Meta-Learning, Model Selection, and Example Selection in Machine Learning Domains with Concept Drift

Ralf Klinkenberg
University of Dortmund, Computer Science Department, Artificial Intelligence Unit
44221 Dortmund, Germany
Ralf.Klinkenberg@cs.uni-dortmund.de
http://www-ai.cs.uni-dortmund.de/

Abstract

For many 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. The interest of the user may change over time. Machine learning approaches handling concept drift have been shown to outperform more static approaches ignoring it in experiments with different types of simulated concept drifts on real-word text data and in experiments on real-world data for the task of classifying phases in business cycles exhibiting real concept drift. While previous concept drift handling approaches only use a single base learning algorithm and employ this same base learner at each step in time, this paper proposes a meta-learning approach allowing the use of alternative learners and automatically selecting the most promising base learner at each step in time. This work in progress investigates, if such a context-dependent selection of the base learner leads to a better adaptation to the drifting concept, i.e. to lower classification error rates, than approaches based on single base learner only. Furthermore it investigates, how much the proposed meta-learning approach allows to speed up the selection process and how much of the gained reduction in the error rate may be lost by that speed-up. The approaches with and without base learner selection and meta-learning are to be compared in experiments using real-world data from the above mentioned domains with simulated and real-world concept drifts, respectively.[1]

1 Introduction

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

The same problem occurs in information filtering, i.e. the adaptive classification of documents with respect to a particular user interest. With the amount of online information and communication growing rapidly, there is an increasing need for automatic information filtering. Information filtering techniques are used, for example, to build personalized news filters, which learn about the news-reading preferences of a user [Lang, 1995; Veltmann, 1997]. Both the interest of the user, i. e. the concept underlying the classification of the texts, and the document content change over time. A filtering system should be able to adapt to such concept changes.

After a formalization of the concept drift problem in Section 2.1 and shortly describing previous approaches to handling concept drift in Sections 2.2 and 2.3, Section 3 describes two machine learning approaches handling this type of concept drift that have been shown to outperform more static approaches ignoring concept drift in experiments with different types of simulated concept drifts on real-word text data (see Section 4 and [Klinkenberg & Joachims, 2000; Klinkenberg & Rüping, 2003; Klinkenberg, 2004]) and on real-world data with real concept drift (see Section 5 and [Klinkenberg, 2003]).

These methods either maintain an adaptive time window on the training data [Klinkenberg & Joachims, 2000], select representative training examples, or weight the training examples [Klinkenberg & Rüping, 2003; Klinkenberg, 2004]. The key idea is to automatically adjust the window size, the example selection, and the example weighting, respectively, so that the estimated generalization error is minimized. The approaches are both theoretically well-founded as well as effective and efficient in practice. Since they do not require complicated parameterization, they are simpler to use and more robust than comparable heuristics.

Section 3 also proposes a meta-learning approach adding an additional degree of freedom to these two concept drift handling techniques by not only selecting the examples best suited for learning at a certain point in time, but by also selecting the base learner (and its parameterization) from a given set of base learners (and parameterizations) that seems to be most approriate for achieving the smallest expected classification error at that point of time.

Furthermore this section outlines, how this approach can be extended to use meta-data on the current situation and experience from previous time points to predict such a selection based on the meta-data and thereby reduce the search space for a good selection. In experiments on the data sets mentioned before, it will first be analysed, whether choosing the base learner at each point in time separately using the first meta-learning approach reduces the classification error, and then, whether the second

[1] Experimental results not covered in this paper are presented at LWA-2005/FGML-2005 and in [Klinkenberg, 2005].

meta-learning approach allows to reduce the model selection search space and computation time without sacrificing too much accuracy. Section 6 summarizes the results of this paper and provides an outlook on further 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 [Klinkenberg & Joachims, 2000; Klinkenberg, 2003]. 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.

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

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

2.2 Heuristic Approaches to Concept Drift

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

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

2.3 Theoretical Approaches to 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,[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.

