation algorithm can be summarized as follows:

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

### 4.2. Example weighting

In information filtering systems, the user may change his interests in a specific topic slowly. In this case, one cannot find a specific time point, at which old examples become irrelevant, but the amount of information one can draw from a certain example will slowly decrease over a longer amount of time. Therefore, the sharp distinction between examples that are kept and examples that are left out in the learning process does not sufficiently represent the process behind the data.

The decreasing importance of older examples can be modeled by assigning a weight $c_i$ to each example $z_i$ and by learning a decision function with respect to these weights, for example by the method shown in Section 3.1 [15]. But how to choose these weights? Of course, a weighting scheme must take into account the variability of the target concept and be adaptive with respect to the actual performance of the learner.

In our approach, the criterion to select the optimal weights is again the estimated performance of the learner on the newest batch of data. This guarantees, that the temporal order of the examples is respected by the weighting schemes (an example from a newly emerging concept looks like an outlier with respect to the old data, but of course is no outlier but highly informative).

In the (global) weighting scheme, we select the weights of the examples solely based on their respective age, for example using an exponential aging function $w_\lambda(x) = \exp(-\lambda t_x)$, where example $x$ was found $t_x$ time steps ago. The larger $\lambda$ is, the sooner an examples becomes irrelevant. In the extreme cases, for $\lambda \to \infty$ we learn only on the newest examples and for $\lambda = 0$ all examples share an equal weight.

To be adaptive, we start several learning runs for each new batch with different values of $\lambda$ and pick the best $\lambda$ at each batch by estimating the performance of the learning result on the last batch of data. In a way, this algorithm is a continuous version of the algorithm of [13] that was presented in the last section: instead of a hard cut to remove uninformative examples, the contribution of these examples to the final learning result is slowly reduced.

### 4.3. Local models: Local example weighting or batch selection

A special characteristic of text classification is the high dimensionality of the examples compared to the number of examples, which makes the examples stand almost orthogonal to each other. This is the reason why in text classification tasks the classes often are linearly separable.

In the concept drift setting, we have a set of topics and assume that the user is interested in a specific subset of topics. In experiments with this setting, one can observe that a subset of multiple topics can be separated from the rest of the data just as good as a single topic. Accordingly, one can also observe, that an SVM classifier that is trained in a concept drift setting, even if it is trained on a small subset of the data, has a low error on a test set from the same example distribution (in our experiments, below 10%). Vice versa, if a concept drift occurs, i.e. if the users interest in a topic changes, almost all of the examples from this topic will be classified falsely and the error rate will grow considerably (with the increase of the error rate depending on the size of this topic in relation to the other topics). A good example of this behavior can be seen in Fig. 2. In this setting, we can see a considerable increase in the classification error after a concept drift occurs at batch 10. In this example, the classifier was re-trained for each new batch on all examples prior to the new batch.

This observation lead us to the local model approach: In the first step, a classifier is learned on only the most recent batch of data. Of course, in most cases this classifier will not be as good as it can be, but we can be sure that it always will be the classifier that is most up-to-date with the drifting concept. Now we can use this classifier to estimate, which batches of data were generated from the same

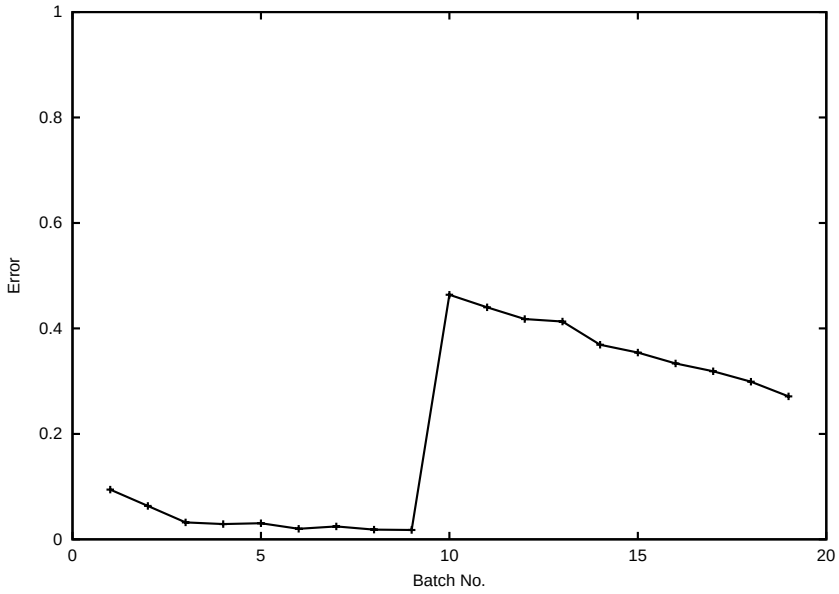


Fig. 2. Typical classification error in a concept drift setting.

