# Boosting Classifiers for Drifting Concepts

Martin Scholz and Ralf Klinkenberg

Artificial Intelligence Group, University of Dortmund, 44221 Dortmund, Germany,
{scholz,klinkenberg}@ls8.cs.uni-dortmund.de,
http://www-ai.cs.uni-dortmund.de/

**Abstract.** This paper proposes a boosting-like method to train a classifier ensemble from data streams. It naturally adapts to concept drift and allows to quantify the drift in terms of its base learners. The algorithm is empirically shown to outperform learning algorithms that ignore concept drift. It performs no worse than advanced adaptive time window and example selection strategies that store all the data and are thus not suited for mining massive streams.

## 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. Knowledge discovery and data mining from time-changing data streams and concept drift handling on data streams have become important topics in the machine learning community and have gained increased attention recently as illustrated by numerous conference tracks and workshops on these topics as well as specially dedicated journal issues (see e.g. [38,30,1,15,3,2]).

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 e.g. to build personalized news filters, which learn about the news-reading preferences of a user [33,53], to filter e-mail [9], or to guide a user's search on the World Wide Web [21]. The interest of the user, i.e. the concept underlying the classification of the texts, changes over time. A filtering system should be able to adapt to such concept changes. Machine learning approaches handling this type of concept drift have been shown to outperform more static approaches ignoring it [26].

After formalizing the concept drift problem in Sec. 2.1, previous approaches handling it are reviewed in Sec. 2.2. Sec. 3 discusses related work on ensemble methods for data streams and introduces a boosting-like algorithm for data streams that naturally adapts to concept drift[1]. In Sec. 4 the approach is evaluated on two real-world datasets with different simulated concept drift scenarios, and on one economic dataset exhibiting real concept drift. Sec. 5 summarizes the results and provides an outlook on future work.

## 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 [26,28,24,25]. Each example $z = (x, y)$ consists of a feature vector $x \in \mathcal{X}$ 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.

$$\underbrace{z_{(1,1)}, \ldots, z_{(1,m)}}_{\text{batch } 1}, \underbrace{z_{(2,1)}, \ldots, z_{(2,m)}}_{\text{batch } 2}, \cdots, \underbrace{z_{(t,1)}, \ldots, z_{(t,m)}}_{\text{batch } t}, \underbrace{z_{(t+1,1)}, \ldots, z_{(t+1,m)}}_{\text{batch } t+1}$$

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

[1] A preliminary short version of this paper was presented at the ECML/PKDD-2005 Workshop on *Knowledge Discovery from Data Streams* [47].

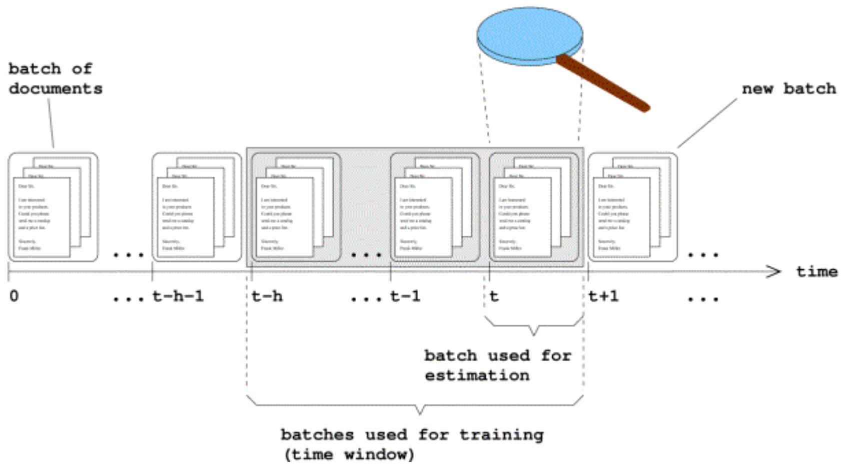


**Fig. 1.** Example of a sliding time window on a stream of text documents: The texts to be classified are assumed to arrive in batches. At time point $t$, a classification model is learned from the data in a window consisting of the $h$ most recent batches. Its performance is estimated on the last available batch only.

---

the example distribution $P_i(x, y)$ and $P_{i+1}(x, y)$ between batches will differ. The learner aims to sequentially predict the labels of the next batch and to minimize the cumulated number of prediction errors. E.g., after batch $t$ the learner can use any subset of the training examples from batches 1 to $t$ to predict the labels of batch $t + 1$.

Fig. 1 illustrates this for the example application of information filtering, were incoming text documents form a stream of documents have to be classified, e.g. into interesting or non-interesting with regard to the interest of a particular user. The documents are assumed to arrive in batches and the correct classification labels for the documents in the batches 1 to $t$ are known to the system. Here a sliding time window consisting of the last $h$ batches, i.e. the batches $t - h$ to $t$, are used to train a classifier, which is then applied to classify the documents in the newly arrived batch $t + 1$, whose labels are not known to the system yet. The most recently received labeled data, batch $t$, is assumed to most closely resemble the current true concept. Hence adaptive time window approaches automatically optimizing the size $h$ of the sliding time window dependent on the current amount of drift and other model selection approaches adaptive to the amount of drift use this last labeled batch $t$ for estimating the performance (prediction error) of their learned models on newly received data (see Sec. 2.3 and Sec. 2.4 for more details.).

### 2.2 Related Work on Concept Drift

In machine learning, changing or drifting concepts are often handled by time windows of fixed or adaptive size on the training data [36,55,34,23,27,19] or by weighting data or parts of the hypothesis according to their age and/or utility for the classification task [32,50]. The latter approach of weighting examples has already been used for information filtering in the incremental relevance feedback approaches of [4] and [5].

For windows of fixed size, the choice of a *"good" window size* is a compromise between fast adaptability (small window) and good generalization in phases without concept change (large window). A fixed window size makes strong assumptions about how quickly the concept changes. The basic idea of *adaptive window management* is to adjust the window size to the current extent of concept drift. While heuristics can adapt to different speed and amount of drift, they involve many parameters that are difficult to tune.

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[2], 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 [17,18] 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

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

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

In practice, however, it usually cannot be guaranteed that the application at hand obeys these restrictions, e.g. a reader of electronic news may change his interests (almost) arbitrarily often and radically. Furthermore the large time window sizes, for which the theoretical results hold, would be impractical. Hence more application oriented approaches rely on far smaller windows of fixed size or on window adjustment heuristics that allow far smaller window sizes and usually perform better than fixed and/or larger windows (see e.g. [55,34,27]). 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.

Drifting concepts can also be learned effectively and efficiently with little parameterization by an error minimization framework for adaptive time windows [26] and example weighting or selection [28,25]. This framework makes use of support vector machines (SVMs) and their special properties, which allow an efficient and reliable error estimation after a single training run [20].

The methods of the framework either maintain an adaptive time window on the training data [26], select representative training examples, or weight the training examples [28,25]. 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. These approaches do not require complicated parameterization and are both, theoretically well-founded as well as effective and efficient in practice. They are described in more detail in Sec. 2.3 and Sec. 2.4 and used in the experiment reported in this paper for comparative purposes. Further related work on concept drift, i.e. approaches using ensemble learners, are described in Sec. 3.1.

### 2.3 Adaptive Time Windows

The *adaptive time window* approach [26,25] used in the experiments reported in this paper as state-of-the-art method for comparisons automatically adjusts the window size to the current extent of concept drift by selecting the window size that minimizes the expected classification error on new examples. This approach follows the idea shown in Fig. 1 in Sec. 2.1. A sliding time window consisting of the last $h$ batches is used to train a classifier for newly arriving unclassified examples. For each point in time (new batch of examples) $t + 1$, the best window size $h$ is determined and a new classifier is learned from the examples in that window. Among all candidate time window sizes $h = 1 \ldots t$, the one minimizing the expected error is chosen by training a classifier for each candidate window size and estimating the error rate of the resulting classifiers on most recently received labeled data, i.e. the examples of the last batch $t$, which is assumed to most closely resemble the current true concept.

For a simple concept drift scenario with a single abrupt concept shift (at batch $t_1$), Fig. 2 illustrates the selected best expected window size at each point in time (current batch) $t$. First, as long as the concept remains stable (from batch $t_0$ to batch $t_1$), the time window grows. When the drift occurs at batch $t_1$, the old data no longer representative for the new concept is dropped and the window size reduced to one batch only. Afterwards the target concept remains stable again (after $t_1$) and hence the time window keeps on growing. Since the window adjustments are only based on the error estimation and minimization, no complicated or domain-dependent parameter tuning is necessary.

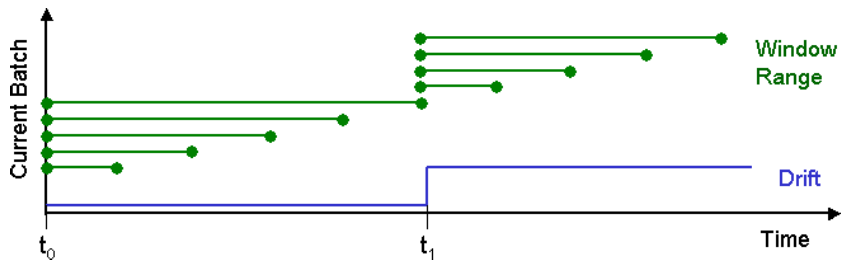


**Fig. 2.** Adaptive Time Window: The sliding time window grows in phases with a stable target concept (batches $t_0$ to $t_1$ and after $t_1$) and is cut off at an abrupt concept shift (at batch $t_1$).

### 2.4 Batch Selection

While the previously described adaptive time window approach can only select a set of connected, i.e. consecutive batches as training set at each point in time, the *batch selection* strategy [28,25] selects all batches sufficiently matching the most recent labeled batch individually and hence is more flexible. At each point in time (batch) $t + 1$, this second state-of-the-art method for handling concept drift used for comparisons in the experiment section of this paper first trains a classifier on batch $t$ only, i.e. on the newest labeled data available, and applies this classifier to the data in each previous batch $1 \ldots t - 1$. Each batch, where this classifier produces an error $\delta$ times as high as or higher than that produced by this classifier on its training batch $t$, is discarded from the training set, because its data is considered to be not close enough to the current target concept. All other batches together with the batch $t$ form the final training set, based on which the final classifier for the new batch $t + 1$ is learned.

