Concept Drift and the Importance of Examples

Ralf Klinkenberg[1] and Stefan Rüping[1]

University of Dortmund
Computer Science Department, LS VIII
44221 Dortmund, Germany

http://www-ai.cs.uni-dortmund.de/

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.

Examples may be important for different reasons. In case of a drifting concept, the importance of an example obviously depends on its age. If a user is interested in several topics, these may be of different importance to her/him. Hence the importance of an example is influenced by the topic it belongs to. Of course these two effects may cumulate.

In this paper we model the importance of an example by weighting its importance for the final decision function. This paper investigates how to handle these two effects with support vector machines extending the approach of [11], which showed that drifting concepts can be learned effectively and efficiently with little parameterization. Several approaches addressing the different effects are compared in experiments on real-world text data.

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. 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.

In this paper, we discuss several approaches to deal with the problem of concept drift. The central question will be, how important older examples are for predicting new instances of the possibly changed concept. Examples may be important for different reasons. In case of a drifting concept, the importance of an example obviously depends on its age. If a user is interested in several topics, these may be of different importance to her/him. Hence the importance of an example is influenced by the topic it belongs to. Of course these two effects may cumulate.

We will extend the approach of [11] by using some special properties of Support Vector Machines, that will prove useful in handling concept drift, for example efficient performance estimation, transduction, and examples weighting. A main goal will be to keep the learning algorithm as effectively, efficiently, and with as little parameterization as possible. Several approaches addressing the different effects are compared in experiments on real-world text data.

2 Concept Drift

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. In machine learning, changing concepts are often handled by time windows of fixed or adaptive size on the training data [18,27,15,12] or by weighting data or parts of the hypothesis according to their age and/or utility for the classification task [14,24]. 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 [9].

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 [4] 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 [13] 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 [27,15,12]. 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.

Sayed, Liu, and Sung [23] describe an approach to incrementally learning support vector machines that handles *virtual* concept drift implied by incrementally learning from several subsamples of a large training set, but they do not address the problem of (*real*) concept drift addressed here.

3 Support Vector Machines

We use Support Vector Machines 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 [26]), therefore we will give only a

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

short introduction of the concepts that are important with respect to our work.

Support Vector Machines are based on the structural risk minimization principle[26] 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

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

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 $(x_i, y_i)_{i=1...n}$ and a complexity term $||w||^2$

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

In their basic formulation, SVMs 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 generalization performance), see Figure 1.

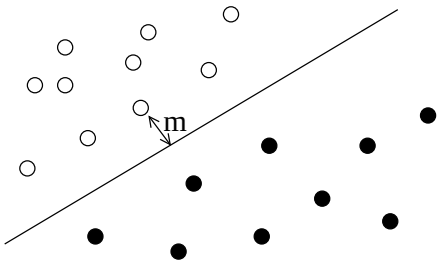

Fig. 1. Maximum margin hyperplane and margin m.

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

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

subject to

$$y_i(w^T x_i + b) \leq 1 - \xi_i, i = 1, \ldots, n \quad (4)$$

$$\xi_i \geq 0, i = 1, \ldots, n \quad (5)$$

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 ξ_i are slack variables that allow for a certain amount of misclassification in equation (4). In practice, this optimization problem can be efficiently solved in its dual form

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

$$\rightarrow \min$$

subject to

$$0 \leq \alpha_i \leq C \quad \forall i = 1 \ldots n \quad (7)$$

$$\sum_{i=1}^{n} \alpha_i = 0 \quad (8)$$

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 α_i are called the Support Vectors.

3.1 Loss Functions

In equation (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 [22]). In particular, one can set an individual weight w_i to each example by replacing definition (3) by

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

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

$$0 \leq \alpha_i \leq w_i C \quad \forall i = 1 \ldots n \quad (10)$$

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 SVMs for Text Classification

It was first noted by Joachims in [5], 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 Support Vector Machines are currently the most successful tool for text classification [8].

4 The Importance of Examples

In a learning problem with drifting concepts as introduced in section 2, we face the problem to decide, how much information from past examples can 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, that differ in the way previous examples are used to construct a new hypothesis. All 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 [6], 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

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 [18] or involve complicated heuristics [27,15,12]. 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 [11], Klinkenberg and Joachims presented 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, a special form of $\xi\alpha$-estimates [6] is used. $\xi\alpha$-estimates are a particularly efficient method for estimating the performance of an SVM.

The window 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.

$$z_{(t,1)}, \ldots, z_{(t,m)} \tag{11}$$

$$z_{(t-1,1)}, \ldots, z_{(t-1,m)}, z_{(t,1)}, \ldots, z_{(t,m)} \tag{12}$$

$$z_{(t-2,1)}, \ldots, z_{(t-2,m)}, z_{(t-1,1)}, \ldots, z_{(t-1,m)}, z_{(t,1)}, \ldots, z_{(t,m)} \tag{13}$$

$$\vdots$$

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

$$Err^m_{\xi\alpha}(h_{\mathcal{L}}) = \frac{|\{i : 1 \leq i \leq m \wedge (\alpha_{(t,i)} R^2_{\Delta} + \xi_{(t,i)}) \geq 1\}|}{m} \tag{14}$$

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: S_{train} training sample 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 w_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.