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

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

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

In practice, however, it usually cannot be guaranteed that the application at hand obeys these restrictions, e.g. a reader of electronic news may change his interests (almost) arbitrarily often and radically. Furthermore the large time window sizes, for which the theoretical results hold, would be impractical. Hence more application oriented approaches rely on far smaller windows of fixed size or on window adjustment heuristics that allow far smaller window sizes and usually perform better than fixed and/or larger windows [Widmer & Kubat, 1996; Lanquillon, 1997; Klinkenberg & Renz, 1998]. While these heuristics are intuitive and work well in their particular application domain, they usually require tuning their parameters, are often not transferable to other domains, and lack a proper theoretical foundation.

2.4 Meta-Learning Approaches to Concept Drift

In some domains, the concept changes in dependence on some (hidden) context variables. For this kind of domains, meta learning schemes were developed for identifying the context variables, i.e. variables indicating the current context, and for learning separate models for each context (see e.g. [Widmer & Kubat, 1996; Widmer, 1997]).

While these approaches assume the existence of such context indicating variables and keep their base learners fixed, the meta learning frameworks proposed in this paper do not assume that there are any context indicator variables and allow to select a different base learner at each point in time.

3 Handling Concept Drift

In a learning problem with drifting concepts as introduced in Section 2.1, we face the problem to decide, how much information from past examples may be used to find a hypothesis that is adequate to predict the class information of future data. Since we do not know, if and when a concept drift happens, there are two opposing effects: On the one hand, the older the data is, the more likely it is that its probability distribution differs from the current distribution that underlies the process, so that the data may be misleading. On the other hand, the more data is used in the learning process, the better the results are if no concept drift occurred since the data arrived.

In this section we present different approaches for learning drifting concepts. They differ in the way previous examples are used to construct a new hypothesis. All of our

approaches share the assumption, that concept drifts do not reverse, i.e. newer examples are always more important than older ones. This assumption was implemented by a common scheme for estimating the performance of a learner: In all experiments, the performance was only calculated on the last batch of data, regardless of how many batches were used in training.

Some base learners offer very effective and efficient error estimators. For support vector machines (SVMs), we used the so-called $\xi\alpha$-estimator of [Joachims, 2000], which estimates the leave-one-out-error of a SVM based solely on the one SVM solution learned with all examples, to get a good estimation of the performance but still be efficient,.

3.1 Adaptive Time Windows

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 [Mitchell et al., 1994] or involve complicated heuristics [Widmer & Kubat, 1996; Lanquillon, 1997; Klinkenberg & Renz, 1998]. A fixed window size makes strong assumptions about how quickly the concept changes. While heuristics can adapt to different speed and amount of drift, they involve many parameters that are difficult to tune.

In [Klinkenberg & Joachims, 2000], Klinkenberg and Joachims presented an approach to automatically 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. For SVMs, to get an estimate of the generalization error, a special form of $\xi\alpha$-estimates [Joachims, 2000] is used.

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

$$\begin{aligned} &\text{batch } t \\ &\text{batch } t-1, \text{batch } t \\ &\text{batch } t-2, \text{batch } t-1, \text{batch } t \\ &\vdots \end{aligned}$$

For each window size it computes a $\xi\alpha$-estimate based on the result of training, considering only the last batch for the estimation (batch t), that is the m most recent training examples $z_{(t,1)}, ..., z_{(t,m)}$ This reflects the assumption that the most recent examples are most similar to the new examples in batch $t+1$. The window size minimizing the $\xi\alpha$-estimate of the error rate is selected by the algorithm and used to train a classifier for the current batch.

The window adaptation algorithm can be summarized as follows:

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

3.2 Example Selection

[Klinkenberg & Rüping, 2003] proposed an extension of the time window adjustment approach by [Klinkenberg & Joachims, 2000] allowing to select or deselect batches individually for the training set rather than only allowing an uninterrupted sequence of batches (time window) as training set.

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

In a second step, the information about the error of the classifier can be used to build a training set for the actual classifier. We can exclude all batches from the new training set, which have a significantly higher classification error than the most recent batch (*batch selection*). By this, we hope to train the final classifier on all data generated by the current model in a way similar to the example selection scheme in Section 3.1 (*adaptive time window*).

3.3 Meta-Learning for Selecting the Best Base Learner: Model Selection for Error Reduction

The two previously described concept drift approaches select the examples best suited for learning using a fixed base learner, e.g. SVMs. While alternative base learners like e.g. a decision-tree or rule-based or instance-based learners could also be used with these approaches, as long as they provide reliable error estimates, the base learner is fixed and has to be chosen before applying these approaches.