Fig. 3 shows the batches selected for training the final classifier for each point in time (current batch) for a drift scenario, where the target concept starts as one concept $c_1$, then abruptly shifts to another concept $c_2$ at batch $t_1$, and later at batch $t_2$ abruptly shifts back to concept $c_1$ again. The illustration demonstrates that the batch selection strategy selects and discards batches not fitting to the current target concept individually and can for example re-

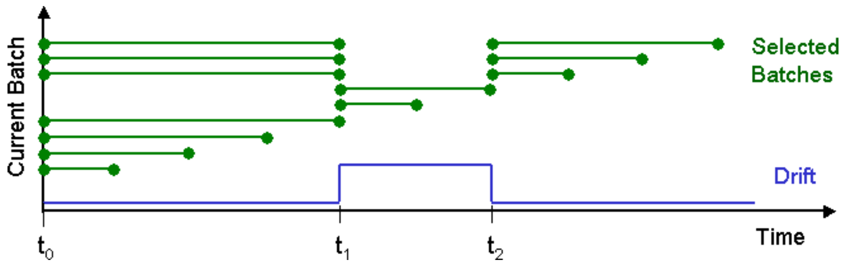


**Fig. 3.** Batch Selection: All batches fitting to the current target concept can be selected individually.

use old data from the time before the concept change in $t_1$ after the second change in $t_2$, where the current target concept is the same as before the first change.

Like the adaptive time window approach, the batch selection strategy is based on error minimization only and not on domain-dependent heuristics. For all experiments reported here and in previous work, the factor $\delta$ for the acceptable error range was simply set to 2, and no optimization of this parameter was performed on any of the data sets used.

For the results for the adaptive time window and batch selection approaches reported in this paper, SVMs were used as base learner in these frameworks, because of their good performance and the availability of efficient error estimators [20,26]. But since the performance of any classification learner can be estimated by e.g. cross-validation or leave-one-out error estimation, these frameworks are not restricted to the use of SVMs as base learner, but can be applied with any other classification learner.

## 3 Adapting Ensemble Methods to Drifting Streams

This section presents a novel ensemble method for data streams, which has some appealing properties in the presence of concept drift. Subsection 3.1 motivates the use of ensembles methods for streaming data, and it briefly reviews existing approaches. The idea behind the ensemble method presented here is sketched in subsection 3.2. In 3.3 the ensemble algorithm is adapted to a streaming scenario with concept drift.

### 3.1 Ensemble Methods for Data Stream Mining

In the recent years many algorithms specifically tailored towards mining from data streams have been proposed. The goals of the algorithms vary, depending on the assumed scenarios. Apart from being able to cope with concept drift, scalability is one of the most important issues. For very large datasets, the induction of classifiers like decision trees is very efficiently possible: Using a sampling strategy based on Hoeffding bounds, the VFDT algorithm efficiently induces a decision tree in constant time [10]. An extended version of this algorithm updates the tree based on a time window of fixed length, which allows to compensate concept drift up to a certain degree [19]. By combining several trees to an ensemble of classifiers, techniques like bagging [7] and boosting [13] have been shown to significantly improve predictions for many datasets. For some of these ensemble algorithms corresponding online variants for data streams have been suggested, see e.g. [39,35].

The SEA algorithm [49] induces an ensemble of decision trees from data streams and explicitly addresses concept drift. It splits the data into batches and fits one decision tree per batch. To predict a label, the base models are combined by an unweighted majority-vote, similar to bagging. As soon as the number of base models exceeds a user specified constant, models are discarded using a heuristic approach. The authors do not report an increase in classification performance compared to a single decision tree learner, but state that the ensemble recovers from concept drifts. The recovery time of this approach seems unnecessarily long, as it only exchanges one model in each iteration and uses no confidence weights.

Another interesting approach [11] is based on unweighted base learners similar to Random Forests [8]. It exploits example selection to include only useful older data into the training set. Examples from earlier batches are only included if predicted correctly by both, the latest model and a recent (assumed optimal) one. Cross-validation experiments may still cause the learner to discard all old examples and to rely on a new model that has been learned from scratch. This allows to adapt more quickly to sudden drift than possible with SEA. However, as the author points out, it is a heuristic selection strategy. Further disadvantages are the required assumption of a fixed marginal distribution, that is a fixed probability to observe a specific instance, regardless of its label, and high computational costs if a different base learner is used.

A recent theoretical analysis suggests, that *weighted* base learners are a preferable alternative in domains with concept drift [29]. The analyzed AddExp algorithm steadily updates the weights of *experts* (base models in an ensemble), and adds a new expert each time the ensemble misclassifies an example. The new experts start to learn from scratch, using a weight that reflects the loss suffered by the ensemble for the current example. All experts are continuously trained on all new examples. The main results reported for AddExp are more of theoretical interest, because the original version requires to maintain an unreasonably large number of experts, and more efficient variants rely on heuristics.

More practically oriented work that addresses learning from data streams in a similar fashion is described in [48] and [54]. The former of these approaches trains a weighted *fixed-size committee* of incremental decision trees, all of which are updated whenever a new example arrives. At each point in time the performance of all base models is estimated based on a *window of fixed size*. Poorly performing models are discarded and replaced by a new decision tree, trained from all subsequently read examples. A disadvantage of this fixed number of base classifier approach is that it does not induce ensembles of *diverse* base models, like boosting algorithms do by appropriately re-weighting examples. The approach rather leads to redundant ensembles, because the examples are not weighted individually for different base learners, and the most recent parts of the training sets are identical. Updating all incremental decision trees simultaneously may turn out to be expensive during concept drifts. It still cannot be expected to outperform adapting time window approaches, as it implicitly maintains heuristically derived weights for the most recent examples.

The approach presented in [54] reads training data in batches, and it trains one classifier per batch. Even during stationary phases without drift, no classifier is ever trained from more than a single batch. Consequently, large batch sizes are required. The performance of each classifier is estimated based on just the most recent classified batch in each iteration, and the inverse (estimated) error rate is used to weight the model. The authors prove that – given that the estimates are precise – this weighting scheme outperforms a single classifier trained from all the data during concept drift. This result is not surprising, especially if compared to boosting, where an improvement in accuracy is expected for each additional base model, given that exact estimates are provided. However, the presented approach does not weight examples as done by boosting procedures. As a consequence, introducing diversity into ensembles is possible only by applying heuristics during a pruning procedure, an aspect that is similar to SEA.

Sec. 3.3 extends the efficient boosting procedure presented in Sec. 3.2 to streams. It trades off predictive performance versus scalability. To this end, the online algorithm reads examples aggregated to batches and decides for each batch, whether to add a new expert to the ensemble or not. Unlike in SEA and similar algorithms, the base models of the ensemble are combined by a weighted majority vote. Subsequent models are trained after re-weighting the examples of the new batch, and each new base classifier model is assigned a weight that depends on both, its own performance and the performance of the remaining ensemble. Adaptation to concept drift works by continuously re-estimating the weights of all ensemble members, similar to the procedure presented in [54], but with each weight fit to the residuals of the (already) weighted ensemble of previous base models. This last aspect is very similar to logistic regression, with the predictions of the base models acting as constructed features.

## 3.2 Ensemble Generation by Knowledge-Based Sampling

As a motivation for the subsequently presented knowledge-based sampling (or example weighting) technique, Fig. 4 illustrates the main idea for a simplified concept drift scenario. The underlying assumption is, that during a concept drift all examples are sampled from a mixture distribution, which can be thought of as a weighted combination of two pure distributions characterizing the target concepts before and after the drift. In the figure the initial target concept is simply referred to as *Concept 1*. Examples are sampled from a corresponding stationary distribution up to the first dotted vertical line. A learning algorithm may simply induce a model from all the data, which will predict *Concept 1*. Now, as the drift starts and a *Concept 2* overlaps *Concept 1*, this model will show a decreasing accuracy. Please note, that in the intermediate batches the label can best be described as a probabilistic combination of different concepts. Even if for the pure concepts the label depends on the features deterministically, the perfect model in between the two concepts can only be derived in terms of Bayes' optimal decision rule. For simplicity we assume diametral concepts in the figure, and a correctly trained model for *Concept 1* when the drift starts. In this situation there is a point in time (a batch), up to which the Bayes' optimal classifier predicts *Concept 1*. This can be concluded from the assumed generative model, because whenever there is a conflict between both concepts, then *Concept 1* is more likely to be correct. The point up to which knowing about *Concept 2* is useless for making predictions is shown as a thick dotted vertical line in Fig. 4. *After* that point it is optimal to purely predict *Concept 2*. However, it just is not clear how to induce an appropriate model without seeing a few batches sampled from a pure corresponding distribution. Such batches are available to the learner only after the last dotted vertical line, but between the second and last line we would already like to predict *Concept 2*.

One of the main motivations for the boosting-like algorithm presented in this work is that, given an accurate model for *Con-*

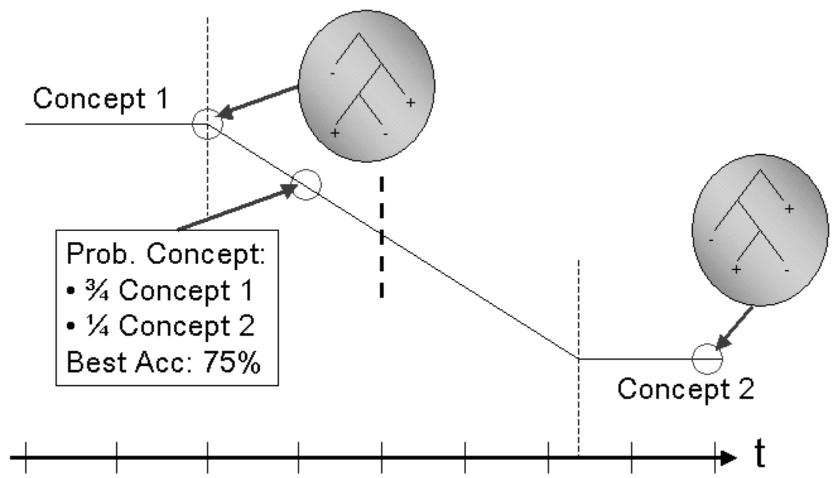


**Fig. 4.** Continuous concept drift, starting with a pure *Concept 1* and ending with a pure *Concept 2*. In between, the target distribution is a probabilistic mixture. It is optimal to predict *Concept 1* before the dotted line, and *Concept 2*, afterwards.