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 not one but highly informative).

In the weighting scheme, we select the weights of the examples solely based on their respective age, for example using a exponential aging function $w_\lambda(x) = \exp(-\lambda t)$, where x was found t time steps ago. The larger λ 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 λ and pick the best λ 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 [11] 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

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 often in text classification tasks the classes 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 a 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

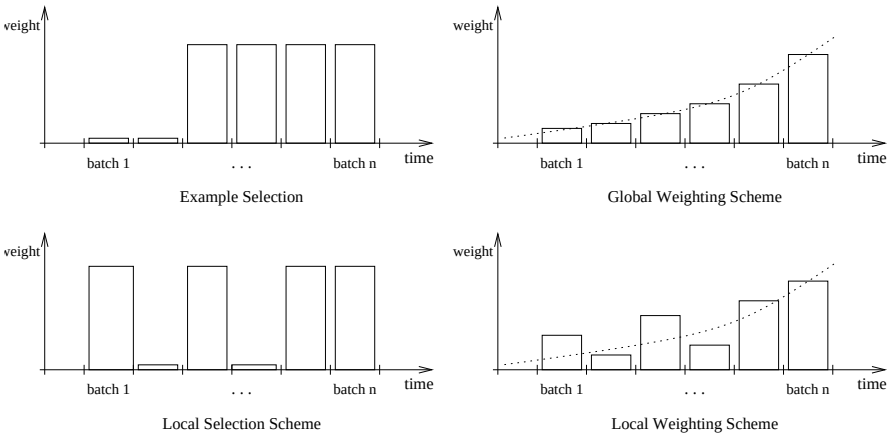

Fig. 2. Comparison of the example weights in the four weighting schemes.

from the same examples 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 (the increase depending on the size of this topic in relation to the other topics). A good example of this behavior can be seen in Figure 3. 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.

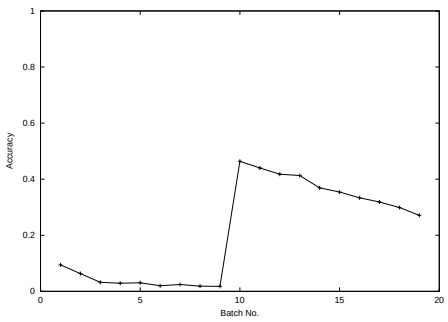

Fig. 3. Typical Classification Error in a Concept Drift Setting.

This observations 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 model (i. e. the same users interest) as the recent batch, by comparing the estimated leave-one-out error of the classifier on the 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 classifiers error can now 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, which have a significantly higher classification error than the most recent batch. By this, we hope to train the final classifier on all data generated by the current model in a way similar to the examples selection scheme in section 4.1. We can also 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. Of course, we can also combine both weighting schemes by multiplying the weights of each example. 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).

4.4 Transduction

Transductive learning [26] differs from inductive learning in that its goal is not to find a hypothesis which is optimal with respect to all data, i.e. the real probability distribution of the examples, but to only find a hypothesis which is optimal for a given test set. That is, transductive learning does not only learn from a given example set, but also from a set of observations (without given classification), whose values are to be predicted. Transduction can improve the learners performance considerably, since a much better estimation of the observations distribution $P(x)$ can be obtained from the training- and test set together (of course, $P(y|x)$ can be only estimated on the training set).

In [7], Joachims shows that the principle of transductive learning is very well suited for the problem of text classification. For example, in information retrieval systems the collection of all available documents is usually known, but the user can only label a very small fraction of all documents as being relevant or not relevant. In the same publication, Joachims also shows how transductive learning can be efficiently performed with Support Vector Machines.

Empirical results (see e.g. [20], [7], [17]) show that unlabeled data can help to significantly improve the performance of text classifiers, especially in case of few labeled examples. As pointed out in [7], it is well known in information

retrieval that words in natural language occur in strong co-occurence patterns (see [25]). 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.

Transduction and Concept Drift Transductive learning has also been applied to the setting of concept drift by Klinkenberg in [10] using the following idea. At each batch, there are usually only comparatively few observations, for which predictions need to be made. Since the performance improvements achieved by transduction are the more significant the more data is used, only using such a small set of unlabeled data does not seem optimal. But the out-of-date examples, whose labels do no longer seem to be representative enough for the current concept, and may be some other unlabeled examples from the same source are still available. Assuming that the process that generated the data (e.g. the news stream) is still approximately the same and that only the user's preferences in what is relevant to him or her and what not changed, the x-values of this examples can be used as unclassified observation for transductive learning. In this way, even for very new topics with little examples, a large collection of documents can be used to estimate $P(x)$.

Klinkenberg [10] describes an extension of [11] (see also section 4.1) exploiting unlabeled and old no longer reliably labeled data in such a transductive way. Its basic idea is to first use the algorithm described in [11] to find a good 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.

Why are separate window sizes $win_{labeled}$ and $win_{unlabeled}$ maintained for labeled and unlabeled data respectively? The probability $P(y|x)$, 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 $P(x)$, 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.