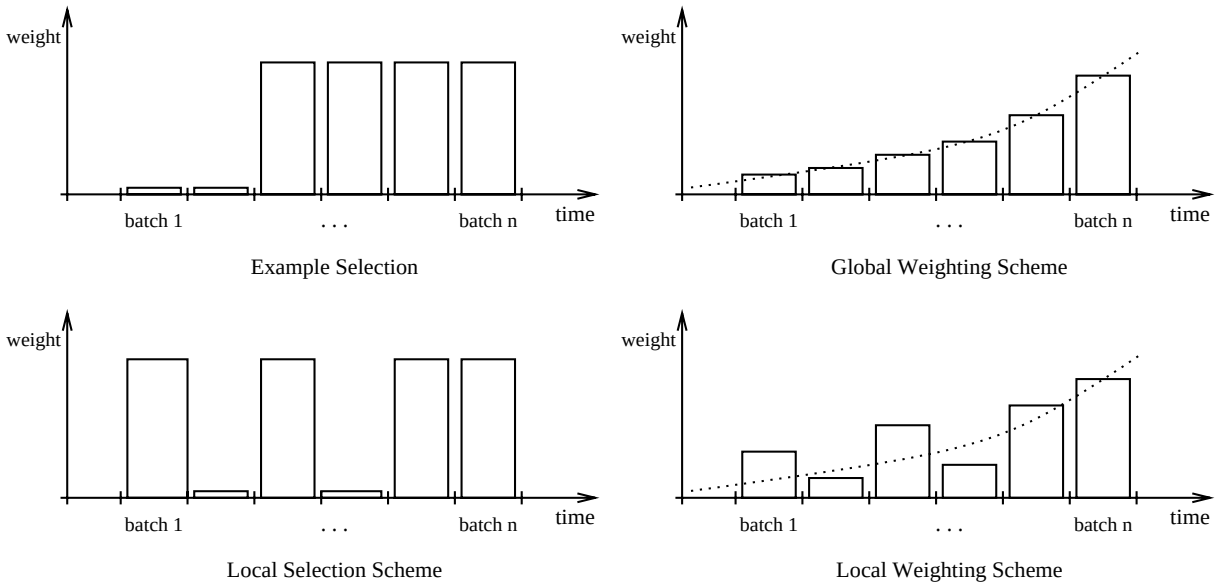


Fig. 3. Comparison of the example weights in the four weighting schemes.

model (i.e. the same users interest) as the most recent batch, by comparing the estimated leave-one-out error of the classifier on the most recent batch to its test error on the other batches. The higher the error, the more unlikely is the data given the model. Note that at this point, it is important to use the leave-one-out-estimation and not the training error to avoid errors by over-fitting the most recent data.

In a second step, the information about the error of the classifier can be used to build a training set for the actual classifier. There are two ways to use this information: We can either exclude all batches from

the new training set that have a significantly higher classification error than the most recent batch (*batch selection scheme*). By this, we hope to train the final classifier on all data generated by the current model in a way similar to the *example selection scheme* in Section 4.1 (*adaptive time window*). Or we can use the error information to adjust the weights for the examples on each batch in a way similar to Section 4.2. The higher the error of the batch, the lower the weights will be. We call this approach the *local weighting scheme*, because here the weights are adjusted locally on each batch in contrast to the *global weighting scheme* in Section 4.2. Figure 3 illustrates these different example selection and weighting schemes. Of course, we can also combine both weighting schemes by multiplying the two weights for each example (*combined weighting scheme*). In the spirit of Bayesian statistics, this combines an a-priori assumption about the importance of the examples (the global weight) with an a-posteriori update (performance measure in the local weight).

## 5. Experiments

### *5.1. Experimental setup and evaluation scheme*

In order to evaluate the learning approaches for drifting concepts proposed in this paper, three simple non-adaptive data management approaches are compared to the adaptive time window approach and to the example weighting and selection strategies, all using SVMs as their core learning algorithm:

- *"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 time window of the fixed size of three batches is used on the training data.[2]
- Window of *"Adaptive Size"*: The window adjustment algorithm described in Section 4.1 (and in [13]) adapts the window size to the current concept drift situation.
- *"Global Weights"*: The examples of old batches are weighted by an exponential aging function according to their age, so that older examples receive lower weights (see Section 4.2).
- *"Local Weights"*: The examples of old batches are weighted according to their fit to a model learned on the most recent batch only, i.e. the weight of an example $x$ of an old batch $t_x$ is inversely proportional to the error rate of that batch on this model: $w_{\text{local}}(x) := 1 - 5 \cdot (\text{error}_{t_x} - 0.1)$, where the weight is set to one for error rates below 10% and to zero for error rates above 30% (see Section 4.3).
- *"Batch Selection"*: The batches producing an error less than twice the estimated error of the newest batch, when applied to a model learned on the newest batch only, receive a weight of one. The weight of all other examples is set to zero.

The experiments are performed in an information filtering domain, a typical application area for learning drifting concepts. 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 [27]

[2]The fixed window size of three batches outperformed smaller and larger fixed window sizes in the following experiments and hence was chosen here.

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 four runs.

While most evaluation methods for machine learning, like e.g. cross-validation, assume that examples are independent and identically distributed, this assumption is clearly unrealistic in the presence of concept drift. Therefore the concept drift approaches proposed in this paper use $\xi\alpha$-estimates (see Section 3.3 and [8]) instead of cross-validation to estimate and optimize the performance of a particular parameterization at each learning step and only estimate the performance on the currently last batch (see Section 4). This is not only more appropriate for the concept drift situation at hand, but also more efficient, because the $\xi\alpha$-estimator can be computed within a single training run.