*cept 1*, it allows to *decompose* the mixture distribution during the concept drift. Thus, it is possible to construct a sample with respect to *Concept 2* as soon as the drift starts (first dotted line). This look-ahead strategy is inherently different from all approaches discussed in Sec. 3.1, and it allows to adapt to drift very quickly. The main reason is that it exploits more of the information encoded in the stream. Please note, that even between the thick and the final dotted lines the model for *Concept 1* is not useless, because it still helps to "purify" subsequent batches by subtracting the deprecated *Concept 1*. The resulting model is a probabilistic ensemble classifier, for which Bayes' optimal decision rule can explicitly be applied when crisp predictions are required.

The algorithm introduced in this paper is based on the sampling strategy suggested in [45]. Patterns are discovered iteratively. A pattern that is found in one iteration extends the user's prior knowledge in the next one. In each iteration the sampling procedure produces training sets that are "orthogonal" to the combined probability estimate corresponding to the prior knowledge. This aspect is very close to boosting classifiers. The idea of removing prior knowledge by biased sampling is formulated in terms of constraints. Formally, this step defines a new distribution, as close to the original function as possible, but orthogonal to the estimates produced by available prior knowledge. Technically this step can be realized by introducing example weights. As a result of switching distributions, the evaluation metrics for model candidates – when applied to these kinds of training sets – are "blinded" regarding the parts of the data that can be concluded from prior knowledge. All that is accounted for in the next iteration is the unexpected component of each model. In the scope of this paper the only kind of "prior" knowledge are the base models yielded by preceding iterations.

For a given instance space $\mathcal{X}$ and nominal class attribute $\mathcal{Y}$ examples are expected to be sampled i.i.d. from an initial distribution $D : \mathcal{X}, \mathcal{Y} \rightarrow \mathbb{R}^+$. Let $h : \mathcal{X} \rightarrow \mathcal{Y}$ denote a base model from hypothesis space $\mathcal{H}$, predicting a class.

As a first constraint, the new distribution $D'$ to be constructed should no longer support the knowledge encoded in the hypothesis $h$. This means that, with respect to $D'$, the observation of any fixed label $y^*$ should be independent of all possible predictions $h(x) = y'$:

$$(\forall y^*, y' \in \mathcal{Y}) : P_{x,y \sim D'} \left[ y = y^* \mid h(x) = y' \right] = P_{x,y \sim D'} \left[ y = y^* \right] \quad (1)$$

If eq. (1) did not hold, then the same hypothesis would allow to derive further information about the true label's conditional distribution given the prediction of $h$.

As a second constraint, the probability of observing a specific class $y^* \in \mathcal{Y}$ and the probability of a specific prediction $y' \in \mathcal{Y}$ should not change from $D$ to $D'$; it is sufficient and possible to remove only the *correlation* between the true label and the predicted label:

$$(\forall y^* \in \mathcal{Y}) : P_{x,y \sim D'} \left[ y = y^* \right] = P_{x,y \sim D} \left[ y = y^* \right] \quad (2)$$

$$(\forall y' \in \mathcal{Y}) : P_{x,y \sim D'} \left[ h(x) = y' \right] = P_{x,y \sim D} \left[ h(x) = y' \right] \quad (3)$$

Eq. (2) ensures that the class skew does not change, which would result in an implicitly altered cost model for misclassifying examples [14]. Eq. (3) avoids to skew the marginal distribution unnecessarily.

Finally, within each partition sharing the same predicted label $y'$ and true class $y^*$, the new distribution is defined proportionally to the initial one; having a hypothesis $h$ as prior knowledge, all instances within such a partition are indistinguishable. Changes to the conditional probabilities within one partition prefer some instance over the others, despite their equivalence with respect to the available prior knowledge. This translates into the following constraint:

$$(\forall x, x' \in \mathcal{X})(\forall y^*, y' \in \mathcal{Y}) : P_{x,y\sim D'}(x = x' \mid h(x) = y', y = y^*) \\ = P_{x,y\sim D}(x = x' \mid h(x) = y', y = y^*) \quad (4)$$

Constraints (1)-(4) induce a unique target distribution. The following definition eases notation.

**Definition 1.** The Lift for a given hypothesis $h : \mathcal{X} \to \mathcal{Y}$, predicted class $y' \in \mathcal{Y}$, and true class label $y^* \in \mathcal{Y}$ is defined as

$$\textsc{Lift}_D^{(h)}(y' \to y^*) := \frac{P_{x,y\sim D}\left[h(x) = y', y = y^*\right]}{P_{x,y\sim D}\left[h(x) = y'\right] \cdot P_{x,y\sim D}\left[y = y^*\right]}$$

Similar to precision, Lift measures the correlation between a specific prediction and a specified true label. A value larger than 1 indicates a positive correlation.

**Theorem 1.** *For any initial distribution $D$ and hypothesis $h \in \mathcal{H}$, constraints (1)-(4) are equivalent to*

$$\forall \langle x, y\rangle \in \mathcal{X} \times \mathcal{Y} : P_{D'}(x, y) = P_D(x, y) \cdot \left(\textsc{Lift}_D^{(h)}(h(x) \to y)\right)^{-1}.$$

A proof is given in [45]. Theorem 1 defines a new distribution to sample from, given a hypothesis $h$ as prior knowledge. Assuming a single hypothesis is not very restrictive, since it is possible to directly incorporate each new base model into a single ensemble classifier. In a classical learning scenario without concept drift, the

1. Initialize $D_1$ with the uniform distribution over the example set $\mathcal{E} = \langle x_1, y_1 \rangle, \ldots, \langle x_m, y_m \rangle$.
2. For $i = 1$ to $n$ do // n: user given number of iterations
   (a) Call BaseLearner($D_i$, $\mathcal{E}$) to find an accurate model $h_i : \mathcal{X} \to \mathcal{Y}$.
   (b) Compute $\textsc{Lift}_{D_i}^{(h_i)}(y' \to y^*)$ applying definition 1.
   (c) $D_{i+1}(x_j, y_j) \leftarrow D_i(x_j, y_j) \cdot \left(\textsc{Lift}_{D_i}^{(h_i)}(h(x_j) \to y_j)\right)^{-1}$ for all $\langle x_j, y_j \rangle \in \mathcal{E}$.
3. Output $\{h_1, \ldots, h_n\}$ and the Lift values. Predict $P(y \mid x)$ with eq. (5).

**Fig. 5.** Algorithm KBS (Knowledge-Based Sampling)

theorem can directly be applied iteratively [46,44]: Base model $h_i$ is selected based on distribution $D_i$. Distribution $D_{i+1}$ is defined by applying theorem 1 to $D_i$ and $h_i$. A corresponding knowledge-based sampling algorithm (KBS) which is not tailored towards learning from streams is depicted in Fig. 5. It boosts weak base learners and has empirically been shown to be competitive to AdaBoost and LogitBoost [44].

The inverse of the re-weighting strategy allows to approximately reconstruct the original distribution $D_1$ as a combination of the single hypotheses. The following formula estimates the odds $\beta(x)$ based on $\{h_1, \ldots, h_n\}$, a sequence of $n$ hypotheses, each of which is the result of a separate iteration of learning:

$$\beta(x) := \frac{P(y = +1)}{P(y = -1)} \cdot \prod_{i=1}^{n} \frac{\textsc{Lift}_{D_i}^{(h_i)}(h_i(x) \;\to\; y = +1)}{\textsc{Lift}_{D_i}^{(h_i)}(h_i(x) \;\to\; y = -1)} \tag{5}$$

This allows to compute estimates of the conditional probabilities as

$$P(y = +1 \mid x) \approx \frac{\beta(x)}{1 + \beta(x)}. \tag{6}$$

The re-weighting scheme used by KBS makes base classifiers rank models according to their contribution to the overall accuracy: After KBS "samples out" a model $h$, the accuracy of all overlapping (correlated) models with respect to the new distribution is reduced according to the degree of overlap. Because of constraint (1), the Lift of the subset with a common prediction

$h(x)$ is 1. For a given model $h : \mathcal{X} \to \mathcal{Y}$ predictive accuracy (ACC) can be rewritten as

$$\textsc{Acc}(h) = \sum_{y^* \in \mathcal{Y}} P(h(x) = y^*) P(y = y^*) \textsc{Lift}^{(h)}(h(x) \to y^*), \quad (7)$$

which is a linear combination of the corresponding LIFT values of the covered subsets. Using examples that were re-weighted with respect to a given model $h$, the base classifier favors models according to their independent contributions [14,46]. Similar to other boosting approaches, the example weights anticipate the expectation given the previously trained models. As motivated before, this is especially useful for handling smooth concept drifts. While sudden drifts require a quick detection and a way to rapidly adjust the working hypothesis, for smooth drifts it is better to collect information on the new target concept over a period of time. Especially if the preceding concept has been identified accurately at the point in time when a drift starts, removing the knowledge about the current concept from the data allows to decompose mixture distributions as required.

### 3.3 A KBS-strategy to learn drifting concepts from data streams

The original KBS-algorithm (Fig. 5) assumes that the complete training set is available in main memory. The first step to adopt to data streams is to read and classify examples iteratively. For subsequent learning steps the re-weighting strategy of KBS allows to compute example weights very efficiently. The data is assumed to arrive in batches, each large enough to train an initial version of a base classifier.

The sizes of the training sets that are effectively used for each model are determined dynamically by the algorithm. Processing a new batch yields two ensemble variants. The first variant appends the current batch to the cache used for training in the last iteration, and it refines the latest base model accordingly. The second variant adds a new model, which is trained using the latest batch, only. Only the ensemble variant performing better on the next batch is kept.