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 $z_{(t-h,1)}, ..., z_{(t,m')}$, considering all training examples outside the window of size $win_{labeled}$ as unlabeled, and on the test examples $z_{(t+1,1)}, ..., z_{(t+1,m')}$

  - compute $\xi\alpha$-estimate on examples $z_{(t+1,1)}, \cdots, z_{(t+1,m')}$
- output: window size which minimizes $\xi\alpha$-estimate ($win_{unlabeled}$)

Other approaches for exploiting unlabeled data: Besides of transduction, there are also other (semi-)supervised approaches for exploiting unlabeled data. Nigam et al. [19,20] 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 [17] 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 [3] 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.

One of the situations, in which a user may change his or her preferences, may occur, when the documts available to the user change, i.e. when the distribution $P(x)$ changes. Lanquillon [16] presents an approach to make use of unlabeled data for the detection of concept drift in such situations.

4.5 Multiple Topics

Until now, we only talked about relevant and irrelevant examples without any further distinction. But in reality, a set of documents (news article, newsgroup postings, emails,...) will contain a whole set of topics and there usually will be more than one relevant topic, for example business email and private email in contrast to spam (of course, in reality the topics will be much more fine-grained).

In the usual task of text classification with fixed concepts, this is not a major problem, because it can be observed that due to the very high dimension it is usually possible to separate multiple relevant topics from the remaining texts as well as it is with single topics.

The difference of multiple topics with concept drift is, that different topics usually will not become relevant or irrelevant at the same time. For examples, if the user takes another job, the relevance of a certain business email may drastically change, but the relevance of private email will still be the same.

In this situation it may help to cluster the documents beforehand and to learn on each cluster separately. The final decision rule will be to mark an examples as relevant, if any of the individual decision rules marks this example as relevant.

The performance of this approach depends largely on the quality of the clustering algorithms results. Not only do we need meaningful clusters, we also need clusters that remain stable over time. We can hope to find these

clusters, if the distribution $P(x)$ remains constant and only $P(y|x)$, i.e. the users interest, changes. In all other cases, the clustering will have to be repeated from time to time.

5 Experiments

5.1 Experimental Setup

In order to evaluate the learning approaches for drifting concepts proposed in this paper, the four simple data management approaches are compared to the adaptive time window approach and 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.
- Window of *"Adaptive Size"*: The window adjustment algorithm [11] proposed in section 4.1 adapts the window size to the current concept drift situation.
- *"Global Weights"*: The examples of old batches are weighted by an exponential weighting function according to their age, so that older example 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 old batch is inversely proportional to the error rate of that batch on this model ($w_{local}(batch) := 1 - 5 * error(batch) - 0.1$, where the weight is set to one for error rates below 10% and to zero for error rates above 30%, see section 4.2).
- *"Combined Weights"*: The examples are weighted by the product of the global and the local weight of their batch.
- *"Zero-One-Weights"* or "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 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 [21] 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.

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

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.[2] 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.

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

Table 2. ICML-2000: 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%)

...

5.2 Experimental Results

...

...

TO DO:

- *Parameter-Variationen mySVM (C = 1,10,100,1000) (linearer Kernel = std. for TCat) (lambda in der Gewichtungsfunktion exp(-lambda * age[batches]) automatisch aus 0.01, 0.1, 0.2, 0.4, 0.6, 1.0, 2.0, 4.0 bestimmt) (exponential weight decay)*

Table 3. Daimler-WS-2002: 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	xx.xx%	x.xx%	x.xx%	x.xx%
	(x.xx%)	(x.xx%)	(x.xx%)	(x.xx%)
Recall	xx.xx%	xx.xx%	xx.xx%	xx.xx%
	(x.xx%)	(x.xx%)	(x.xx%)	(x.xx%)
Precision	xx.xx%	xx.xx%	xx.xx%	xx.xx%
	(x.xx%)	(x.xx%)	(x.xx%)	(x.xx%)
Scenario B:				
Error	xx.xx%	x.xx%	x.xx%	x.xx%
	(x.xx%)	(x.xx%)	(x.xx%)	(x.xx%)
Recall	xx.xx%	xx.xx%	xx.xx%	xx.xx%
	(x.xx%)	(x.xx%)	(x.xx%)	(x.xx%)
Precision	xx.xx%	xx.xx%	xx.xx%	xx.xx%
	(x.xx%)	(x.xx%)	(x.xx%)	(x.xx%)
Scenario C:				
Error	xx.xx%	x.xx%	x.xx%	x.xx%
	(x.xx%)	(x.xx%)	(x.xx%)	(x.xx%)
Recall	xx.xx%	xx.xx%	xx.xx%	xx.xx%
	(x.xx%)	(x.xx%)	(x.xx%)	(x.xx%)
Precision	xx.xx%	xx.xx%	xx.xx%	xx.xx%
	(x.xx%)	(x.xx%)	(x.xx%)	(x.xx%)

- *Tabelle mit Error, Recall, Preccision, evtl. F1, std.dev.*
- *Plot ueber Zeit (Batches), a la ICML-2000: FullMem, NoMem, FixedSize, AdaptiveSize, (ZeroOneWeight)*
  Scen. A+B+C
- *Plot ueber Zeit (Batches), a la ICML-2000: (AdaptiveSize), ZeroOneWeight, Expo.Local, Expo.Global, Expo.Combi*
  Scen. A+B+C

...