The idea now is, to also let the concept drift handling frame work select the most appropriate learner at each time step. SVMs usually perform well with large data sets and hence in phases with no or only little concept drift. However, they may have problems with very small data sets. Shortly after a concept drift, when only few representative examples are available for training, another base learner may be more appropriate.

By not only selecting the example set, but also the learner (and its parameterization), we add an additional degree of freedom to the model selection process. In the following we call this type of framework selecting the most promising combination of example set and base learner (and parameterization) *meta learner of type 1*. The meta learner selects the combination of example selection and base learner selection that minimizes the expected error.

3.4 Meta-Learning for Predicting the Best Model Selection to Speed-Up Learning

The *meta learner of type 1* expands the search space for the best expected model selection significantly in the hope to reduce the classification error. In this section we therefore introduce a second meta-learning framework, *meta learner of type 2*, that reduces this expanded search space in order to speed up the search.

While *meta learner of type 1* did not learn on the meta-level, but did a complete search of the search space on that level, we now want to use meta-learning to guide this search. One may expect the choice of the appropriate learner at a certain point in time to depend on the expected best size of the training set. Hence the attributes of the training examples on the meta-level include the following, where one training example is constructed for each previously seen batch after the previously chosen base learner has been evaluated on that batch: number of old batches used for training at the previous batch[3], number of non-interrupted most recent batches used for training (i.e. number of batches without a detected concept drift), most succesful learner on the previous batch[4], most succesful learner overall on all batches seen so far[5], and possibly further meta-attributes.

The first goal attribute to be learned by the meta-learner is the prediction of the best (n_1) base learner(s) given the description of a time point by these meta attributes. The second goal attribute to be learned by the meta-learner is the prediction of the best (n_2) example set selection(s) given the description of a time point by these meta attributes. The meta learner is trained whenever a new meta example becomes available, i.e. after each batch.

The prediction of the meta model can then be used to predict the best n_1 expected example set selections and the best n_2 expected base learners and to test only test these combinations instead of all possible combinations of base learners and example set selections. The meta learner used for this framework can be any learner providing a ranking of the target class attribute values for a new meta example.

The choice of n_1 and n_2 allows to trade-off learning time (i.e. search space size) and thereby the amount of time required for learning versus classification error. The larger n_1 and n_2, the more complete is the search and hence the better the result can be expected to be.

Furthermore one can set n_1 and n_2 in different ways if only the choice of the example selection or the choice of the base learner are really critical but not both.

[3]The best choice for the base learner can be expected to depend on the (expected best) size of the training set, since some learners are better on small data sets and others on larger ones, i.e. different learners have different learning curves.

[4]Indicator for a locally successful learner.

[5]The choice of the learner certainly depends on the overall data set and its (unknown) characteristics and hence the previous success on this data set should be a relevant indicator.

4 Evaluation on Simulated Concept Drift Scenarios

4.1 Experimental Setup and Evaluation Scheme: Simulated Scenarios on Business News

In order to evaluate the learning approaches for drifting concepts proposed in this paper, three simple non-adaptive data management approaches are compared to the adaptive time window approach and to the batch selection strategy, 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.[6]
- *"Adaptive Window"*: The window adjustment algorithm described in Section 3.1 (and in [Klinkenberg & Joachims, 2000]) adapts the window size to the current concept drift situation.
- *"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, i.e. they are selected for the final training set. The weight of all other examples is set to zero, i.e. they are deselected (see Section 3.2 and [Klinkenberg & Rüping, 2003]).

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

The performance of the classifiers is measured by prediction error. All reported results are estimates averaged over four runs. For more detailed results including precision and recall results and graphical plots of the performance and the selected window size over time see [Klinkenberg & Joachims, 2000; Klinkenberg & Rüping, 2003; Klinkenberg, 2004].

While most evaluation methods for machine learning, like e. g. cross-validation, assume that examples are independent and identically distributed, this assumption is clearly unrealistic in the presence of concept drift. Therefore the concept drift approaches proposed in this paper use $\xi\alpha$-estimates [Joachims, 2000] instead of cross-validation to estimate and optimize the performance of a particular parameterization at each learning step and only estimate the performance on the currently last batch (see Section 3.1). This is not only more appropriate for the concept drift situation at hand, but also more efficient, because the $\xi\alpha$-estimator can be computed within a single training run.