Such an evaluation problem occurs not only within each time step of a concept drift scenario handled by one of the concept drift frameworks (internal evaluation and parameter optimization), but it also occurs when several such frameworks are to be compared like here (external evaluation and overall performance comparison). Just like in the earlier case, evaluation methods like cross-validation are inappropriate for the latter case. In this paper, we use repeated runs with simulated concept drift scenarios to obtain averaged and thereby statistically more reliable results for the overall comparison.

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.[3] The texts of each category are distributed as equally as possible over the 20 batches.

In the three scenarios, a document is considered relevant at a certain point in time, if it matches the interest of the simulated user at that time. For each TREC topic and each batch in each scenario the probability that a document from this topic is relevant for the user interest at this time (batch) is specified. In the scenarios simulated here, the user interest changes between the topics 1 and 3. Documents of the classes 4, 5, and 6 are never relevant in any of these scenarios. Figure 4 shows the probability of being relevant for a document of category 1 at each batch for each of the three scenarios. Documents of category 3 are specified to always have the inverse relevance probability of documents of category 1, i.e. *1.0 – relevance of category 1*. 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.

The experiment were conducted with the machine learning environment YALE [5,21,23].[4] For all time window, example weighting, and example selection approaches, support vector machines were

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

[4] YALE: available at *http://yale.cs.uni-dortmund.de/*.

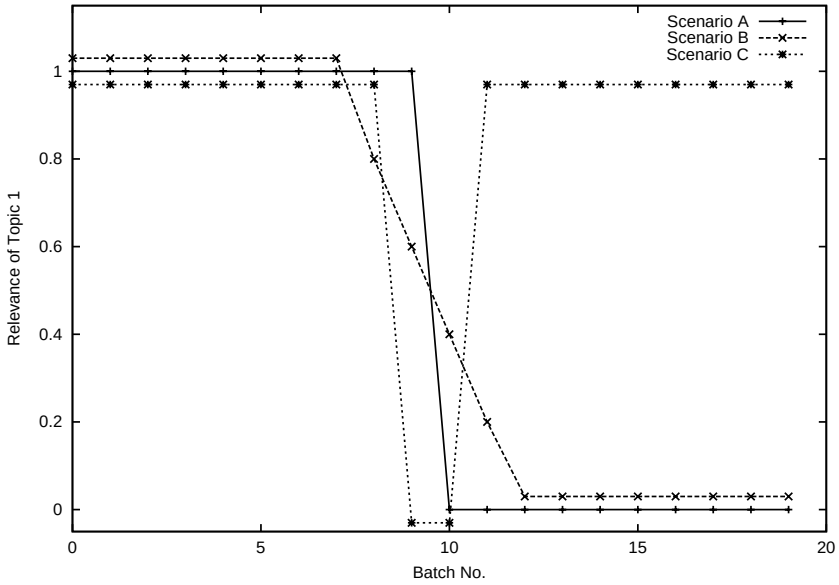


Fig. 4. Relevance of the TREC topic 1 in the concept change scenarios A, B, and C. The relevance of TREC topic 3 is *1.0 – relevance of topic 1*. The relevance of all other topics is always zero.

used as core learning algorithm. Here we chose the SVM implementation *mySVM* [24].[5] Since linear kernels are the standard kernel type used for text classification problems, and since more complex kernel types usually do not perform better on text classification tasks, only linear and no other kernel types were tried here. After preliminary experiments to determine a good value for the capacity constant $C$ of the SVM, which allows to trade off model complexity versus generalization, testing the values $C \in \{1, 10, 100, 1000\}$, the value $C = 1000$ was chosen for the experiments described here.

For the global weighting scheme, an exponential aging function of the form $w_\lambda(x) = \exp(-\lambda t_x)$, where example $x$ was found $t_x$ time steps ago, was employed unless mentioned otherwise. At each batch, the value of the parameter $\lambda$ is automatically chosen among $\lambda \in \{0.01, 0.1, 0.2, 0.4, 0.6, 1.0, 2.0, 4.0\}$, such that the chosen value optimizes the expected performance of the learner.

### *5.2. Experimental results*

Table 1 shows the results of all time window, example weighting, and batch selection approaches on all scenarios in terms of the metrics prediction error, recall, and precision. While the example weighting methods outperform the trivial non-adaptive approaches, the adaptive time window approach and the batch selection strategy clearly outperform both, the non-adaptive approaches and the example weighting methods.

Figures 5 to 8 compare the prediction error rates of all strategies on scenario A over time, i.e. the graphs depict the prediction error on the following batch for each point in time. Figure 5 shows the error rates of the trivial non-adaptive methods full memory, no memory, and fixed size. Figure 6 presents the error rates of the example weighting methods global and local weighting. Figure 7 compares the example selection strategies using an adaptive time window and batch selection, respectively, and Figure 8 finally compares the best approach of each of the previous groups of strategies.

[5] *mySVM*: available at *http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/*.