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 importance of this choice,

we also tried the same experiments using a sigmoidal weighting function instead 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 $a \in \{1.00 * t_0, 0.85 * t_0, 0.68 * t_0, 0.50 * t_0\}$ and $b \in \{0.01 * t_0, 0.10 * t_0, 0.30 * t_0, 0.60 * t_0, 5.00 * t_0\}$, 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.

Summarizing the results of the concept drift experiments in this information domain, one can observe that example selection by an adaptive time window or a zero-one-weighting scheme seems to work better than a gradual weighting scheme depending on the age of and/or performance on the training examples.

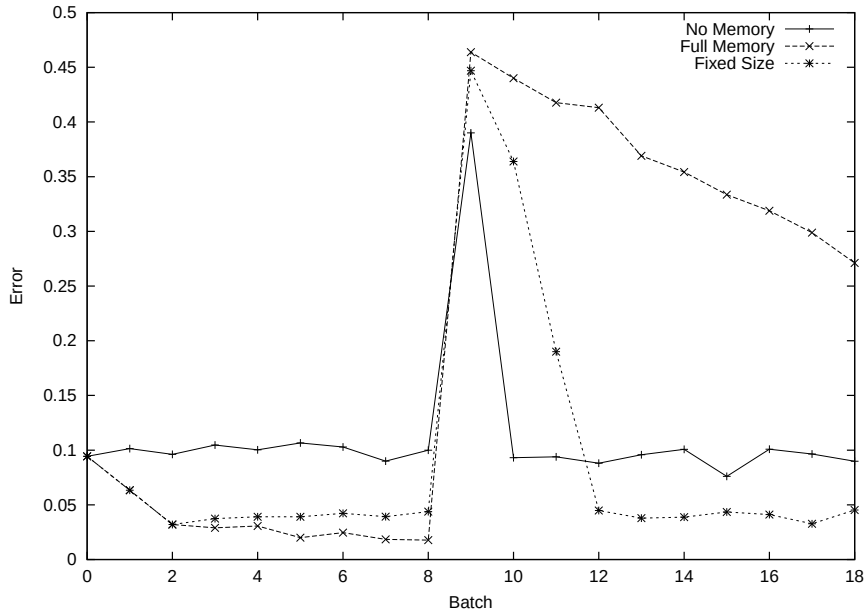

Fig. 4. Classification Errors of the Trivial Approaches on each Batch in Scenario A.

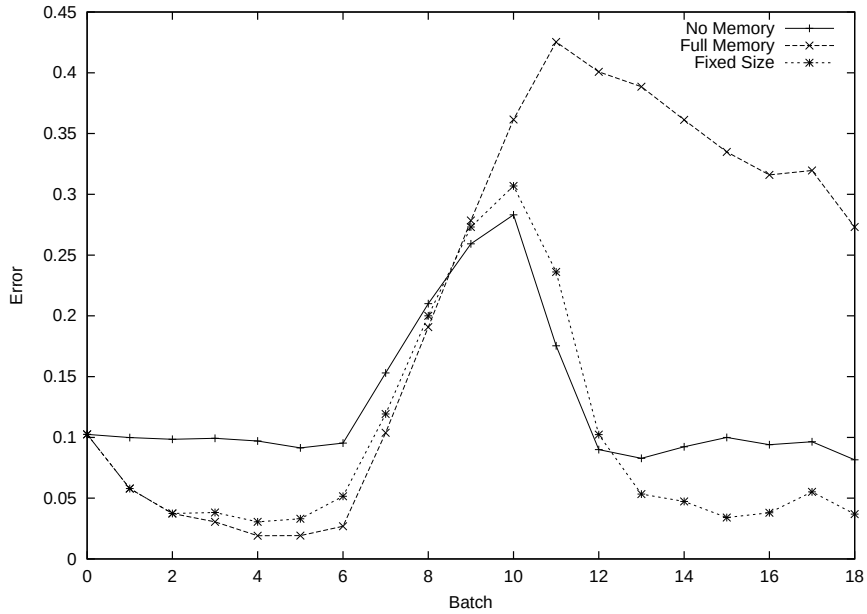

Fig. 5. Classification Errors of the Trivial Approaches on each Batch in Scenario B.

6 Summary and Conclusions

In this paper, we proposed several methods for handling concept drift with support vector machines using different strategies to account for the different importance of examples to the current target concept to be learned, where the importance of examples may depend on its age or its topic, and where also unlabeled documents may have an impact.

We extended the approach of [11] by using some special properties of SVMs useful in handling concept drift, for example efficient performance estimation, transduction, and examples weighting, keeping the learning algorithm as effectively, efficiently, and with as little parameterization as possible. Several approaches addressing the different effects were compared in experiments on real-world text data.

Acknowledgments

The financial support of the Deutsche Forschungsgemeinschaft (Collaborative Research Centers SFB 475, "Reduction of Complexity for Multivariate Data Structures", and SFB 531, "Computational Intelligence") is gratefully acknowledged.