Initialize empty ensemble $\mathbf{M} := \emptyset$.
While not end of stream, do

1. Read next batch $\mathcal{E}_k$ in iteration $k$.
2. Predict $P(y \mid x)$ for all $x \in \mathcal{E}_k$ with current ensemble $\mathbf{M}$ (eq. (5)).
3. Read true labels of $\mathcal{E}_k$.
4. If alternative ensemble $\mathbf{M}^*$ exists:
   (a) Compare accuracy of $\mathbf{M}$ and $\mathbf{M}^*$ wrt. $\mathcal{E}_k$.
   (b) $\mathbf{M} \leftarrow$ better ensemble, discard worse ensemble.
   (c) If $\mathbf{M}^*$ is discarded: $\mathcal{E}^* \leftarrow \mathcal{E}_{k-1}$ (shrink cache to one batch).
5. Initialize $D_1$: Uniform distribution over $\mathcal{E}_k$.
6. For $i \in \{1, \ldots, |\mathbf{M}|\}$, do:
   (a) Apply $h_i$ to make predictions for $\mathcal{E}_k$.
   (b) Recompute $\textsc{Lift}_{D_i}^{(h_i)}(y' \to y^*)$ based on $\mathcal{E}_k$ (Def. 1).
   (c) Update the $\textsc{Lifts}$ of $h_i$ in $\mathbf{M}$.
   (d) $D_{i+1}(x_j, y_j) \leftarrow D_i(x_j, y_j) \cdot \left(\textsc{Lift}_{D_i}^{(h_i)}(h(x_j) \to y_j)\right)^{-1}$ for all $\langle x_j, y_j \rangle \in \mathcal{E}_k$.
7. Call $\textsc{BaseLearner}(D_{|M|+1}, \mathcal{E}_k)$, get new model $h_{|M|+1} : \mathcal{X} \to \mathcal{Y}$.
8. Compute $\textsc{Lift}_{D_{|M|+1}}^{(h_{|M|+1})}(y' \to y^*)$ (Def. 1).
9. Add model $h_{|M|+1}$ (and its $\textsc{Lifts}$) to ensemble $\mathbf{M}$.
10. If this is the first batch, then $\mathcal{E}^* = \mathcal{E}_k$ (no alternative ensemble).
11. Else
    (a) $\mathcal{E}^* \leftarrow \mathcal{E}^* \cup \mathcal{E}_k$, (extend cache by last batch)
    (b) $\mathbf{M}^* \leftarrow \text{clone}(\mathbf{M})$
    (c) discard last base model of $\mathbf{M}^*$
    (d) repeat steps 5-9 for $\mathcal{E}^*$ and $\mathbf{M}^*$ instead of for $\mathcal{E}_k$ and $\mathbf{M}$

**Fig. 6.** Algorithm KBS-Stream

The strategy serves two purposes. First, for stationary distributions a new model is trained only, if there is empirical evidence that this increases the accuracy of the resulting ensemble. This will generally happen if the learning curve of the latest model has leveled off, see e.g. [22], and the data set is well suited for boosting. Second, if sudden concept drift (concept shift) occurs, the same estimation procedure instantly suggests to add a new model, which will help to overcome the drift.

The second step of adopting KBS to data streams is to foresee a re-computation phase in which base model performances are updated with respect to the current distribution. In fact, we believe that this is a main advantage of using weighted ensembles in a concept drift scenario. For stationary distributions the weights vary marginally, while for smoothly drifting scenarios they are systematically shifted and even allow to quantify and interpret the drift in terms of previously found patterns or models. This will be discussed in Sec. 3.4. Even sudden drifts do not pose a problem, as they automatically result in radically reduced weights of previously trained models, and in high weights of subsequently trained models, if these parameters are re-estimated from new data. The response time to drifts is very short. Since the streaming variant of KBS is closely coupled to the accurate KBS boosting algorithm, the predictive performance is expected to outperform those of single base models for many datasets. Pruning of ensembles can efficiently be addressed during weight re-computation; whenever a model does not reach a fixed minimum advantage over random guessing on the latest batch, it is discarded. This is a natural and common pruning strategy for boosting algorithms, e.g. also found in the WEKA implementation [56] of ADABOOST [13].

The algorithm is depicted in Fig. 6. It loops until the stream ends. Lines 1-2 apply the current ensemble to the new batch without knowing the correct labels. Lines 3-4 check whether continuing the training of the latest model with the latest batch outperforms adding a new model trained on that batch[3]. Only the better of these two ensembles is kept. Lines 5 and 6 recompute the LIFT parameters of all base models. To this end, the models are iteratively applied to the new batch, and the weights are adjusted similarly to the learning phase. Finally, lines 8-11 train two variants of the ensemble again, $\mathbf{M}^*$ being the one extending the cache and updating the newest model appropriately, and $\mathbf{M}$ being the one that adds a new model, which is trained using only the newest data.

One degree of freedom is left in line 2: The algorithm may use $\mathbf{M}$ or $\mathbf{M}^*$ to classify the new batch, as the performance of both is unknown at that time. For the experiments two variants have been implemented. The first one (KBS$_{stream}$) always uses ensemble $\mathbf{M}^*$, since models trained from larger batches are generally more reliable. The second variant (KBS$_{hold_out}$) uses a hold out set of 30% from the last batch to decide which ensemble to

[3] The pseudocode does not assume an incremental base learner, but trains new models on cached data. For incremental base learners no cache is required.

use. Alternatively, one could perform more reliable (but also more costly) cross-validation experiments, or apply the $\xi\alpha$-estimator for support vector machines [26], which is as efficient as the validation approach suggested in [11]. However, in our experiments we found errors due to the hold-out estimation negligible[4] compared to the systematic one-batch-delay between the distributions for training and application. For this reason we currently do not further address this kind of validation.

If incremental base learners are used, then only the latest batch needs to be stored. The runtime is dominated by adjusting the most recent model to this data, and by applying all base models to it. This avoids the combinatorial explosion and memory requirements of advanced time windowing and batch selection techniques, respectively (see Sec. 4.1). Incremental variants exist for many popular learning algorithms, in particular for decision trees [52] and support vector machines [42].

### 3.4 Quantifying Concept Drift

An appropriate combination of several base classifiers often allows to increase predictive accuracy over that achieved by an average single classifier. As a disadvantage, the results lose interpretability to a certain extent. In principle a similar argument also applies in the context of concept drift, but it is interesting to see, that the proposed technique allows to extract a *different* kind of information in this setting: It allows to track the kind of drift underlying the data stream by analyzing the weights of individual base learners.

Please recall, that unlike methods that continuously retrain all models [48,29], the KBS algorithm "freezes" all models but the latest one. The weights of all frozen models are re-estimated continuously, by applying them in chronological order to the current batch, weighting examples accordingly, and by estimating the Lift values of all models based on these weighted examples. This is exactly the same procedure as during the training phase,

[4] There is one exception that will be reported in Sec. 4.4. It is an extreme case in which the batch size has been chosen much too small for the algorithm.

so any significant deviation from the initial model weights indicates a corresponding change of the underlying distribution. An advantage of using the LIFTs as performance estimates is that the corresponding weight vectors have a clear semantics at different points in time, and are thus comparable without any additional artificial normalization. Only the weight of the latest model is not yet comparable between different iterations, because it is still continuously refined by the boosting procedure. As model weights are re-estimated in chronological order, this has no effects on the remaining ensemble.

The idea of drift quantification is illustrated for the scenario sketched in Fig. 4, which was discussed in Sec. 3.2 as a motivation for the knowledge-based sampling approach to overcome concept drift. For simplicity we assume a base learner like a support vector machine, that continuously improves with additional training data and does not benefit much from boosting. During the stationary distribution before the drift, the KBS algorithm fits a single model to the training data. Let this model capture the deterministic relation between features and label well. It may perform differently well when predicting a positive or a negative label, but the LIFTs will vary just marginally from batch to batch, since the underlying distribution does not change. When the drift starts, a significant change in the distribution makes the model perform worse, and the LIFT ratios (positive to negative) slowly approach 1. More interestingly, the re-weighted batch suddenly allows to fit a separate model, which, together with the first model, has a higher estimated accuracy. The first model is frozen, but its estimates are continuously updated. The drift takes several batches, so the new model will first reflect minor effects on the data. The small LIFTs of the second model – estimated on the re-weighted batch – cause an unbalanced ensemble of these two models, with the first one having a much higher overall impact. The second model is now refined throughout the drift and reflects an increasingly important aspect of the data, while the first model loses accuracy from batch to batch. Consequently, the LIFT ratios and hence the importance of the first model are decreased by the KBS algorithm, in favor of the second one. If the first model

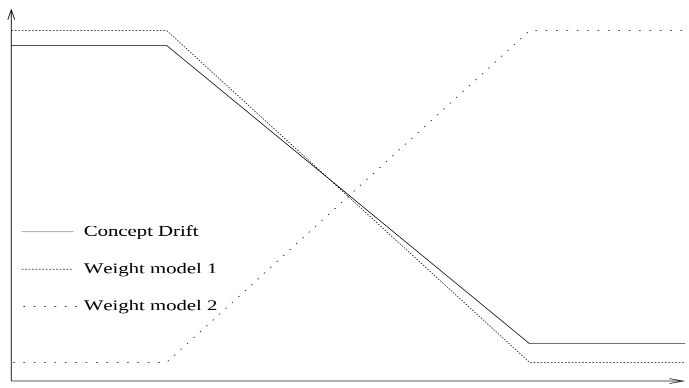


**Fig. 7.** An idealistic change of model weights over time. The solid line depicts a drift from an initial to a new target concept. The dotted lines show how the base learner weights reflect the presence of their represented target concepts.

has become useless after the drift, it has no significant advantage over random guessing (LIFT $\approx 1$), so it is discarded automatically. This is not always the case, because different target concepts often overlap.

Fig. 7 depicts an ideal change of ensemble weights over time for the described scenario. Until the drift starts, the first base model is the only one, assumed to be accurate, and it consequently receives a high weight. This weight is continuously changed based on estimates of the current accuracy. As this weight decreases, the importance and weight of model two increases. In this ideal situation the accuracy is reflected by the maximum of the two dotted lines, which is optimal with respect to Bayes' rule. Sec. 4.4 reports corresponding results of experiments with real-world data.