Table 1
Error, recall, and precision of all time window, example weighting, and example selection methods for all scenarios averaged over 4 trials with 20 batches

| | Full Memory | No Memory | Fixed Size | Global Weights | Local Weights | Adaptive Size | Batch Selection |
|---|---|---|---|---|---|---|---|
| Scen. A: | | | | | | | |
| Error | 21.11% | 11.16% | 9.03% | 8.45% | 8.98% | 6.65% | 6.15% |
| Recall | 46.70% | 52.59% | 70.39% | 73.75% | 73.81% | 77.90% | 80.24% |
| Precision | 64.66% | 91.71% | 87.64% | 87.64% | 85.92% | 91.57% | 91.73% |
| Scen. B: | | | | | | | |
| Error | 21.30% | 12.64% | 9.76% | 10.62% | 12.29% | 9.06% | 9.33% |
| Recall | 43.89% | 46.81% | 65.91% | 65.72% | 64.16% | 68.72% | 69.85% |
| Precision | 64.05% | 89.74% | 87.14% | 84.82% | 81.36% | 88.06% | 87.90% |
| Scen. C: | | | | | | | |
| Error | 8.60% | 12.73% | 11.19% | 8.80% | 8.78% | 8.56% | 7.55% |
| Recall | 71.14% | 35.38% | 58.30% | 68.57% | 71.15% | 68.97% | 74.84% |
| Precision | 83.28% | 88.66% | 78.84% | 84.29% | 82.59% | 86.65% | 87.79% |

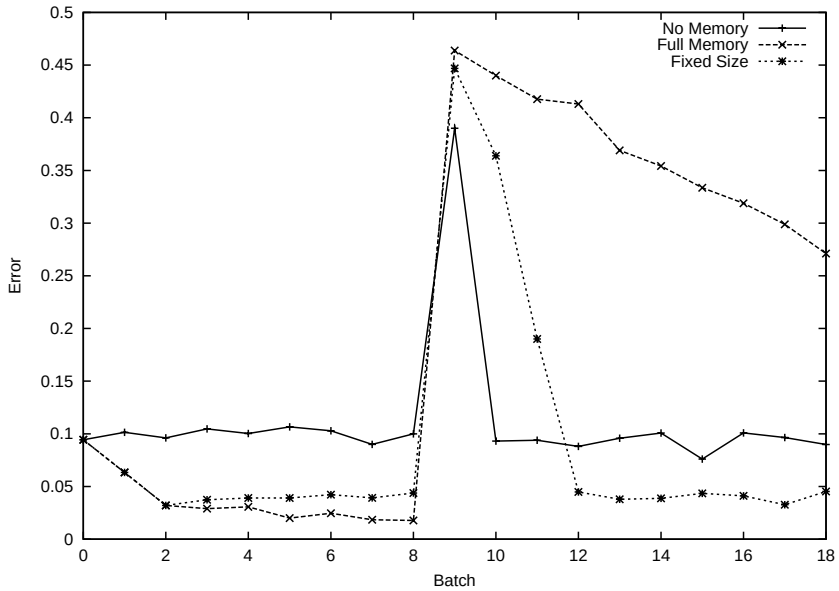


Fig. 5. Classification errors of the trivial approaches on each batch in scenario A (averaged over 4 runs).

Among the two example selection strategies, batch selection performs better than the adaptive time window approach, especially on scenario C as Table 1 and Fig. 9 show, where the initial concept reflecting the user interest, topic 1, is only shortly interrupted by a concept shift to topic 3, and then returns to topic 1 again. In the batches after the second concept shift in scenario C, the adaptive time window can only capture the data after the second concept shift, if it is to exclude the no longer representative data between the two concept shifts, while the batch selection strategy can also use the earlier data of the time before the first concept shift and selectively exclude only the no longer relevant batches between the two concept shifts. This allows the batch selection to maintain a larger consistent training set and thereby to better generalize resulting in a lower error rate.

From the results of the non-adaptive approaches (full memory, no memory, and fixed size), we can see that the error is the lowest if the time window contains all time points from the current model and

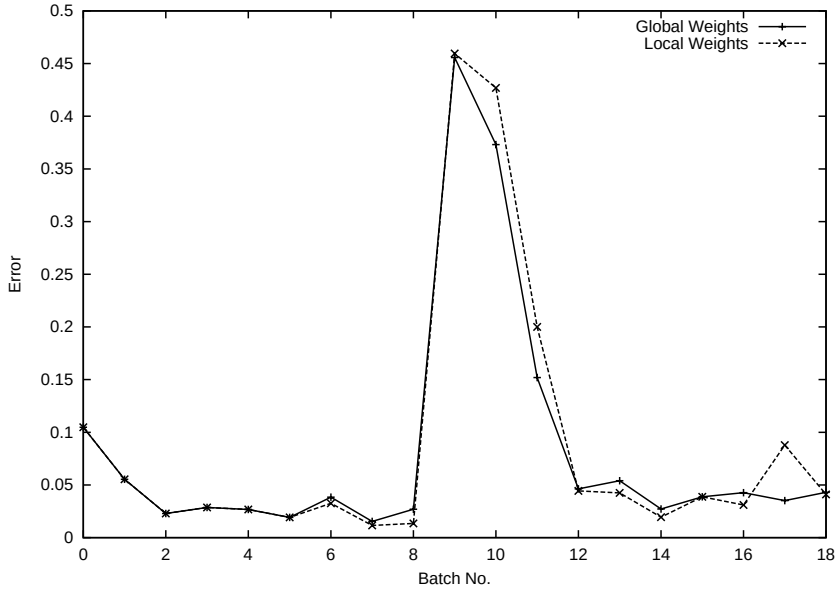


Fig. 6. Classification errors of the weighting approaches on each batch in scenario A (averaged over 4 runs).