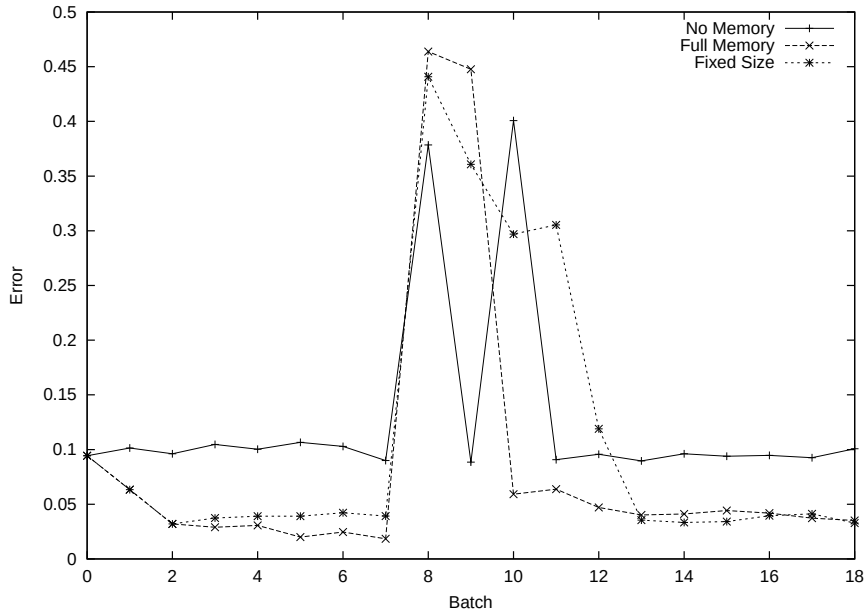

Fig. 6. Classification Errors of the Trivial Approaches on each Batch in Scenario C.

References

1. 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.
2. 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.
3. Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Annual Conference on Computational Learning Theory (COLT-98)*, 1998.
4. David P. Helmbold and Philip M. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14:27–45, 1994.
5. T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of the European Conference on Machine Learning*, pages 137 – 142, Berlin, 1998. Springer.
6. 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.

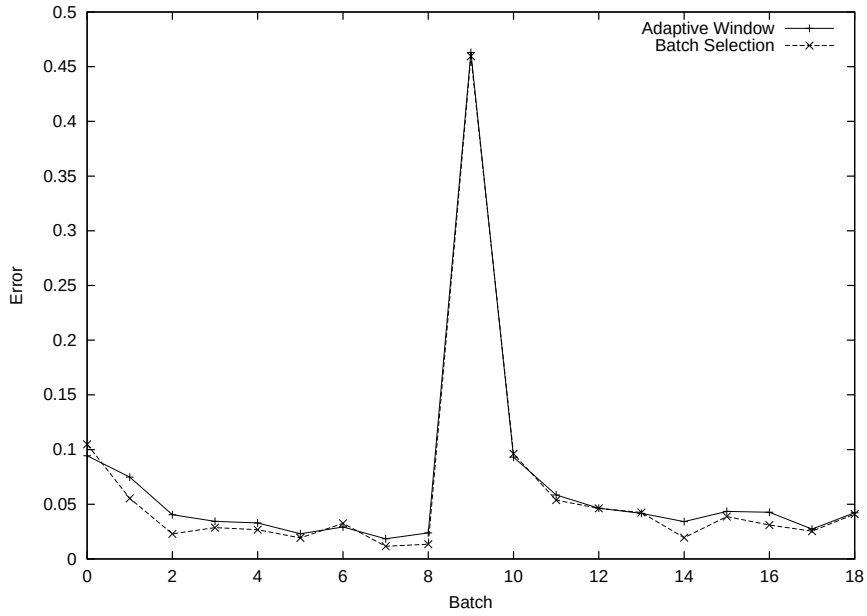

Fig. 7. Classification Errors of the Batch Selection Approaches on each Batch in Scenario A.

7. Thorsten Joachims. Transductive inference for text classification using support vector machines. In *International Conference on Machine Learning (ICML)*, Bled, Slowenien, 1999.
8. Thorsten Joachims. *The Maximum-Margin Approach to Learning Text Classifiers: Methods, Theory, and Algorithms.* PhD thesis, Fachbereich Informatik, Universität Dortmund, 2001.
9. Ralf Klinkenberg. *Maschinelle Lernverfahren zum adaptiven Informationsfiltern bei sich verändernden Konzepten.* Masters thesis, Fachbereich Informatik, Universität Dortmund, Germany, feb 1998.
10. Ralf Klinkenberg. Using labeled and unlabeled data to learn drifting concepts. In Miroslav Kubat and Katharina Morik, editors, *Workshop notes of the IJCAI-01 Workshop on* Learning from Temporal and Spatial Data, pages 16–24, Menlo Park, CA, USA, 2001. IJCAI, AAAI Press. Held in conjunction with the International Joint Conference on Artificial Intelligence (IJCAI).
11. Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 487–494, San Francisco, CA, USA, 2000. Morgan Kaufmann.
12. 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.