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

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

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

In the three scenarios, a document is considered relevant at a certain point in time, if it matches the interest of the simulated user at that time. For each TREC topic and each batch in each scenario the probability that a document from this topic is relevant for the user interest at this time (batch) is specified. In the scenarios simulated here, the user interest changes between the topics 1 and 3. Documents of the classes 4, 5, and 6 are never relevant in any of these scenarios. Figure 1 shows the probability of being relevant for a document of category 1 at each batch for each of the three scenarios. Documents of category 3 are specified to always have the inverse relevance probability of documents of category 1, i.e. *1.0 - relevance of category 1*. In the first scenario (*scenario A*), first documents of category 1 are considered relevant for the user interest and all other documents irrelevant. This changes abruptly (concept shift) in batch 10, where documents of category 3 are relevant and all others irrelevant. In the second scenario (*scenario B*), again first documents of category 1 are considered relevant for the user interest and all other documents irrelevant. This changes slowly (concept drift) from batch 8 to batch 12, where documents of category 3 are relevant and all others irrelevant. The third scenario (*scenario C*) simulates an abrupt concept shift in the user interest from category 1 to category 3 in batch 9 and back to category 1 in batch 11.

The experiments were conducted with the machine learning environment YALE [Ritthoff et al., 2001; Fischer et al., 2003; Mierswa et al., 2003a; Mierswa et al., 2003b].[8] For all time window and example selection approaches, support vector machines were used as core learning algorithm. Here we chose the SVM implementation *mySVM*[9] [Rüping, 2000]. Since linear kernels are the standard kernel type used for text classification problems, and since more complex kernel types usually do not perform better on text classification tasks, only linear and no other kernel types were tried here. After preliminary experiments to determine a good value for the capacity constant C of the SVM, which allows to trade off model complexity versus generalization, testing the values $C \in \{1, 10, 100, 1000\}$, the value $C = 1000$ was chosen for the experiments described here.

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

[8] `http://yale.cs.uni-dortmund.de/` = http://yale.sf.net/

[9] `http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/`

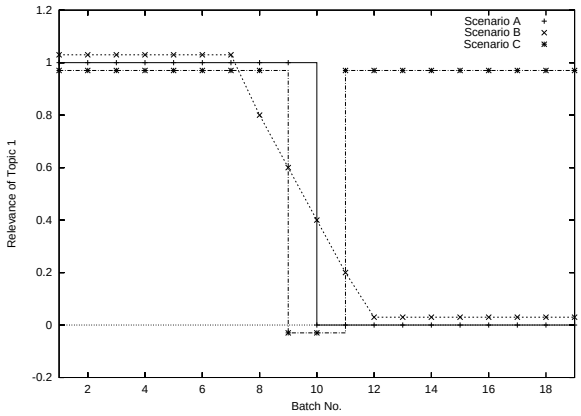

Figure 1: Relevance of the TREC topic 1 in the concept change scenarios A, B, and C. The relevance of TREC topic 3 is *1.0 - relevance of topic 1*. The relevance of all other topics is always zero.

Table 1: Error of all time window and example selection methods for all scenarios averaged over 4 trials with 20 batches.

	Full Memory	No Memory	Fixed Size	Adaptive Size	Batch Selection
Scen. A	21.11%	11.16%	9.03%	**6.65%**	**6.15%**
Scen. B	21.30%	12.64%	9.76%	**9.06%**	**9.33%**
Scen. C	8.60%	12.73%	11.19%	**8.56%**	**7.55%**

4.2 Experimental Results

Table 1 shows the results of all static and adaptive time window and batch selection approaches on all scenarios in terms of prediction error. The adaptive time window approach and the batch selection strategy clearly outperform the trivial non-adaptive approaches.

