Predicting Phases in Business Cycles Under Concept Drift

Ralf Klinkenberg

University of Dortmund, Germany
Department of Computer Science, Artificial Intelligence Unit
E-Mail: klinkenberg@ls8.cs.uni-dortmund.de
WWW: 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. Both the interest of the user and the document content change over time. Machine learning approaches handling this type of 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. In this paper, these approaches to learning drifting concepts are applied to the problem of classifying phases in business cycles. Their performance is compared to the more static approaches on real-world data for this classification task, in order to evaluate whether this domain also exhibits concept drifts and whether the concept drift approaches also allow performance gains in this domain. While previous studies were based on simulated concept drift scenarios, the experiments in this domain are not based on any simulated drift, but on the real concept drift inherent to this real-world data. Hence this paper provides significant support for the applicability of the proposed machine learning approaches to handling concept drift in real-world problems.

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], to filter e-mail [Cohen, 1996], or to guide a user's search on the World Wide Web [Joachims *et al.*, 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. This paper proposes 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 5 and [Klinkenberg and Joachims, 2000; Klinkenberg and Rüping, 2003; Klinkenberg, 2003]). They learn drifting concepts effectively and efficiently and require little parameterization.

In this paper, these approaches to learning drifting concepts are applied to the problem of classifying phases in business cycles. Their performance is compared to the more static approaches on real-world data for this classification task, in order to evaluate whether this domain exhibits concept drifts and whether the concept drift approaches also allow performance gains in this domain. While previous studies were based on simulated concept drift scenarios, the experiments in this domain are not based on any simulated drift, but on the real concept drift inherent to this real-world data. Hence this paper provides significant support for the applicability of the proposed machine learning approaches to handling concept drift in real-world problems.

After formalizing the concept drift problem in Section 2.1 and shortly describing previous approaches to handling concept drift in Sections 2.2 and 2.3, the underlying machine learning method used for the experiments presented in this paper, support vector machines (SVM), is depicted in Section 3 including a subsection on effective and efficient performance estimation for SVMs (Section 3.1). Section 4 proposes two approaches for handling concept drift making use of SVMs and these error estimators. These approaches are then evaluated on several simulated concept drift scenarios on real-world text data (Section 5) and on an economic problem exhibiting real concept drift (Section 6). Section 7 summarizes the results of this paper.

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 and 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.

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

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

2.2 Heuristic Approaches to 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 and Kubat, 1996; Lanquillon, 1997; Klinkenberg and Renz, 1998] or by weighting data or parts of the hypothesis according to their age and/or utility for the classification task [Kunisch, 1996; Taylor *et al.*, 1997]. The latter approach of weighting examples has already been used for information filtering in the incremental relevance feedback approaches of [Allan, 1996] and [Balabanovic, 1997]. In this paper, the earlier approach maintaining a window of adaptive size is explored. 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,[1] but drifting concepts are provably efficiently learnable (at least for certain concept classes), if the rate or the extent of drift is limited in particular ways.

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

While Helmbold and Long restrict the extend of drift, Kuh, 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 and Kubat, 1996; Lanquillon, 1997; Klinkenberg and Renz, 1998]. While these heuristics are intuitive and work well in their particular application domain, they usually require tuning their parameters, are often not transferable to other domains, and lack a proper theoretical foundation.

3 Support Vector Machines (SVM)

Support vector machines are based on the principle of structural risk minimization (SRM) [Vapnik, 1998] from statistical learning theory. In their basic form, SVMs learn linear decision rules,

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

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

Optimization Problem 1 (SVM (primal))

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

$$\textit{subject to:} \quad \forall_{i=1}^{n} : y_i[\vec{w} \cdot \vec{x}_i + b] \geq 1 - \xi_i \qquad (2)$$

$$\forall_{i=1}^{n} : \xi_i \geq 0 \qquad (3)$$

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

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

Optimization Problem 2 (SVM (dual))

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

$$\textit{subject to:} \quad \sum_{i=1}^{n} y_i \alpha_i = 0$$

$$\forall_{i=1}^{n} : 0 \leq \alpha_i \leq C$$

Joachims has developed a fast algorithm for computing the solution to this optimization problem [Joachims, 1999; 2001]. We use this algorithm implemented in *mySVM*[2] for the classification experiments described in this paper.