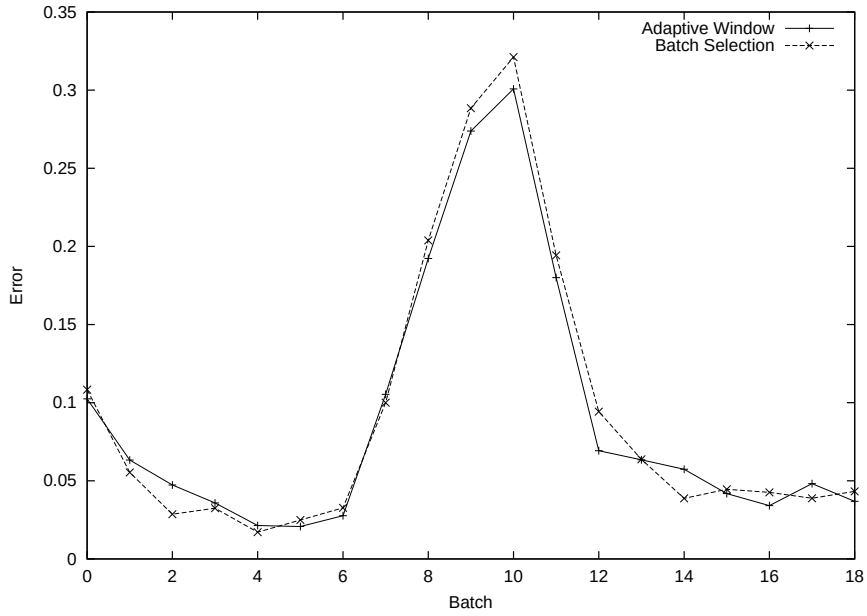

Fig. 8. Classification Errors of the Batch Selection Approaches on each Batch in Scenario B.

13. 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.
14. Gerhard Kunisch. *Anpassung und Evaluierung statistischer Lernverfahren zur Behandlung dynamischer Aspekte in Data Mining.* Masters thesis, Fachbereich Informatik, Universität Ulm, Germany, jun 1996.
15. Carsten Lanquillon. *Dynamic Neural Classification.* Masters thesis, Fachbereich Informatik, Universität Braunschweig, Germany, oct 1997.
16. Carsten Lanquillon. Information filtering in changing domains. In T. Joachims, A. McCallum, M. Sahami, and L. Ungar, editors, *Working Notes of the IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 41–48, Stockholm, Sweden, August 1999.
17. 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.
18. 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.
19. 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.

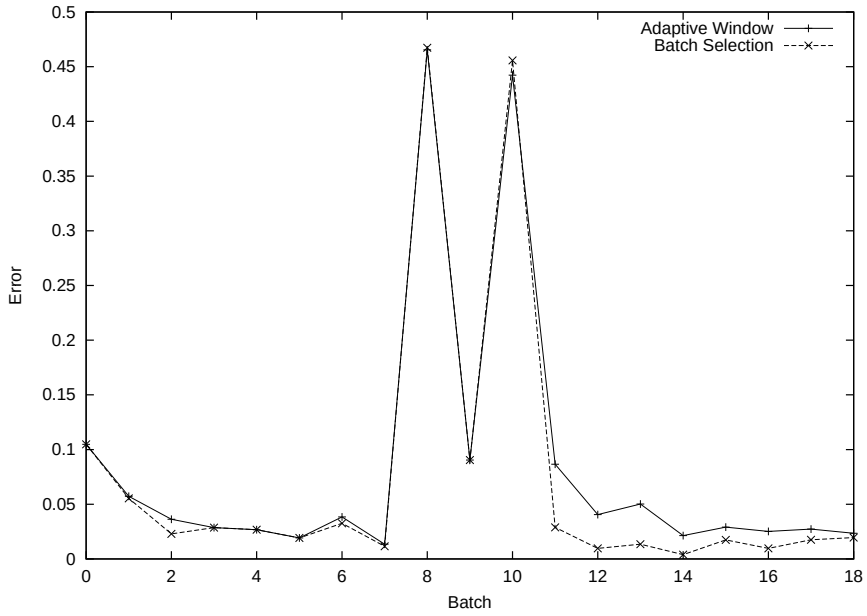

Fig. 9. Classification Errors of the Batch Selection Approaches on each Batch in Scenario C.

20. 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.
21. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
22. A. Smola, B. Schölkopf, and K.-R. Müller. General cost functions for support vector regression. In L. Niklasson, M. Boden, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, 1998.
23. Nadeem Ahmed Syed, Huan Liu, and Kah Kay Sung. Handling concept drifts in incremental learning with support vector machines. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, New York, 1999. ACM Press.
24. 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.
25. 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.
26. V. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, GB, 1998.
27. Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(2):69–101, 1996.