# 4 Experiments

## 4.1 Experimental Setup and Evaluation Scheme

In order to evaluate the KBS learning approach for drifting concepts, it is compared to the adaptive time window approach, to the batch selection strategy, and to three simple non-adaptive data management approaches.

**Full Memory:** The learner generates its classification model from all previously seen examples, i.e. it cannot "forget" old examples.

**No Memory:** The learner 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 a fixed size of $n = 3$ batches is used on the training data.

**Adaptive Window:** A window adjustment algorithm adapts the window size to the current concept drift situation (see Sec. 2.3 and [26,25]).

**Batch Selection:** 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, are selected for the final training set. All other examples are deselected (see Sec. 2.4 and [28,25]).

The performance of the classifiers is measured by prediction error. All results reported in Sec. 4.2 and 4.3 using three simulated concept drift scenarios on real-world data are averaged over four runs, each based on a different random ordering of the examples into a stream. The results reported in Sec. 4.4 are from a single run only, because the examples are taken in their real order and no artificial concept drift is simulated or imposed, but there is a real concept drift inheret to this real-world data set. The experiments were conducted with the machine learning environment YALE [12,40], the SVM implementation *mySVM* [41], and two learners from the WEKA toolbox [56], namely a support vector machine (SMO-SVM) and a decision tree learner (J48), as well as the meta-learner ADABOOST provided by WEKA.

### 4.2 Evaluation on Simulated Concept Drifts with TREC Data

The first set of experiments is performed in an information filtering domain, a typical application area for machine learning methods able to handle drifting concepts [23,27,26]. Text documents are represented as attribute-value vectors (*bag of words* model),

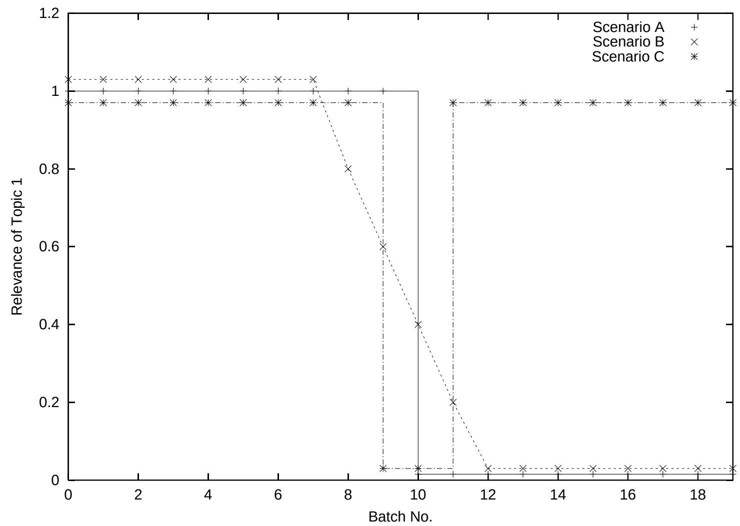


**Fig. 8.** Relevance of the first topic/concept over time in the concept change scenarios A, B, and C, respectively. The relevance of the second relevant topic/concept is *1.0 - relevance of topic 1.*

---

where each distinct word corresponds to a feature whose value is the "ltc"-TF/IDF-weight [43] of that word in that document. The experiments use a subset of 2608 documents of the data set of the *Text REtrieval Conference (TREC)*. Each of the real-world business news texts is assigned to one or several categories, five of which are considered here.

Three concept change scenarios are simulated following the experimental set-up in [26,28,25]. The texts are randomly ordered into a stream and split into 20 batches of equal size containing 130 documents each. In all scenarios, a document is considered relevant at a certain point in time, if it matches the interest of the simulated user at that time. The user interest changes between two of the topics, while documents of the remaining three topics are never relevant. Fig. 8 shows the probability of being relevant for a document of the first category at each batch for each of the three scenarios; this also implies the probability of the second (sometimes) relevant topic. *Scenario A* is an abrupt concept shift from the first to the second topic in batch 10. In *Scenario B* the

| | Full Memory | No Memory | Fixed Size | Adaptive Size | Batch Selection | KBS stream | KBS hold_out |
|---|---|---|---|---|---|---|---|
| Scen. A | 21.11% | 11.16% | 9.03% | 6.65% | 6.15% | 6.89% | **5.88%** |
| Scen. B | 21.30% | 12.64% | 9.76% | 9.06% | 9.33% | **8.64%** | 9.50% |
| Scen. C | 8.60% | 12.73% | 11.19% | 8.56% | **7.55%** | 10.11% | 8.27% |

**Table 1.** Error of all time window and example selection methods vs. KBS.

user interest changes slowly from batch 8 to batch 12. *Scenario C* simulates an abrupt concept shift in the user interest from the first to the second topic in batch 9 and back to the first in batch 11. Tab. 1 compares the results of all static and adaptive time window and batch selection approaches on all scenarios in terms of prediction error [26,28,25] to the two variants of KBS. The results are averaged over four runs with different random orderings of the examples. In all cases the learning algorithm was a support vector machine (SVM) with linear kernel.

The KBS algorithm manages well to adapt to all three kinds of concept drift. Tracking the ensembles revealed, that during stationary distributions the current model was continuously refined. After a concept shift (*scenario A*) a new model was trained, and the old model received a significantly lower weight. It was not discarded, however, since it still helped to identify the three topics which are always irrelevant. The hold out set helped to identify the better of the two ensembles reliably at classification time. In *scenario B* five or more models were trained. The ensemble accurately adopted to the drift, but at classification time the systematic one-batch-delay of the hold out estimate was sometimes misleading. In *scenario C* the full memory approach is already competitive to all but the batch selection scenario. Without the hold out set KBS applies the depreciated model one iteration too long for each concept shift. This delay increases the error rate by about 2%. This problem is circumvented by using a hold out set. In essence the KBS algorithm performed very well on this domain, and it even outperformed computationally more expensive approaches. Only in *scenario C* the batch selection method is clearly superior, probably because it is the only method able to

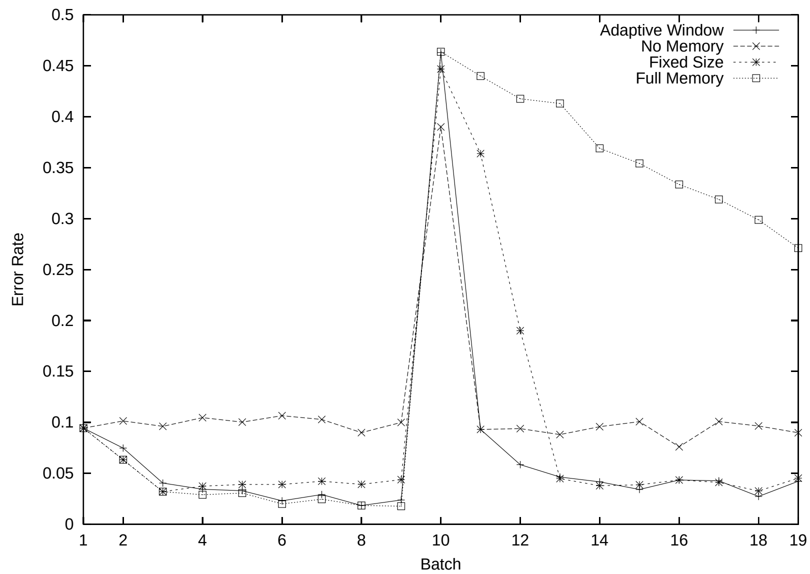


**Fig. 9.** TREC data, scenario A: Error rate over time for the non-adaptive methods versus the adaptive time window approach.

---

concatenate the data before the first and after the second concept shift to a single training set.

While Tab. 1 lists the error rates of the different learning strategies averaged over time, i.e. over all batches, and over all four repeated runs of the experiments, figures 9 to 12 show the error rates of the different learning strategies over time, i.e. at each batch, also averaged over all runs.

Fig. 9 compares the non-adaptive methods to the adaptive time window approach in concept drift scenario A. Always learning on all available labeled data ignoring any possible concept drift that may have happened (*Full Memory*) leads to good generalization and consequently low error rates as long as no concept drift occurs. But as soon as a concept drift occurs, the error rate goes up and only very slowly decreases again, because all the old data, which is no longer representative for the current target concept, still is part of the training set and hinders effective learning.

The opposite approach of not storing any old data except for the last labeled batch (*No Memory*) allows a maximally fast adap-

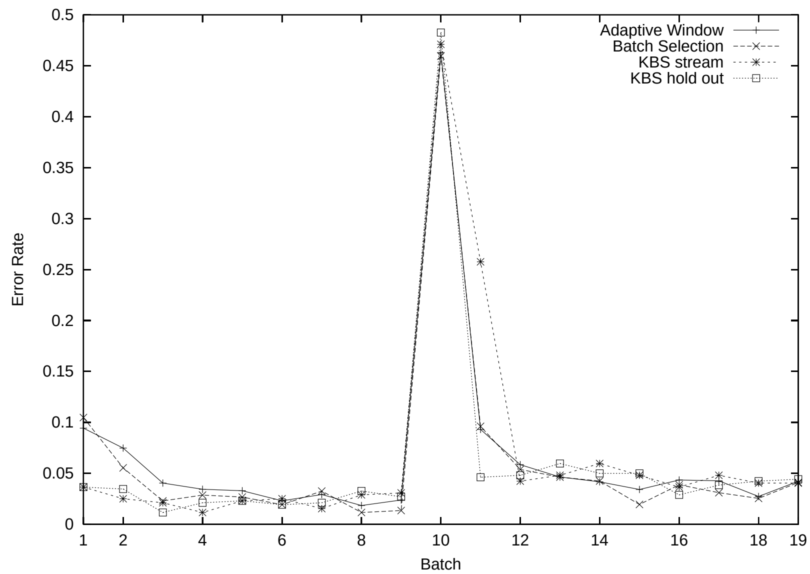


**Fig. 10.** TREC data, scenario A: Error rate over time for the adaptive time window and batch selection techniques versus the two KBS variants.