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

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

Support vectors are those training examples $\vec{x}_i$ with $\alpha_i > 0$ at the solution. From the solution of optimization problem 2, the decision rule can be computed as

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

The training example $(\vec{x}_{usv}, y_{usv})$ for calculating b must be a support vector with $\alpha_{usv} < C$. Finally, the training losses ξ_i can be computed as

$$\xi_i = max(1 - y_i \left[\vec{w} \cdot \vec{x}_i + b\right], 0).$$

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

3.1 Effective and Efficient Error Estimation

The performance of an SVM can be effectively and efficiently estimated by a special form of $\xi\alpha$-estimates [Joachims, 2000]. $\xi\alpha$-estimators are based on the idea of leave-one-out estimation [Lunts and Brailovskiy, 1967]. The leave-one-out estimator of the error rate proceeds as follows. From the training sample $S = ((x_1, y_1), \cdots, (x_n, y_n))$ the first example (x_1, y_1) is removed. The resulting sample $S^{\backslash 1} = ((x_2, y_2), \cdots, (x_n, y_n))$ is used for training, leading to a classification rule $h_{\mathcal{L}}^{\backslash 1}$. This classification rule is tested on the held out example (x_1, y_1). If the example is classified incorrectly it is said to produce a leave-one-out error. This process is repeated for all training examples. The number of leave-one-out errors divided by n is the leave-one-out estimate of the generalization error.

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

$$d = |\{i : (\alpha_i R_\Delta^2 + \xi_i) \geq 1\}| \tag{4}$$

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

$$Err_{\xi\alpha}^n(h_{\mathcal{L}}) = \frac{|\{i : (\alpha_i R_\Delta^2 + \xi_i) \geq 1\}|}{n} \tag{5}$$

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

4 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. To get a good estimation of the performance but still be efficient, we used the so-called $\xi\alpha$-estimator of [Joachims, 2000] (see also Section 3.1), which estimates the leave-one-out-error of a SVM based solely on the one SVM solution learned with all examples.

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

In [Klinkenberg and 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. To get an estimate of the generalization error, a special form of $\xi\alpha$-estimates [Joachims, 2000] (see also Section 3.1) is used.

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

$$
\begin{aligned}
&z_{(t,1)}, \dots, z_{(t,m)} \\
&z_{(t-1,1)}, \dots, z_{(t-1,m)}, z_{(t,1)}, \dots, z_{(t,m)} \\
&z_{(t-2,1)}, \dots, z_{(t-2,m)}, z_{(t-1,1)}, \dots, z_{(t-1,m)}, z_{(t,1)}, \dots, z_{(t,m)} \\
&\vdots
\end{aligned}
$$

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

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

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` $h \in \{0, \dots, t-1\}$
  - train SVM on examples $z_{(t-h,1)}, \dots, z_{(t,m)}$
  - compute $\xi\alpha$-estimate on examples $z_{(t,1)}, \dots, z_{(t,m)}$
- `output:` window size which minimizes $\xi\alpha$-estimate

4.2 Example Selection

[Klinkenberg and Rüping, 2003] proposed an extension of the time window adjustment approach by [Klinkenberg and Joachims, 2000] allowing to select or deselect batches individually for the training set rather than only allowing an un-interleaved 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 4.1 (*adaptive time window*).

5 Evaluation on Simulated Concept Drift Scenarios

5.1 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.[3]
- *"Adaptive Window"*: The window adjustment algorithm described in Section 4.1 (and in [Klinkenberg and 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 4.2 and [Klinkenberg and 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 and Buckley, 1988] of that word in that document. Words occurring less than three times in the training data or occurring in a given list of stop words are not considered. Each document feature vector is normalized to unit length to abstract from different document lengths.

The performance of 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 and Joachims, 2000; Klinkenberg and Rüping, 2003; Klinkenberg, 2003].

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

While most evaluation methods for machine learning, like e. g. cross-validation, assume that examples are independent and identically distributed, this assumption is clearly unrealistic in the presence of concept drift. Therefore the concept drift approaches proposed in this paper use $\xi\alpha$-estimates (see Section 3.1 and [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 4.1). 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.

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.[4] 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; 2003b].[5] For all time window and example selection approaches, support vector machines were used as core learning algorithm. Here we chose the SVM implementation *mySVM*[6] [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.