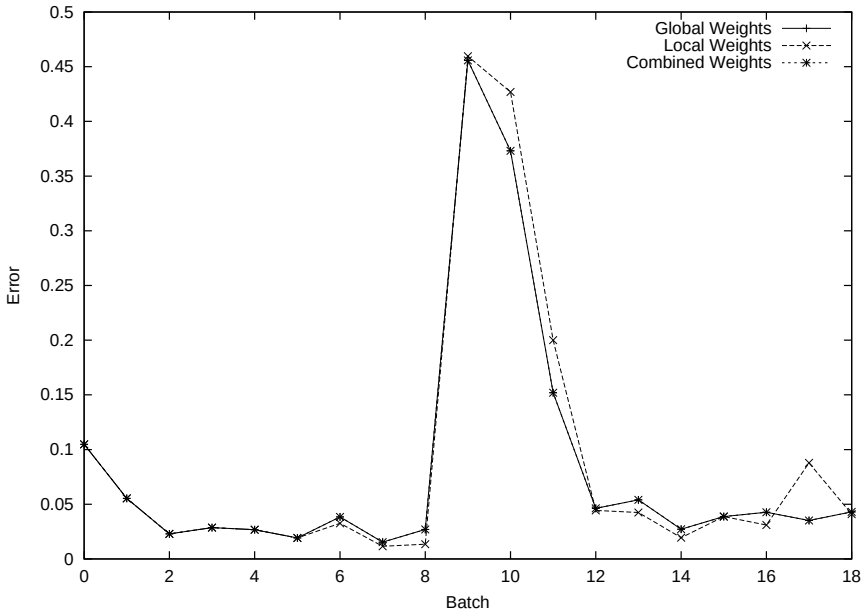

Fig. 10. Classification Errors of the Weighting Approaches on each Batch in Scenario A.

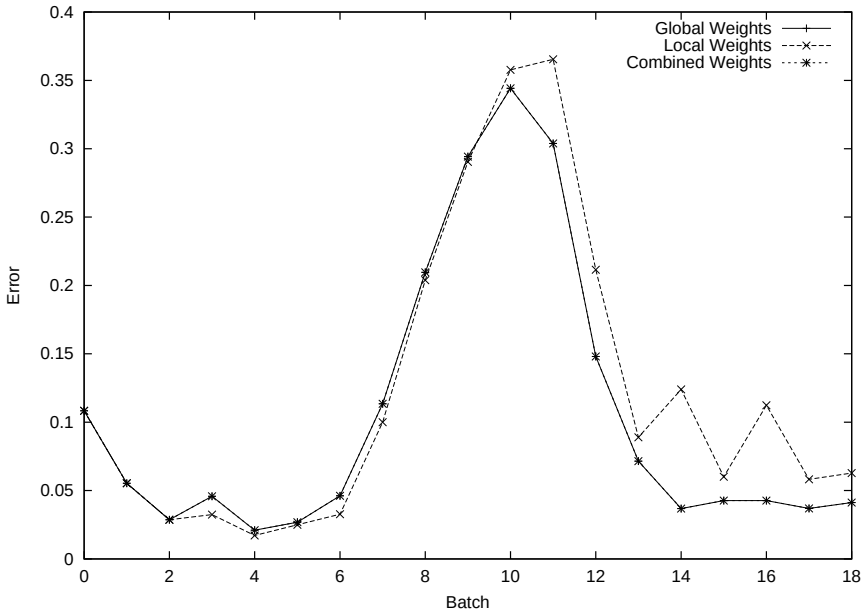

Fig. 11. Classification Errors of the Weighting Approaches on each Batch in Scenario B.

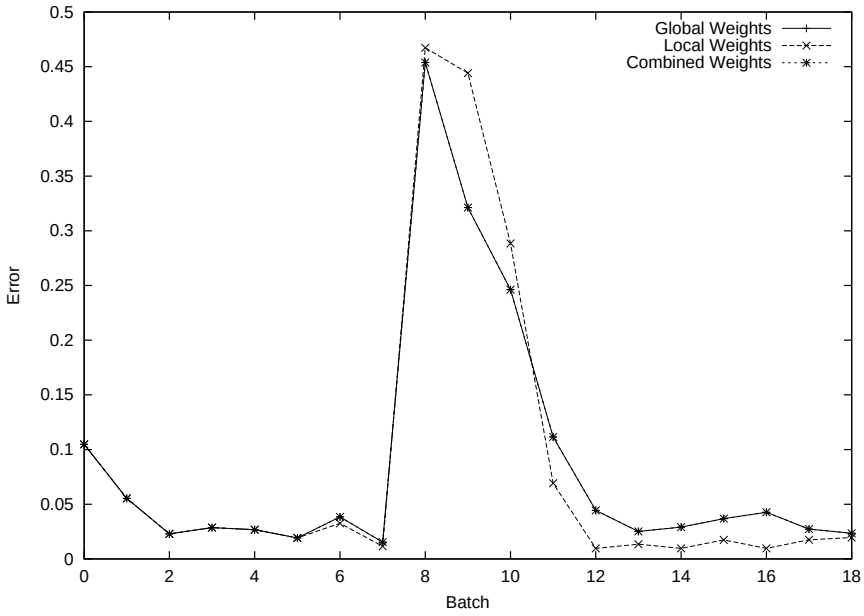

Fig. 12. Classification Errors of the Weighting Approaches on each Batch in Scenario C.