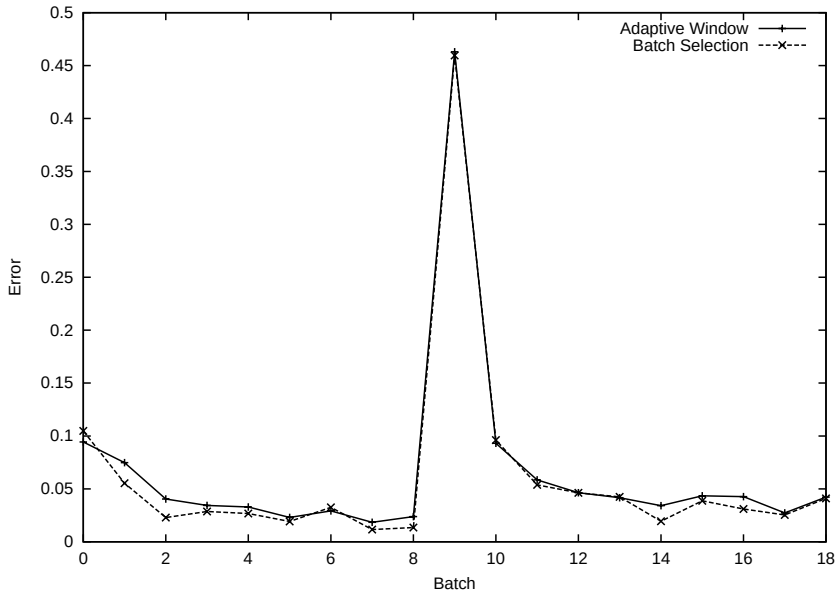


Fig. 7. Classification errors of the batch selection approaches on each batch in scenario A (averaged over 4 runs).

no others. For example in scenario A, as depicted in Fig. 5, as long as no concept drift occurs, the full memory approach has the lowest error. Immediately after the concept drift, the no memory approach quickly returns to its previous error level, while the fixed size memory approach takes longer until it finally reaches a lower error than the no memory approach again. For the full memory approach, even nine time points after the concept shift the error rate is still three times higher than the error of the other strategies. Of course, these findings are not very surprising.

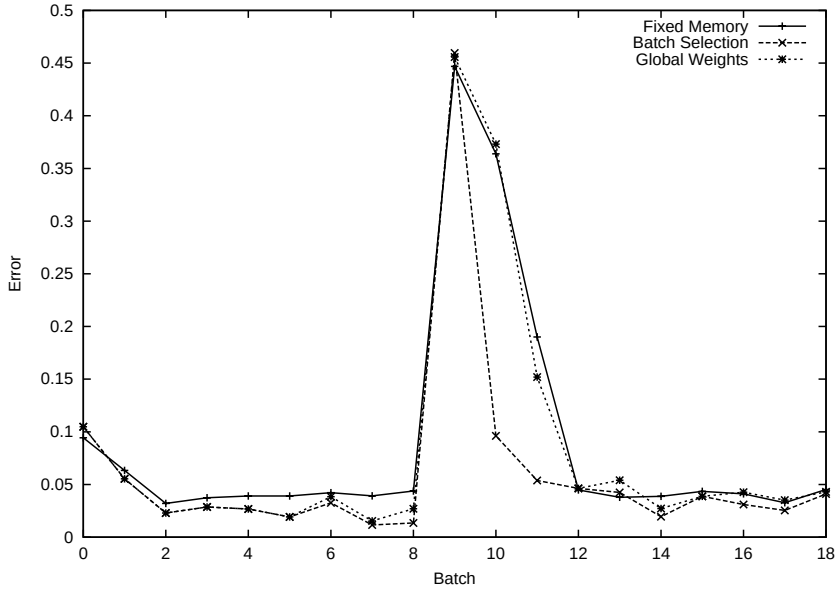


Fig. 8. Classification errors of the best approaches of each type on each batch in scenario A (averaged over 4 runs).

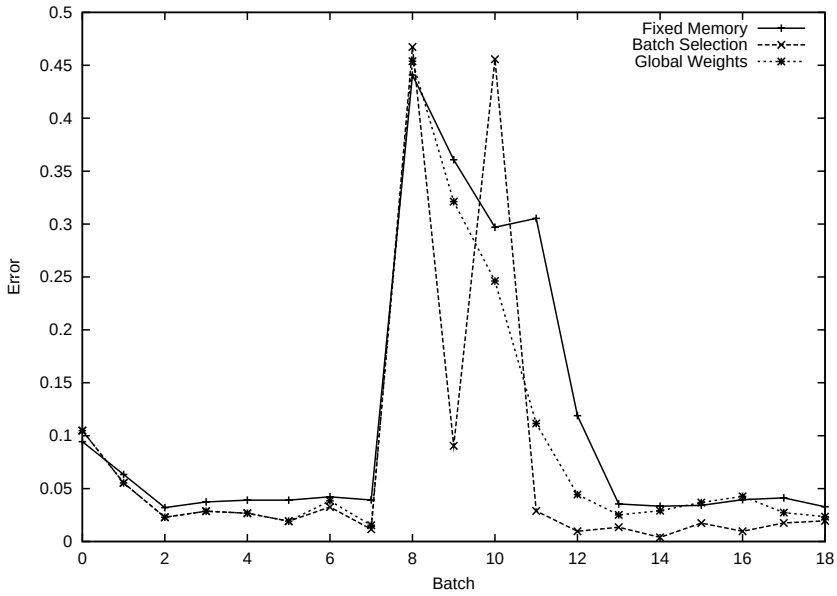


Fig. 9. Classification errors of the best approaches of each type on each batch in scenario C (averaged over 4 runs).