---

tation to concept drift and a correspondingly quick recovery of the error rate. However, the baseline error of this second simple strategy in phases without concept drift is comparatively high, i.e. more than twice as high as that of the other strategies, and hence the overall averaged error rate listed in Tab. 1 is also not so favorable.

Using a sliding time window of *Fixed Size* means a compromise between these two extremes, i.e. an acceptable baseline error and a better recovery speed than the full memory method, but the comparison with the two extremes shows, that the performance of such a static window approach is only a compromise and trade-off between adaptability in phases with concept drift and low error rate in stable phases and hence still leaves a lot of potential for improvements for more adaptive strategies. The described behavior of the non-adaptive methods also explains their high error rates in Tab. 1 and motivates the use of adaptive approaches to handling concept drift.

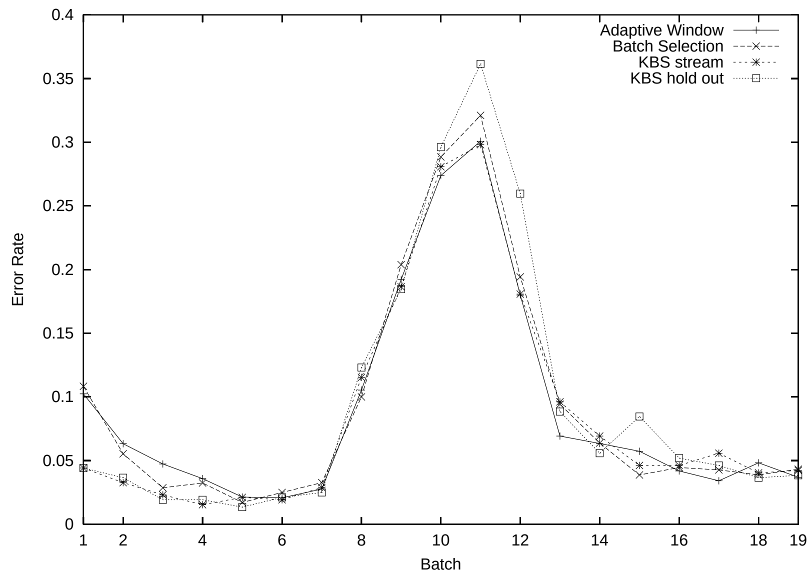


**Fig. 11.** TREC data, scenario B: Error rate over time for the adaptive time window and batch selection techniques versus the two KBS variants.

---

The *Adaptive Window* approach is able to combine the good generalization performance of the full memory method in stable phases without concept drift by keeping as much still representative data as possible with the fast adaptability of the no memory method by dropping all misleading old data immediately when the drift occurs. Hence the adaptive time window manages to combine the advantages of the two static extremes by adapting to the current extent of drift.

Fig. 10 compares the adaptive time window and batch selection strategies to the two variants of KBS in the same concept drift scenario A. Like adaptive time window and batch selection, both KBS variants achieve low baseline error rates and adapt quickly to the concept drift. Using a hold-out set allows KBS to quicker adapt to the drift and consequently a quicker recovery of the error rate.

Also in concept drift scenario B, as Fig. 11 shows, the two variants of KBS exhibit a similar behavior in terms of error rate over time to the adaptive time window and batch selection strate-

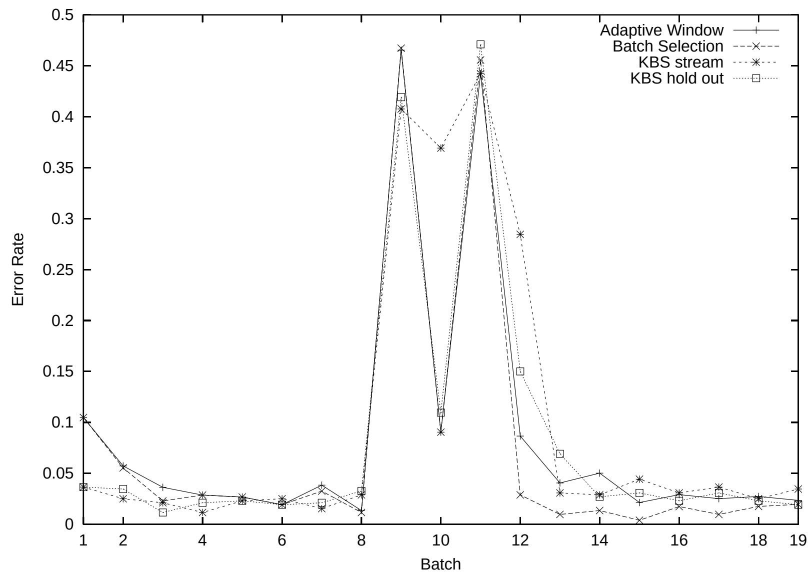


**Fig. 12.** TREC data, scenario C: Error rate over time for the adaptive time window and batch selection techniques versus the two KBS variants.

gies as far as the low base line error and the adaptability to the drift are concerned. In concept drift scenario C depicted in Fig. 12 applies the same, if KBS uses a hold-out set, but KBS does not adapt as quickly without the hold-out set. In this scenario with a re-occurring target concept, the batch selection strategy has the advantage of being able to re-use old data from before previous concept drifts and to do so quickly and thereby slightly outperforms the other approaches. However, even in this scenario and even more so in the other scenarios, KBS performs competitively well or even better as shown by the behavior over time in the plots and the overall average error rate in Tab. 1.

## 4.3 Evaluation on Simulated Drifts with Satellite Image Data

The second set of experiments used the satellite image dataset from the UCI library [6], a real-world dataset for classification. It contains no (known) drift over time, so we simulated concept drifts with the same technique as described in Sec. 4.2: The data

| | J48 Fixed Memory | J48 Full Memory | AdaBoost+J48 Full Memory | KBS stream | KBS hold_out |
|---|---|---|---|---|---|
| Scen. A | 11.06% | 21.65% | 20.51% | 9.50% | **9.43%** |
| Scen. B | 11.25% | 21.22% | 19.93% | 11.60% | **10.92%** |
| Scen. C | 12.70% | 10.83% | **9.50%** | 11.55% | 10.45% |

**Table 2.** Averaged prediction errors for satellite image dataset.

set was randomly ordered into a stream and split into 20 batches of equal size (321 examples per batch) and only two (*grey soil* and *very damp grey soil*) of the six classes were selected to be relevant. The same three drift scenarios *A*-*C* as in Sec. 4.2 were simulated, where the two selected classes corresponded to the two selected topics in the TREC experiments. The results are averaged over four runs with different random orderings of the examples.

Since decision trees are a more typical base learner for ensemble methods, we chose the J48 algorithm from the Weka toolbox as base learner for the experiments on this data set. We compared KBS to the non-adaptive *fixed size* window (of 3 batches) and *full memory* strategies. In addition to running J48 stand-alone, we also run AdaBoost on top of J48. For all runs the default settings of the learners were used. The results are listed in Tab. 2. Unlike for the experiments on the TREC data, the results of KBS could always be improved by using a hold out set. As before, *Scenario C* can well be tackled by a full memory approach, which is exploited by AdaBoost. For the other scenarios KBS is better than the fixed size window learner, and much better than the full memory approach.

### 4.4 Handling Real Drift in Economic Real-World Data: Predicting Phases in Business Cycles

The third evaluation domain is a task from economics based on real-world data exhibiting real concept drift. The quarterly data describes the West German business cycles from 1954 to 1994 [16]. Each of the 158 examples is described by 13 indicator variables. The task is to predict the current phase of the business cycle of the West German economy. In accordance to findings from Theis and

| | Full Memory | No Memory | Fixed Size | Adaptive Size | Batch Selection | KBS stream | KBS hold_out |
|---|---|---|---|---|---|---|---|
| 5 batches | 32.80% | 27.20% | 24.00% | 24.80% | 24.80% | 24.60% | **17.46%** |
| 15 batches | 28.08% | 28.77% | **20.55%** | 24.80% | 23.29% | 26.03% | 28.77% |

**Table 3.** Prediction error for business cycle data.

Weihs [51], we use two phases instead of four for the description of the business data, as described in [37].

The following experiments compare the performance of the KBS data stream algorithm to previously reported results [24], so the same number of batches (5 and 15) were used. The timely order of the examples (quarters) was preserved and no artificial concept drift was simulated. Hence the results are from a single run only and not averaged over several runs like in the previous sections, because the examples are taken in their real order and no artificial concept drift is imposed, but there is a real concept drift inheret to this real-world data set as demonstrated by previous experiments [24]. All approaches compared here use support vector machines (SVMs) as their base learners.

The results of these two evaluations are shown in Tab. 3. The column for the fixed time window approach lists the results for the fixed size that performed best. The fact that this approach performs well may be due to the cyclic nature of the domain. However, the size is generally not known in advance, and as shown in [24] using other fixed window sizes, leads to significant drops in performance. The results for 15 batches shows that KBS does not perform well, if each batch consists of less than a dozen examples. The reason is that it is not possible to get reliable probability estimates from such small data sets. The algorithm could cache older data in such cases, but it is more reasonable to choose larger batch sizes. Using just 5 batches (31 examples) already improves the situation, so KBS performs similar to the fixed size, adaptive size, and to the batch selection approach. The hold out set turns out to be surprisingly effective for larger batches. This result provides first evidence that KBS is able to adapt classifier ensembles to different kinds of concept drift found in real-world datasets.

### 4.5 Empirical drift quantification

The final experiments of this article investigate whether the claims made in Sec. 3.4 are realistic in practice. Two examples extracted during real experiments with the TREC data are presented to illustrate, how KBS base model weights allow to characterize kind and intensity of concept drifts in practice. No sophisticated methods for pruning or model evaluation during learning are applied, in order not to falsify any results. In the experiments the four LIFT values of models making boolean predictions are reduced to a single weight per model. For $\mathcal{Y} = \{-1, +1\}$ a model that estimates odds ratios as described by eq. (5) is transformed into a classifier of the form

$$\hat{y} := \operatorname{sign}\left(w_0 + \sum_{i=1}^{n} w_i h_i(x)\right),$$

with offset weight $w_0$ and model weights $w_1, \ldots, w_n \in \mathbb{R}$. Weight vectors could be re-weighted, but are not in the following figures, in order to ease the comparison of model impacts from one iteration to the next. The transformation works by (i) considering only the LIFT and prior *ratios*, which results in two weights per model and one "offset" term, (ii) transforming the corresponding Bayes optimal decision function by applying the logarithm, which results in a linear model, and (iii) centering the two weights per model by shifting offsets to the constant model-independent term $w_0$, so that a single weight suffices.