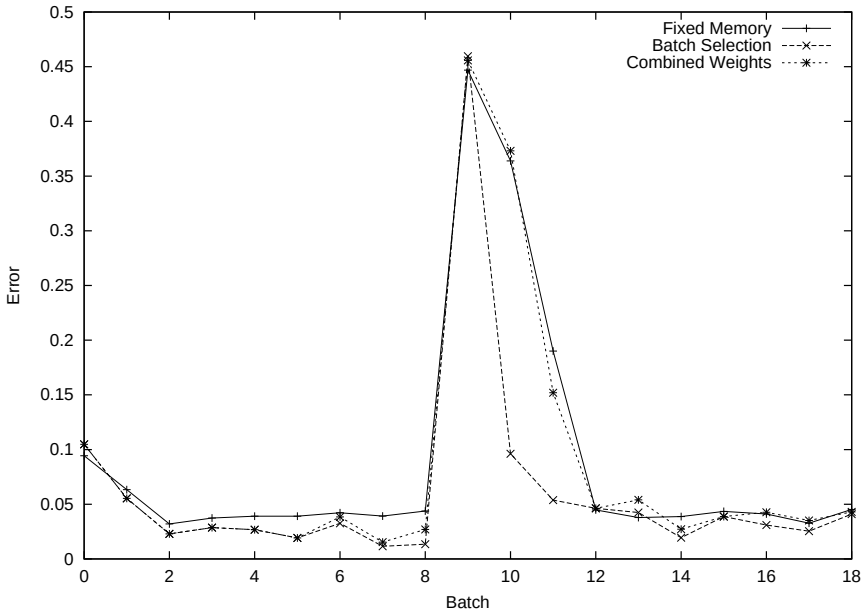

Fig. 13. Classification Errors of the Best Approaches of each Type on each Batch in Scenario A.

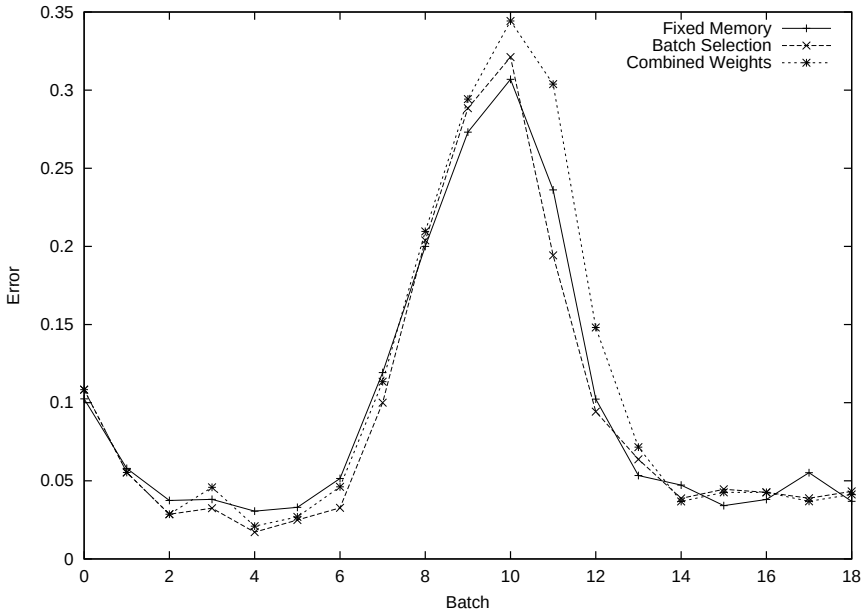

Fig. 14. Classification Errors of the Best Approaches of each Type on each Batch in Scenario B.

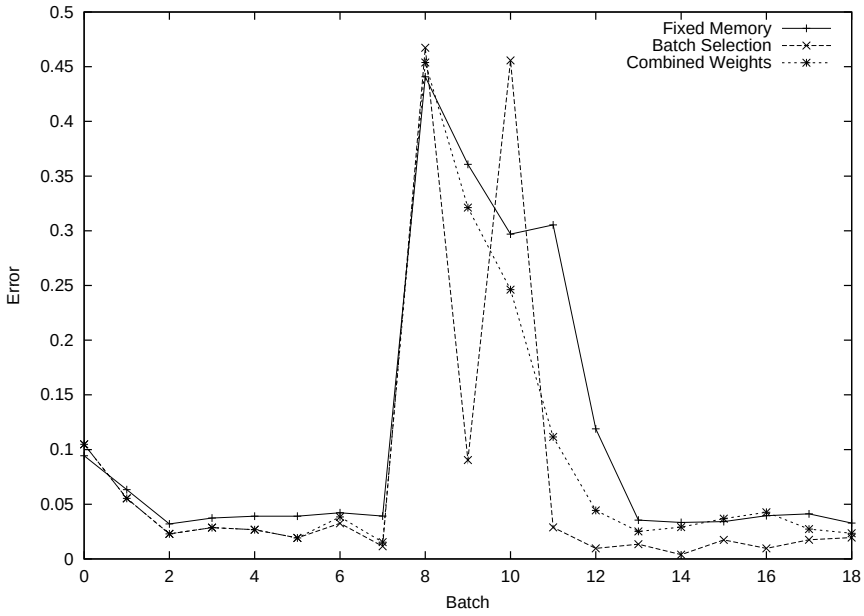

Fig. 15. Classification Errors of the Best Approaches of each Type on each Batch in Scenario C.