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

From the results of the non-adaptive approaches (full memory, no memory, and fixed size), we can see that the error is the lowest if the time window contains all time points from the current model and no others. For example in scenario A, as depicted in Figure 2, as long as no concept drift occurs, the full memory approach has the lowest error. Immediately after the concept drift the no memory approach

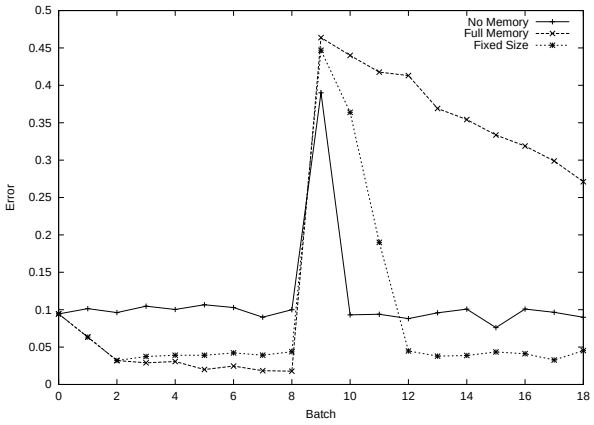

Figure 2: Classification errors of the trivial approaches on each batch in scenario A (averaged over 4 runs).

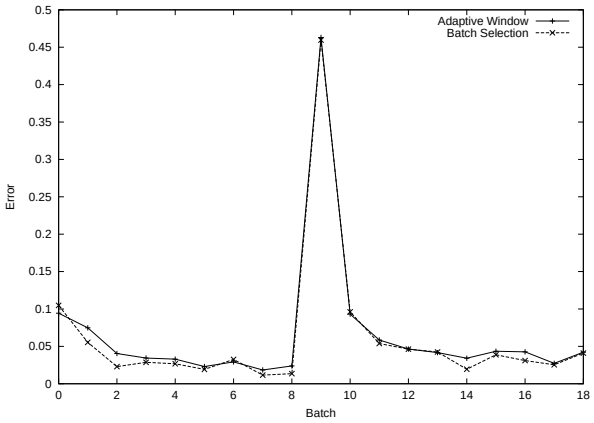

Figure 3: Classification errors of the batch selection approaches on each batch in scenario A (averaged over 4 runs).

quickly returns to its previous error level, while the fixed size memory approach takes longer until it finally reaches a lower error than the no memory approach again. For the full memory approach, even nine time points after the concept shift the error rate is still three times higher than the error of the other strategies. Of course, these findings are not very surprising.

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

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

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

A more detailed description of the results including plots showing the performance of the different approaches and the selected window sizes over time can be found in [Klinkenberg & Joachims, 2000; Klinkenberg & Rüping, 2003; Klinkenberg, 2004].

4.3 Concept Drift Handling with Meta-Learning

Results for allowing the concept drift framework to choose from different base learners, i.e. support vector machines (SVMs), decision tree learners (Weka/J48), and nearest neighbor learning and comparing the performance of the approaches with base learner selection and meta-learning will be presented in [Klinkenberg, 2005].

5 Evaluation on a Real-World Concept Drift Problem From Economics

5.1 Predicting Phases in Business Cycles

The second evaluation domain is a task from economics based on real-world data. The quarterly data describes the West German Business Cycles From 1954 to 1994 [Heilemann & Münch, 1998; Heilemann & Münch, 1999]. Each of the 158 examples is described by 13 indicator variables. The task is to predict the current phase of the business cycle.

While Heilemann and Münch use a model with four phases for the business cycle, in which each cycle consists of a lower turning point, an upswing, an upper turning point, and a down swing, where the turning points cover several month, Theis and Weihs have shown that in clustering analysis of West German macro-economic data at most three clusters can be identified [Theis & Weihs, 1999]. The first two clusters roughly correspond to the cycle phases of upswing and downswing and the eventual third cluster corresponds to the period of the oil-crisis around 1971. This suggests that two phases instead of four may be more suitable for the description of the business data.

While linear discriminant analysis as a baseline model achieves 54% accuracy using uni-variate rules (according to [Morik & Rüping, 2002a]) for the four phase model on this data set, sophisticated statistical models achieve 63% accuracy. Sondhauß and Weihs incorporate economic background knowledge into business cycle analysis by using advanced Markov Switching Models to express knowledge about the past and the transition probability to the next phase [Sondhauß & Weihs, 2001].