For practical use, though, these non-adaptive approaches are not very useful, as it cannot be determined beforehand, when and how often a concept shift will occur, so an optimal static time window cannot be set. The longer the time window is, the lower error the classifier can achieve if no concept drift occurs, but the shorter the time window is, the faster it will adjust to a new concept. In general, balancing between the two extremes of no and full memory, the fixed size approach seems to work best.

The performance of the example weighting approaches was slightly better than that of the non-adaptive

time window approaches. The selection of the weighting parameter gives the algorithm the ability to better adapt to concept changes. As can be seen in Fig. 6, the different weighting approaches all had very similar results. Unfortunately, as closer inspection of the weighting parameters chosen by the algorithm shows, there seems to be no pattern in the parameters in relation to the occurrence or absence of concept drift, so it is questionable if the performance of this approach is a result of a better model of the importance of the examples over time.

As an alternative to the exponential weighting function used for the example weighting approaches in the experiments described above, one may use different functions. In order to assess the influence of this choice, we also tried the same experiments using a sigmoidal weighting function instead of the exponential one. Assuming that the batches are consecutively numbered with increasing numbers and that the newest batch of labeled examples has the number $t_0$, the weight of an old batch with the number $t < t_0$ is given by the function $tanh((t-a)/b)+1)$, where the best combination of the two parameters $a$ and $b$ is automatically selected from the following two value sets $a \in \{1.00 \cdot t_0, 0.85 \cdot t_0, 0.68 \cdot t_0, 0.50 \cdot t_0\}$ and $b \in \{0.01 \cdot t_0, 0.10 \cdot t_0, 0.30 \cdot t_0, 0.60 \cdot t_0, 5.00 \cdot t_0\}$ at each batch, so that the expected error of the final model learned on all batches is minimized on the newest batch. Interestingly, the results for the sigmoidal weighting scheme do not significantly differ from those of the exponential weighting scheme, and hence are not reported in detail here. This may be an indication that weighting examples is not an appropriate model for concept drift in this case.

The adaptive window and the batch selection approach adjust very well to concept drift. In all three scenarios, the error rate quickly reaches its prior level after a concept drift occurred. In scenario C, the batch selection approach outperforms the adaptive window method, because the more flexible way of selecting the final training set allows it to exclude the two outlier batches and use all other data, while the adaptive window method can only use the information before the first concept shift if it also includes the outliers in the middle.

Summing up, the batch selection strategy achieves the lowest error of all tested approaches (Fig. 8). An explanation for this may be, that outliers, even if there a relatively few and they receive a low weight, seriously hurt the performance of the SVM classification. Since the special properties of text data – very high dimensionality and linear separability – make it easy to identify large groups of outliers, the batch selection method can reliably choose the largest possible set of training examples that are useful to construct the final hypothesis.

## 6. Incremental learning of drifting concepts

While the machine learning approaches proposed in this paper can very well be used online in applications, they are not incremental in nature. Section 3.2 pointed to some publications on incremental support vector machines [3,29], but the approaches presented in this paper only use non-incremental support vector learning for several reasons.

Firstly, as pointed out in Section 3.2, these incremental SVM approaches do not address the problem of concept drift. They implicitly assume a stable distribution of examples and their labels. In case of concept drifts this assumption is of course not valid and hence these approaches would need to be significantly modified. Secondly, in the domain at hand, the non-incremental approaches were efficient enough to be used in an online fashion, even handling thousands of examples, and hence the potential speed-up gained by using an incremental technique did not seem very significant. And thirdly, since incremental learning sometimes means to sacrifice at least some accuracy as compared to non-incremental learning (see e.g. [29]), an incrementalization of the proposed methods did not seem appealing enough so far.

However, if one wants to incrementalize the proposed methods, e.g. in-order to keep the memory and time consumption of the methods at a lower level and to thereby improve the scalability of these methods to very large scale applications, there are three simple extensions that can be taken into this direction:

1. Limit the number of past examples that may be considered for learning by defining a maximum time window size on the examples that may be considered or by discarding examples whose weight falls be below a pre-defined threshold, respectively: This requires almost no change in the implementation and sets an upper bound on the memory and computation time requirements of the example selection and weighting approaches, but may of course mean some loss of accuracy, because the generalization may drop with fewer training examples.
2. Employ an incremental SVM approach as described in [29] or [25] and from the old training examples store only those that are support vectors: The same non-incremental *mySVM* implementation as for the experiments described in this paper could be used, if the outer concept drift framework for weighting or selecting examples as described in this paper is extended to take care of discarding old examples that are no support vectors from their batches. Of course this may again lead to a loss of accuracy in the results as it does in the case without (real) concept drift [26,29].
3. Employ the incremental SVM training approach by Cauwenberg and Poggio [3] and use the outer concept drift framework for selecting examples (or choosing a time window size) to decide which old examples to decrement from the current hypothesis: This approach does not suffer from a loss of accuracy as compared to the non-incremental approach, but it is computationally more expensive than the approach above (2.).