Fig. 13 exemplarily shows the weights of the base classifiers for a KBS application over time. As before, the algorithm is applied to the TREC dataset with a simulated concept shift (*scenario A*), using a support vector machine with linear kernel as the base learner. The base models are all trained over a period of time, and afterwards only their weights are adjusted. The performance of the initial model is directly estimated from the (unweighted) most recent batch. Model weights are upper-bounded artificially in the figure to ease visualization. Until the concept shift occurs in the middle of the figure, the first model is refined by extending the training set batch by batch. That way, the model reaches high confidence, which varies a bit due to estimates based on small batches (130 examples). The first batch sampled from the new distribution already decreases the weight of the initial classifier rapidly. The classifier is “frozen”, and KBS introduces a second classifier, which is now refined for several iterations. The first model still turns out to be useful, but with a negative weight, which indicates that the opposite of the initial target concept is correlated to the new target concept. The precise weights of both models vary a bit, but converge after a while. Refining the second model by further examples does no longer improve the overall accuracy, so at this point the KBS estimates freeze the second model as well, and it introduces a third one. This step allows to increase the expressiveness of the underlying model language wherever this seems promising.

The second experiment provides a realistic counterpart to the motivating example with slow concept drift (*scenario B*), which has been presented in Sec. 3.1. Fig. 14 shows how the weights of all involved base models change over time. Just one “outlier model”, which is directly removed from the ensemble by the KBS algorithm after induction, has been removed from the figure, in order not to overload the figure. The initial model reaches a high weight during the first stationary phase, which reflects highly confident predictions. The confidence decreases rapidly during the drift, and after only a few batches sampled from a new stationary target distribution, the initial model is even discarded by the learner (batch 16). Two new models are introduced during the drift, which both quickly lose weight as the first target concept diminishes. Please recall, that KBS re-weights the batches as if they were sampled from the pure target distribution of the new concept. In this sense, the early batches during the drift can be considered to have a higher noise level than later ones, which explains the decreasing weights. The learner still fits each classifier based on a couple of consecutive batches during the drift. Reaching the new stationary distribution, the weights of both intermediate models converge, because they contain a fixed amount of information about the new target concept. The final model is

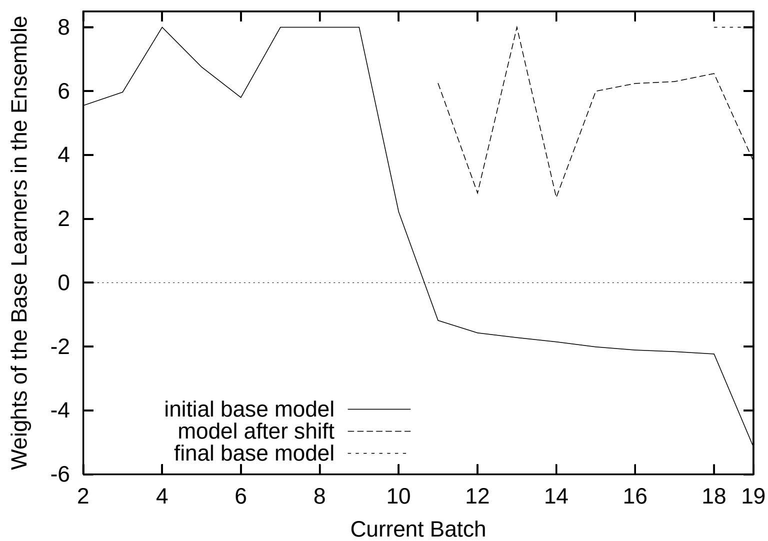


**Fig. 13.** Base model weights of a KBS ensemble for the simulated scenario A on the TREC data. Only the most recent model is refined, the others are frozen and just continuously re-weighted with respect to the latest batch. The concept shift occurs where the weight of the initial model drops drastically (batch 10).

induced after the drift ends, at a point where the previous model weights have almost converged.

Although the curves are not as simple as in the ideal case sketched earlier, the example illustrates how the weights of base learners can be used to identify the kind and degree of a concept drift underlying a data stream. We expect a higher robustness of the sketched quantification property as the batch size for estimating base model performances increases.

## 5 Conclusion and Future Work

This paper presents a new ensemble method for learning from data streams. At each iteration, base models are induced and re-weighted continuously, considering the latest batch of examples, only. Unlike other ensemble methods, the proposed strategy adapts very early and quickly to different kinds of concept drift. The algorithm has low computational costs. It has empirically

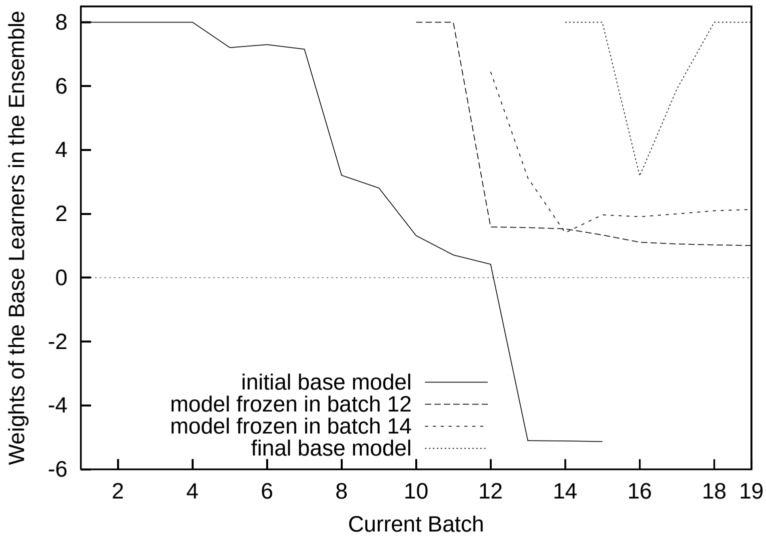


**Fig. 14.** Base model weights for another KBS ensemble trained for the simulated concept drift scenario B on the TREC data.

---

been shown to be competitive to, and often to even outperform sophisticated adaptive window and batch selection strategies. As a further advantage, it allows to track the kind and extent of concept drift.

Interesting directions for future work are evaluations of more precise and robust strategies to estimate model weights and comparisons of different pruning techniques. In addition to the presented alternatives, to either refine the latest model or to add a new one to the ensemble, other variants like learning from scratch or from up to a few batches could continuously be evaluated in parallel. This would allow to replace a complex ensemble by a single, equally well performing base model during stationary phases. Based on more precise model weights gained by cross-validation or similar techniques, the drift quantification property of KBS could be extended to predict the kind of drift to be expected in the near future. Approaches like linearly extrapolating the recently observed drift should be compared to more complex ones, like meta-learning. Finally, a comparison of all recently proposed techniques for classifier induction in the presence of concept drift on standardized simulated drift scenarios would be desirable.

## Acknowledgements

This work was supported by the Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center on the Reduction of Complexity for Multivariate Data Structures (SFB 475) at University of Dortmund in Germany.

## References

1. Jesus S. Aguilar-Ruiz and Paul R. Cohen. ACM Symposium on Applied Computing (SAC-2004), Special Track on Data Streams. In Hisham Haddad, Andrea Omicini, Roger L. Wainwright, and Lorie M. Liebrock, editors, *Proceedings of the 2004 ACM Symposium on Applied Computing (SAC), Nicosia, Cyprus, March 14-17, 2004*, 2004. http://www.informatik.uni-trier.de/∼ley/db/conf/sac/sac2004.html.
2. Jesus S. Aguilar-Ruiz and Francisco J. Ferrer-Troyano. ACM Symposium on Applied Computing (SAC-2006), Special Track on Data Streams. In *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC)*, 2006.
3. Jesus S. Aguilar-Ruiz and Joao Gama, editors. *Second International Workshop on Knowledge Discovery from Data Streams*, Porto, Portugal, October 7th 2005. In conjunction with ECML/PKDD-2005, 16th European Conference on Machine Learning (ECML) and 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD). http://www.niaad.liacc.up.pt/∼jgama/IWKDDS/.
4. James Allan. Incremental relevance feedback for information filtering. In H. P. Frei, editor, *Proc. 19th Annual ACM SIGIR Conf. on Research and Development in Information Retrieval (SIGIR '96), Zürich, Swiss, August 18-22, 1996*, pages 270–278, New York, NY, USA, 1996. ACM Press.
5. Marko Balabanovic. An adaptive web page recommendation service. In W. L. Johnson, editor, *Proc. First Int'l Conf. on Autonomous Agents*, pages 378–385, New York, NY, USA, 1997. ACM Press.
6. C.L. Blake and C.J. Merz. *UCI Repository of machine learning databases*. Dept. of Information and Computer Sciences, University of California - Irvine (UCI), Irvine, CA, USA, 1998. http://www.ics.uci.edu/∼mlearn/MLRepository.html.
7. Leo Breiman. Bagging predictors. *Machine Learning*, 13(2):30–37, 1996.
8. Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
9. William 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, 1996. AAAI Press.
10. Pedro Domingos and Geoff Hulten. Mining High Speed Data Streams. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '00)*, pages 71–80, 2000.