Morik and Rüping applied an inductive logic programming approach also using domain knowledge on this data set and achieved an accuracy of 53% for the four phase model and of 81.5%[10] for the two phase model [Morik &

[10] For comparison with the results reported later in this section: 81.5% accuracy obviously correspond to a classifi cation error rate of 100% - 81.5% = 18.5%. However, this result was obtained for leave-one-cycle-out validation, i. e. also using data from future cycles and thereby violating the timely ordering of the data. The experiments described in this paper preserve this timely ordering and hence the learners then obviously have less data to learn from.

Table 2: Error of all time window and example selection methods for splits of the business cycle data into 5 and 15 batches, respectively.

	Full Memory	No Memory	Fixed Size	Adaptive Size	Batch Selection
5 batches	32.80%	27.20%	**24.00%**	**24.80%**	**24.80%**
15 batches	28.08%	28.77%	**20.55%**	**24.80%**	**23.29%**

Rüping, 2002b; Morik & Rüping, 2002a]. In the following experiments, we also use this two phase model mapping all time points classified as upper turning point to upswing and all quarters of a year classified as lower turning point to downswing. However, we do not make any use of background domain knowledge.

The timely order of the examples (quarters) was preserved and no artificial concept drift was simulated. The two questions of interest are, whether this domain actually exhibits concept drift behavior and whether our approaches are able handle it.

5.2 Experimental Results

The experiments were again performed using the machine learning environment YALE and the *mySVM* as underlying learner. Two evaluations were performed, one splitting the data into 5 batches of equal size and one splitting it into 15 batches of equal size. Both splits did not change the timely order of the example and did not impose any artificial concept drift.

The results of these two evaluations are shown in Table 2. The results for the fixed time window approach correspond to the results of the fixed size that performed best, i.e. they are the result of an offline parameter optimization considering the performance on all batches. In practice, this optimal fixed size is not known in advance at any particular point in time before the last batch. Hence in a real application one would have to guess this size in advance from experience, while the adaptive time window approach and the batch selection approach are able to adapt automatically online. These optimal results for the fixed size approach are only provided for comparison in order to show the performance that this approach can maximally reach theoretically.

What do the results tell about the domain?

Since even the simple fixed window size approach significantly outperforms the full memory approach, one may conclude that this domain exhibits real concept drift behavior. Otherwise learning on all available training data should allow a better generalization and lead to better results. A closer inspection of the results and the adaptively chosen window sizes suggests that the data of the early years, i.e. the first business cycles, follow slightly other rules than those of the latter years. The automatically chosen window sizes tend to exclude only the first few batches.

What do the results tell about the concept drift handling approaches?

Obviously the adaptive time window approach and the batch selection approach handle this concept drift very well. Only the fixed size time window approach can theoretically compete, if the optimal fixed time window size is known in advance, which is usually not possible in a real-world application. As shown in Table 3, using other fixed window sizes, leads to significant drops in performance to error levels, mostly well above those of the two adaptive approaches.

Therefore it is not surprising that their generalization performance is not as good.

Table 3: Error of the fixed time window method with fixed window sizes from $1...5$ batches for splits of the business cycle data into 5 and 15 batches, respectively.

	No Memory (1 batch)	Fixed Size 2 batches	Fixed Size 3 batches	Fixed Size 4 batches	Fixed Size 5 batches
5 batches	27.20%	**24.00%**	26.40%	32.80%	32.80%
15 batches	28.77%	23.29%	**20.55%**	23.29%	23.97%

The fact that the fixed size approach is competitive in this domain may be due to the cyclic nature of the domain. A well chosen time window length of several business cycles seems to be sufficient for a good generalization result and to sufficiently early drop the data of the first few years that significantly reduce the performance of larger time windows and the full memory approach, because they seem to obey somewhat other rules than the latter years.

So, in conclusion for this domain, if a fixed window size can be optimized offline, which is not possible in an online application, or such a fixed window size can be well guessed by a domain expert, the simple time window approach is quite competitive, For the online setting of this real-world task, however, the two adaptive concept drift adjustment techniques seem to be more appropriate, since they do not rely on an offline optimization or a good expert guess, but adapt to the current concept drift automatically.

5.3 Concept Drift Handling with Meta-Learning

Results for allowing the concept drift framework to choose from different base learners, i.e. support vector machines (SVMs), decision tree learners (Weka/J48), and nearest neighbor learning and comparing the performance of the approaches with base learner selection and meta-learning will be presented in [Klinkenberg, 2005].