These three alternatives are all rather straight forward and turn the concept drift frameworks proposed in this paper into versions with bounded resource requirements (1.) and incremental versions (2. and 3.), respectively.

## 7. Summary and conclusions

This paper proposed several methods for handling concept drift with support vector machines using different strategies to account for the different importance of examples for the current target concept to be learned. The methods either maintain an adaptive time window on the training data [13], select representative training examples, or weight the training examples [15]. The key idea is to automatically adjust the window size, the example selection, and the example weighting, respectively, so that the estimated generalization error is minimized. The approaches are both theoretically well-founded as well as effective and efficient in practice. Since they do not require complicated parameterization, they are simpler to use and more robust than comparable heuristics. Furthermore we described how the proposed example selection and weighting approaches can be turned into incremental approaches.

Summarizing the results of the concept drift experiments in the information filtering domain, one can observe that example selection by an adaptive time window or by batch selection seems to work better than a gradual weighting scheme depending on the age of and/or performance on the training examples. The importance of an example can easily and successfully be expressed by including it in or excluding it from the training set. A continuously valued representation of the importance or fit of the example does not provide additional useful information. It remains to be investigated, how much the reported results depend on the special properties of text data and the way the concept drift scenarios were simulated. In order to judge the transferability of the proposed approaches to other concept drift problems, experiments on further domains with different properties should be performed.

Since most evaluation methods for machine learning, like e.g. cross-validation, assume that examples are independent and identically distributed, which is clearly unrealistic in the case of concept drift, alternative evaluation schemes were used to estimate and optimize the performance of each learning step within the concept drift handling frameworks (internal evaluation) as well as to evaluate and compare the different frameworks (external evaluation) using $\xi\alpha$-estimators and repeated concept drift scenario simulations, respectively.

## Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center on Computational Intelligence (SFB 531) at University of Dortmund.

## References

[1] J. Allan, Incremental relevance feedback for information filtering, in: *Proceedings of the Nineteenth ACM Conference on Research and Development in Information Retrieval*, H.P. Frei, ed., New York, NY, USA. ACM Press, 1996, pp. 270–278.

[2] M. Balabanovic, An adaptive web page recommendation service, in: *Proceedings of the First International Conference on Autonomous Agents*, W.L .Johnson, ed., New York, NY, USA. ACM Press, 1997, pp. 378–385.

[3] G. Cauwenberghs and T. Poggio, Incremental and decremental support vector machine learning, in: *Advances in Neural Information Processing Systems* (*NIPS-2000*), (vol. 13), Cambridge, MA, USA. MIT Press, 2001.

[4] W.W. Cohen, Learning rules that classify e-mail, in: *Proceedings of the 1996 AAAI Spring Symposium on Machine Learning in Information Access* (*MLIA '96*), Stanford, CA, USA. AAAI Press, 1996.

[5] S. Fischer, R. Klinkenberg, I. Mierswa and O. Ritthoff, YALE: Yet Another Learning Environment – Tutorial. Technical Report CI-136/02, Collaborative Research Center 531 (SFB 531 Computational Intelligence), University of Dortmund, Germany, 2002, http://yale.cs.uni-dortmund.de/.

[6] D.P. Helmbold and P.M. Long, Tracking drifting concepts by minimizing disagreements, *Machine Learning* **14**(1) (1994), 27–45.

[7] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, in: *Proceedings of the European Conference on Machine Learning* (*ECML*), C. Nédellec and C. Rouveirol, eds, Berlin, Germany. Springer, 1998.

[8] T. Joachims, Estimating the generalization performance of a SVM efficiently, in: *Proceedings of the Seventeenth International Conference on Machine Learning* (*ICML-2000*), P. Langley, ed., San Francisco, CA, USA. Morgan Kaufmann, 2000, pp. 431–438.

[9] T. Joachims, *The Maximum-Margin, Approach to Learning Text Classifiers: Methods, Theory, and Algorithms, PhD thesis*, Computer Science Department, University of Dortmund, Germany, 2001.

[10] T. Joachims, D. Freitag and T. Mitchell, Web-Watcher: A tour guide for the world wide web, in: *Proceedings of International Joint Conference on Artificial Intelligence* (*IJCAI-97*), (vol. 1), Morgan Kaufmann, 1997, pp. 770–777.

[11] R. Klinkenberg, *Maschinelle Lernverfahren zum adaptiven Informationsfiltern bei sich verändernden Konzepten*, Masters thesis, Computer Science Department, University of Dortmund, Germany, 1998.

[12] R. Klinkenberg, *Informed parameter setting for support vector machines: Using additional user knowledge in classification tasks*, Technical Report CI-126/02, Collaborative Research Center 531 (SFB 531 Computational Intelligence), University of Dortmund, Germany. ISSN, 2002, 1433–3325.

[13] R. Klinkenberg and T. Joachims, Detecting concept drift with support vector machines, in: *Proceedings of the Seventeenth International Conference on Machine Learning* (*ICML*), P. Langley, ed., San Francisco, CA, USA. Morgan Kaufmann, 2000, pp. 487–494.

[14] R. Klinkenberg and I. Renz, Adaptive information filtering: Learning in the presence of concept drifts, in: *Workshop Notes of the ICML/AAAI-98 Workshop on Learning for Text Categorization held at the Fifteenth International Conference on Machine Learning* (*ICML-98*), M. Sahami, M. Craven, T. Joachims and A. McCallum, eds, Menlo Park, CA, USA. AAAI Press, 1998, pp. 33–40.