11. Wei Fan. Systematic Data Selection to Mine Concept-Drifting Data Streams. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '04)*, pages 128–137, Seattle, WA, USA, 2004. ACM Press.
12. Simon Fischer, Ralf Klinkenberg, Ingo Mierswa, and Oliver Ritthoff. YALE: Yet Another Learning Environment – Tutorial. Technical Report CI-136/02, Collaborative Research Center 531, University of Dortmund, Dortmund, Germany, June 2002. ISSN 1433-3325. http://yale.sf.net/ .
13. Yoav Freund and Robert R. Schapire. A decision–theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.
14. Johannes Fürnkranz and Peter Flach. ROC 'n' Rule Learning – Towards a Better Understanding of Covering Algorithms. *Machine Learning*, 58(1):39–77, 2005.
15. Joao Gama and Jesus S. Aguilar-Ruiz, editors. *First International Workshop on Knowledge Discovery in Data Streams*, Pisa, Italy, September 24th 2004. In conjunction with ECML/PKDD-2004, 15th European Conference on Machine Learning (ECML) and 8th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD). http://www.lsi.us.es/∼aguilar/ecml2004/.
16. Ullrich Heilemann and Heinz Josef Münch. Classification of West German business cycles. Technical Report 11, Collaborative Research Center on Reduction of Complexity for Multivariate Data (SFB 475), University of Dortmund, Dortmund, Germany, 1999.
17. David P. Helmbold and Philip M. Long. Tracking drifting concepts using random examples. In Leslie G. Valiant and Manfred K. Warmuth, editors, *Proceedings of the Fourth Annual Workshop on Computational Learning Theory (COLT '91)*, pages 13–23, San Mateo, CA, USA, 1991. Morgan Kaufmann.
18. David P. Helmbold and Philip M. Long. Tracking drifting concepts by minimizing disagreements. *Machine Learning*, 14:27–45, 1994.
19. Geoff Hulten, Laurie Spencer, and Pedro Domingos. Mining Time-Changing Data Streams. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*, pages 97 – 106, 2001.
20. Thorsten Joachims. Estimating the generalization performance of a SVM efficiently. In Pat Langley, editor, *Proceedings of the International Conference on Machine Learning*, pages 431–438, San Francisco, CA, USA, 2000. Morgan Kaufman.
21. Thorsten Joachims, Dayne Freitag, and Tom Mitchell. WebWatcher: A tour guide for the world wide web. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, volume 1, pages 770 – 777. Morgan Kaufmann, 1997.
22. George H. John and Pat Langley. Static Versus Dynamic Sampling for Data Mining. In *Proceedings of the Second International Conference on Knowledge Discovery in Databases and Data Mining*, 1996.
23. Ralf Klinkenberg. Maschinelle Lernverfahren zum adaptiven Informationsfiltern bei sich verändernden Konzepten. Master's thesis, Computer Science Department, University of Dortmund, Dortmund, Germany, February 1998. http://www-ai.cs.uni-dortmund.de/DOKUMENTE/klinkenberg_98a.ps.gz.

24. Ralf Klinkenberg. Predicting phases in business cycles under concept drift. In Andreas Hotho and Gerd Stumme, editors, *LLWA 2003 – Tagungsband der GI-Workshop-Woche* Lehren – Lernen – Wissen – Adaptivität*, Proceedings of the Workshop Week* Teaching – Learning – Knowledge – Adaptivity *of the National German Computer Science Society (GI) / Annual Workshop on Machine Learning*, pages 3–10, Karlsruhe, Germany, October 2003. http://km.aifb.uni-karlsruhe.de/ws/LLWA/fgml/final/klinkenberg.pdf.
25. Ralf Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis (IDA), Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift*, 8(3):281–300, May 2004.
26. Ralf Klinkenberg and Thorsten Joachims. Detecting concept drift with support vector machines. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 487–494, San Francisco, CA, USA, 2000. Morgan Kaufmann. http://www-ai.cs.uni-dortmund.de/DOKUMENTE/klinkenberg_joachims_2000a.ps.gz.
27. 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/AAAI-98 Workshop* Learning for Text Categorization, pages 33–40, Menlo Park, CA, USA, 1998. AAAI Press. http://www-ai.cs.uni-dortmund.de/DOKUMENTE/klinkenberg_renz_98a.ps.gz.
28. Ralf Klinkenberg and Stefan Rüping. Concept drift and the importance of examples. In Jürgen Franke, Gholamreza Nakhaeizadeh, and Ingrid Renz, editors, *Text Mining – Theoretical Aspects and Applications*, pages 55–77. Physica-Verlag, Berlin, Germany, 2003.
29. Jeremy Z. Kolter and Marcus A. Maloof. Using Additive Expert Ensembles to Cope with Concept Drift. In *Proceedings of the 22nd International Conference on Machine Learning (ICML-2005)*, pages 449–456, New York, NY, USA, 2005. ACM Press.
30. Miroslav Kubat, Joao Gama, and Paul E. Utgoff. *Intelligent Data Analysis (IDA) Journal, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift, Vol.8, No.3*. 2004.
31. 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.
32. Gerhard Kunisch. Anpassung und Evaluierung statistischer Lernverfahren zur Behandlung dynamischer Aspekte in Data Mining. Master thesis, Fachbereich Informatik, Universität Ulm, Germany, June 1996.
33. Ken Lang. NewsWeeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML '95)*, pages 331–339, San Francisco, CA, USA, 1995. Morgan Kaufmann.
34. Carsten Lanquillon. Dynamic neural classification. Master thesis, Fachbereich Informatik, Universität Braunschweig, Germany, October 1997.
35. Herbert K. H. Lee and Merlise A. Clyde. Lossless Online Bayesian Bagging. *Journal of Machine Learning Research*, 5:143–151, February 2004.
36. Tom Mitchell, Rich Caruana, Dayne Freitag, John McDermott, and David Zabowski. Experience with a learning personal assistant. *Communications of the ACM (CACM)*, 37(7):81–91, July 1994.

37. Katharina Morik and Stefan Rüping. A multistrategy approach to the classification of phases in business cycles. In Taprio Elomaa, Heikki Mannila, and Hannu Toivonen, editors, *Machine Learning: ECML 2002*, volume 2430 of *Lecture Notes in Artificial Intelligence*, pages 307–318, Berlin, 2002. Springer.
38. Gholamreza Nakhaeizadeh, I. Bruha, and Charles Taylor, editors. *Workshop Notes on Dynamically Changing Domains: Theory Revision and Context Dependence Issues, 9th European Conf. on Machine Learning (ECML '97)*, Prague, Czech Republic, April 1997.
39. Nikunj C. Oza and Stuart Russell. Online Bagging and Boosting. In *Eighth International Workshop on Artificial Intelligence and Statistics*, Key West, Florida, USA, 2001.
40. Oliver Ritthoff, Ralf Klinkenberg, Simon Fischer, Ingo Mierswa, and Sven Felske. YALE: Yet Another Machine Learning Environment. In Ralf Klinkenberg, Stefan Rüping, Andreas Fick, Nicola Henze, Christian Herzog, Ralf Molitor, and Olaf Schröder, editors, *LLWA 01 – Tagungsband der GI-Workshop-Woche Lernen – Lehren – Wissen – Adaptivität*, number 763 in Forschungsberichte des Fachbereichs Informatik, Universität Dortmund, pages 84–92, Dortmund, Germany, October 2001. ISSN 0933-6192. http://yale.sf.net/.
41. Stefan Rüping. *mySVM Manual*. Universität Dortmund, Lehrstuhl Informatik VIII, 2000. http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.
42. Stefan Rüping. Incremental learning with support vector machines. In Nick Cercone, T.Y. Lin, and Xindong Wu, editors, *Proceedings of the 2001 IEEE International Conference on Data Mining (ICDM '01)*, pages 641–642. IEEE, 2001.
43. G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.
44. Martin Scholz. Comparing Knowledge-Based Sampling to Boosting. Technical Report 26, Collaborative Research Center on the Reduction of Complexity for Multivariate Data Structures (SFB 475), University of Dortmund, Dortmund, Germany, 2005.
45. Martin Scholz. Knowledge-Based Sampling for Subgroup Discovery. In Katharina Morik, Jean-Francois Boulicaut, and Arno Siebes, editors, *Local Pattern Detection*, volume LNAI 3539 of *Lecture Notes in Artificial Intelligence*, pages 171–189. Springer, 2005.
46. Martin Scholz. Sampling-Based Sequential Subgroup Mining. In R. L. Grossman, R. Bayardo, K. Bennett, and J. Vaidya, editors, *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '05)*, pages 265–274, Chicago, Illinois, USA, August 2005. ACM Press.
47. Martin Scholz and Ralf Klinkenberg. An Ensemble Classifier for Drifting Concepts. In J. Gama and J. S. Aguilar-Ruiz, editors, *Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams*, pages 53–64, Porto, Portugal, October 2005. In conjunction with ECML/PKDD '05, 16th European Conference on Machine Learning (ECML) and 9th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD). http://www.niaad.liacc.up.pt/~jgama/IWKDDS/.
48. Kenneth O. Stanley. Learning Concept Drift with a Committee of Decision Trees. Technical Report AI-03-302, Department of Computer Sciences, University of Texas at Austin, Austin, TX, USA, 2003.

49. W. Nick Street and YongSeog Kim. A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '01)*, pages 377–382, 2001.
50. Charles Taylor, Gholamreza Nakhaeizadeh, and Carsten Lanquillon. Structural change and classification. In G. Nakhaeizadeh, I. Bruha, and C. Taylor, editors, *Workshop Notes on Dynamically Changing Domains: Theory Revision and Context Dependence Issues, 9th European Conf. on Machine Learning (ECML '97), Prague, Czech Republic*, pages 67–78, April 1997.
51. Winfried Theis and Claus Weihs. Clustering techniques for the detection of business cycles. Technical Report 40, Collaborative Research Center on the Reduction of Complexity for Multivariate Data Structures (SFB 475), University of Dortmund, Dortmund, Germany, 1999.
52. P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.
53. Georg Veltmann. Einsatz eines Multiagentensystems zur Erstellung eines persönlichen Pressespiegels. Master's thesis, Computer Science Department, University of Dortmund, Dortmund, Germany, May 1997.
54. Haixun Wang, Wei Fan, Philip S. Yu, and Jiawei Han. Mining Concept-Drifting Data Streams using Ensemble Classifiers. In L. Getoor, T. E. Senator, P. Domingos, and C. Faloutsos, editors, *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '03)*, pages 226–235, Washington, DC, USA, 2003. ACM Press.
55. Gerhard Widmer and Miroslav Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(2):69–101, 1996.
56. Ian Witten and Eibe Frank. *Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 2000.