6 Summary and Conclusions

This paper proposed a meta-learning framework for handling concept drift automatically selecting the most promising base learner (along with its most promising parametrization) and the best set of examples to learn from at each step in time, so that the estimated generalization error is minimized. This framework is expected to improve the accuracy of the learning system by selecting the most appropriate learner not globally once for all times but for each step in time seperately (on a meta-level) and optionally to speed-up the meta-level process by reducing the search space at each step in time by meta-learning. Restricting the meta-level search space may of course lead to a decrease in accuracy. Experiments using real-world data from the above mentioned domains with simulated and real-world concept drifts, respectively, comparing the performance of the approaches with and without base learner selection and meta-learning are presented in a technical report by the author to appear in November 2005 [Klinkenberg, 2005].

References

Allan, J. (1996). Incremental relevance feedback for information filtering. *Proceedings of the Nineteenth ACM Conference on Research and Development in Information Retrieval* (pp. 270–278). New York, NY, USA: ACM Press.

Balabanovic, M. (1997). An adaptive web page recommendation service. *Proceedings of the First International Conference on Autonomous Agents* (pp. 378–385). New York, NY, USA: ACM Press.

Fischer, S., Klinkenberg, R., Mierswa, I., & Ritthoff, O. (2003). YALE*: Yet Another Learning Environment – Tutorial* (Technical Report CI-136/02 (2nd edition)). Collaborative Research Center on Computational Intelligence (SFB 531), University of Dortmund, Dortmund, Germany. ISSN 1433-3325, http://yale.cs.uni-dortmund.de/.

Heilemann, U., & Münch, H. J. (1998). *Forecasting the stage of a business cycle* (Technical Report). Rheinisch-Westfälisches Institut für Wirtschaftsforschung (RWI), Essen, Germany and Gerhard-Mercator-University Duisburg, Duisburg, Germany. Paper presented at the Project LINK-Meeting in Rio de Janeiro, September 14-18, 1998.

Heilemann, U., & Münch, H. J. (1999). *Classification of west german business cycles* (Technical Report 11). Collaborative Research Center on Reduction of Complexity for Multivariate Data (SFB 475), University of Dortmund, Germany.

Helmbold, D. P., & Long, P. M. (1994). Tracking drifting concepts by minimizing disagreements. *Machine Learning, 14*, 27–45.

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

Klinkenberg, R. (1998). Maschinelle Lernverfahren zum adaptiven Informationsfiltern bei sich verändernden Konzepten. Masters thesis, Computer Science Department, University of Dortmund, Germany.

Klinkenberg, R. (2003). Predicting phases in business cycles under concept drift. *LLWA 2003 – Tagungsband der GI-Workshop-Woche Lehren – Lernen – Wissen – Adaptivität* (pp. 3–10). Karlsruhe, Germany.

Klinkenberg, R. (2004). Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis (IDA), Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift*. Volume 8, Number 3.

Klinkenberg, R. (2005). *Meta-learning for concept drift handling* (Technical Report). Computer Science Department, University of Dortmund, Dortmund, Germany. Technical report to appear in November 2005.

Klinkenberg, R., & Joachims, T. (2000). Detecting concept drift with support vector machines. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)* (pp. 487–494). San Francisco, CA, USA: Morgan Kaufmann.

Klinkenberg, R., & Renz, I. (1998). Adaptive information filtering: Learning in the presence of concept drifts. *Workshop Notes of the ICML/AAAI-98 Workshop on* Learning for Text Categorization *held at the* Fifteenth International Conference on Machine Learning *(ICML-98)* (pp. 33–40). Menlo Park, CA, USA: AAAI Press.

Klinkenberg, R., & Rüping, S. (2003). Concept drift and the importance of examples. In J. Franke, G. Nakhaeizadeh and I. Renz (Eds.), *Text mining – Theoretical aspects and applications*, 55–77. Heidelberg, Germany: Physica-Verlag.

Kuh, A., Petsche, T., & Rivest, R. L. (1991). Learning time-varying concepts. *Advances in Neural Information Processing Systems* (pp. 183–189). San Mateo, CA, USA: Morgan Kaufmann.