[15] R. Klinkenberg and S. Rüping, Concept drift and the importance of examples, in: *Text Mining – Theoretical Aspects and Applications*, J. Franke, G. Nakhaeizadeh and I. Renz, eds, Springer, Berlin, Germany, 2003.

[16] A. Kuh, T. Petsche and R.L. Rivest, Learning timevarying concepts, in: *Advances in Neural Information Processing Systems*, (vol. 3), San Mateo, CA, USA. Morgan Kaufmann, 1991, pp. 183–189.

[17] G. Kunisch, *Anpassung und Evaluierung statistischer Lernverfahren zur Behandlung dynamischer Aspekte in Data Mining*, Masters thesis, Computer Science Department, University of Ulm, Germany, 1996.
[18] K. Lang, NewsWeeder: Learning to filter netnews, in: *Proceedings of the Twelfth International Conference on Machine Learning* (*ICML '95*), San Francisco, CA, USA. Morgan Kaufmann, 1995, pp. 331–339.
[19] C. Lanquillon, *Dynamic neural classification. Masters thesis*, Computer Science Department, University of Braunschweig, Germany, 1997.
[20] A. Lunts and V. Brailovskiy, Evaluation of attributes obtained in statistical decision rules, *Engineering Cybernetics* **3** (1967), 98–109.
[21] I. Mierswa, R. Klinkenberg, S. Fischer and O. Ritthoff, A Flexible Platform for Knowledge Discovery Experiments: Yale – Yet Another Learning Environment, in: *LLWA-2003: Lehren – Lernen – Wissen – Adaptivität*, Proceedings of the Workshop of the Special Interest Groups Machine Learning, Knowledge Discovery, and Data Mining (FGML), Intelligent Tutoring Systems (ILLS), and Adaptivity and User Modeling in Interactive Systems (ABIS) of the German Computer Science Society (GI), University of Karlsruhe, Karlsruhe, Germany, 2003.
[22] T. Mitchell, R. Caruana, D. Freitag, J. McDermott and D. Zabowski, Experience with a learning personal assistant, *Communications of the ACM* **37**(7) (1994), 81–91.
[23] O. Ritthoff, R. Klinkenberg, S. Fischer, I. Mierswa and S. Felske, YALE: Yet Another Machine Learning Environment, in: *LLWA 01 – Tagungsband der GI-Workshop-Woche Lernen – Lehren – Wissen – Adaptivität*, R. Klinkenberg, S. Rüping, A. Fick, N. Henze, C. Herzog, R. Molitor and O. Schröder, eds, Technical Report No. 763, Department of Computer Science, University of Dortmund. http://yale.cs.uni-dortmund.de/, 2003, pp. 84–92.
[24] S. Rüping, *mySVM-Manual. Artificial Intelligence Unit*, Department of Computer Science, University of Dortmund, Germany, 2000, http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.
[25] S. Rüping, Incremental learning with support vector machines, in: *Proceedings of the 2001 IEEE International Conference on Data Mining*, N. Cercone, T. Lin and X. Wu, eds, 2001, pp. 641–642. IEEE.
[26] S. Rüping, *Incremental learning with support vector machines*, Technical Report 18, Collaborative Research Center 475 (SFB 475 Reduction of Complexity for Multivariate Data Structures), University of Dortmund, Dortmund, Germany, 2002.
[27] G. Salton and C. Buckley, Term weighting approaches in automatic text retrieval, *Information Processing and Management* **24**(5) (1988), 513–523.
[28] A. Smola, B. Schölkopf and K.-R. Müller, General cost functions for support vector regression, in: *Proceedings of the 8th International Conference on Artificial Neural Networks*, L. Niklasson, M. Boden and T. Ziemke, eds, 1998.
[29] N.A. Syed, H. Liu and K.K. Sung, *Handling concept drifts in incremental learning with support vector machines*, In Proceedings of the Fifth ACM SIG KDD International Conference on Knowledge Discovery and Data Mining (KDD-99), New York, NY, USA. ACM Press, 1999.
[30] N.A. Syed, H. Liu and K.K. Sung, *Incremental learning with support vector machines*, in Proceedings of the Workshop on Support Vector Machines at the International Joint Conference on Articial Intelligence (IJCAI-99), Stockholm, Sweden, 1999.
[31] C. Taylor, G. Nakhaeizadeh and C. Lanquillon, Structural change and classification, in: *Workshop Notes of the ECML-97 Workshop on Dynamically Changing Domains: Theory Revision and Context Dependence Issues held at the Ninth European Conference on Machine Learning*, G. Nakhaeizadeh, I. Bruha and C. Taylor, eds, 1997, pp. 67–78.
[32] V. Vapnik, *Statistical Learning Theory*, Wiley, Chichester, GB, 1998.
[33] G. Veltmann, *Einsatz eines Multiagentensystems zur Erstellung eines persönlichen Pressespiegels. Master's thesis*, Computer Science Department, University of Dortmund, Germany, 1997.
[34] G. Widmer and M. Kubat, Learning in the presence of concept drift and hidden contexts, *Machine Learning* **23**(2) (1996), 69–101.