Kunisch, G. (1996). Anpassung und Evaluierung statistischer Lernverfahren zur Behandlung dynamischer Aspekte in Data Mining. Masters thesis, Computer Science Department, University of Ulm, Germany.

Lang, K. (1995). NewsWeeder: Learning to filter netnews. *Proceedings of the Twelfth International Conference on Machine Learning (ICML '95)* (pp. 331–339). San Francisco, CA, USA: Morgan Kaufmann.

Lanquillon, C. (1997). Dynamic neural classification. Masters thesis, Computer Science Department, University of Braunschweig, Germany.

Mierswa, I., Klinkenberg, R., Fischer, S., & Ritthoff, O. (2003a). A Flexible Platform for Knowledge Discovery Experiments: YALE – Yet Another Learning Environment. *LLWA-2003: Lehren – Lernen – Wissen – Adaptivität, Proceedings of the Workshop of the Special Interest Groups Machine Learning, Knowledge Discovery, and Data Mining (FGML), Intelligent Tutoring Systems (ILLS), and Adaptivity and User Modeling in Interactive Systems (ABIS) of the German Computer Science Society (GI)*. University of Karlsruhe, Karlsruhe, Germany. Short version of [Mierswa et al., 2003b].

Mierswa, I., Klinkenberg, R., Fischer, S., & Ritthoff, O. (2003b). *A Flexible Platform for Knowledge Discovery Experiments:* YALE *– Yet Another Learning Environment* (Technical Report). Collaborative Research Center on Computational Intelligence (SFB 531), University of Dortmund, Dortmund, Germany. ISSN 1433-3325. Long version of [Mierswa et al., 2003a].
http://yale.cs.uni-dortmund.de/.

Mitchell, T., Caruana, R., Freitag, D., McDermott, J., & Zabowski, D. (1994). Experience with a learning personal assistant. *Communications of the ACM, 37*, 81–91.

Morik, K., & Rüping, S. (2002a). *An inductive logic programming approach to the classification of phases in business cycles* (Technical Report 19). Collaborative Research Center on Reduction of Complexity for Multivariate Data (SFB 475), University of Dortmund, Dortmund, Germany.

Morik, K., & Rüping, S. (2002b). A multistrategy approach to the classification of phases in business cycles. *Machine Learning: ECML 2002* (pp. 307–318). Berlin, Germany: Springer.

Ritthoff, O., Klinkenberg, R., Fischer, S., Mierswa, I., & Felske, S. (2001). YALE: Yet Another Machine Learning Environment. *LLWA'01 – Tagungsband der GI-Workshop-Woche* Lernen – Lehren – Wissen – Adaptivität (pp. 84–92). University of Dortmund, Dortmund, Germany. Technical Report 763, ISSN 0933-6192.
http://yale.cs.uni-dortmund.de/.

Rüping, S. (2000). *mysvm manual*. Artificial Intelligence Unit, Computer Science Department, University of Dortmund, Germany.
http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.

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

Sondhauß, U., & Weihs, C. (2001). Incorporating background knowledge for better prediction of cycle phases. *Workshop notes of the IJCAI-01 Workshop on* Learning from Temporal and Spatial Data (pp. 38–44). Menlo Park, CA, USA: AAAI Press. Held in conjunction with the International Joint Conference on Artificial Intelligence (IJCAI).

Taylor, C., Nakhaeizadeh, G., & Lanquillon, C. (1997). Structural change and classification. *Workshop Notes of the ECML-97 Workshop on* Dynamically Changing Domains: Theory Revision and Context Dependence Issues *held at the* Ninth European Conference on Machine Learning (pp. 67–78).

Theis, W., & Weihs, C. (1999). *Clustering techniques for the detection of business cycles* (Technical Report 40). Collaborative Research Center on Reduction of Complexity for Multivariate Data (SFB 475), University of Dortmund, Dortmund, Germany.

Veltmann, G. (1997). Einsatz eines Multiagentensystems zur Erstellung eines persönlichen Pressespiegels. Master's thesis, Computer Science Department, University of Dortmund, Germany.

Widmer, G. (1997). Tracking context changes through meta-learning. *Machine Learning, 27*, 259–286.

Widmer, G., & Kubat, M. (1996). Learning in the presence of concept drift and hidden contexts. *Machine Learning, 23*, 69–101.