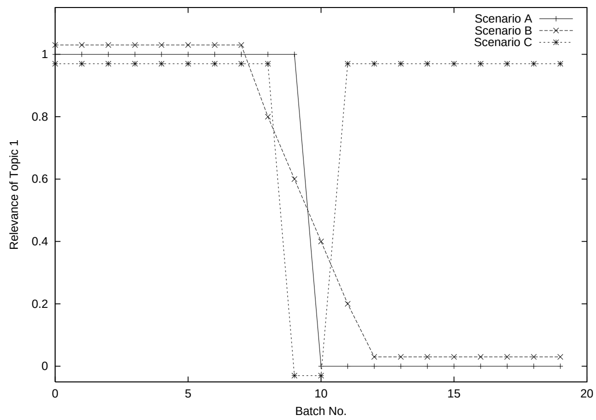

Figure 1: Relevance of the TREC topic 1 in the concept change scenarios A, B, and C. The relevance of TREC topic 3 is *1.0 - relevance of topic 1*. The relevance of all other topics is always zero.

5.2 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 long as no concept drift occurs, the full memory approach has the lowest error. Immediately after the concept drift the no memory approach quickly returns to its previous error level, while the fixed size memory approach

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

[5] `http://yale.cs.uni-dortmund.de/`

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

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%**

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 and Joachims, 2000; Klinkenberg and Rüping, 2003; Klinkenberg, 2003].

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

6.1 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 and Münch, 1998; 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 and 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.

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%**

While linear discriminant analysis as a baseline model achieves 54% accuracy using uni-variate rules (according to [Morik and 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 and 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%[7] for the two phase model [Morik and Rüping, 2002b; 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.

6.2 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

[7] For comparison with the results reported later in this section: 81.5% accuracy obviously correspond to a classification 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. 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%

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.

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.

7 Summary and Conclusions

This paper proposed two methods for handling concept drift with support vector machines using different strategies to account for the different importance of examples for the current target concept to be learned. The methods either maintain an adaptive time window on the training data [Klinkenberg and Joachims, 2000] or select representative training examples [Klinkenberg and Rüping, 2003; Klinkenberg, 2003]. The key idea is to automatically adjust the window size and the example selection, 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. While previous studies were limited to simulated concept drift scenarios, the experiments described in this paper showed, that real-world data exhibits typical concept drift behavior and that the proposed methods for handling concept drift were able to deal with this real concept drift.

8 Acknowledgments

This work was supported by the Deutsche Forschungsgemeinschaft (DFG)[8], Collaborative Research Centers on *Reduction of Complexity for Multivariate Data Structures* (SFB 475)[9] and on *Computational Intelligence* (SFB 531)[10] at University of Dortmund. We are grateful to Stefan Rüping for providing his support vector machine implementation *mySVM*[11].

References

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

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

[Cohen, 1996] 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.

[Fischer *et al.*, 2003] S. Fischer, R. Klinkenberg, I. Mierswa, and O. Ritthoff. 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, August 2003. ISSN 1433-3325, http://yale.cs.uni-dortmund.de/.

[Heilemann and Münch, 1998] Ullrich Heilemann and Heinz Josef Münch. Foreecasting the stage of a business cycle. Technical report, Rheinisch-Westfälisches Institut für Wirtschaftsforschung (RWI), Essen, Germany and Gerhard-Mercator-University Duisburg, Duisburg, Germany, 1998. Paper presented at the Project LINK-Meeting in Rio de Janeiro, September 14-18, 1998.

[Heilemann and Münch, 1999] 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, Germany, 1999.

[8] `http://www.dfg.de/`

[9] `http://www.sfb475.uni-dortmund.de/`

[10] `http://sfbci.uni-dortmund.de/`

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

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

[Joachims *et al.*, 1997] 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-97)*, volume 1, pages 770 – 777. Morgan Kaufmann, 1997.

[Joachims, 1999] T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, 1999.

[Joachims, 2000] Thorsten Joachims. Estimating the generalization performance of a SVM efficiently. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-2000)*, pages 431–438, San Francisco, CA, USA, 2000. Morgan Kaufmann.

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

[Klinkenberg and Joachims, 2000] 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-2000)*, pages 487–494, San Francisco, CA, USA, 2000. Morgan Kaufmann.

[Klinkenberg and Renz, 1998] Ralf Klinkenberg and Ingrid Renz. Adaptive information filtering: Learning in the presence of concept drifts. In Mehran Sahami, Mark Craven, Thorsten Joachims, and Andrew McCallum, editors, *Workshop Notes of the ICML/AAAI-98 Workshop on* Learning for Text Categorization *held at the* Fifteenth International Conference on Machine Learning *(ICML-98)*, pages 33–40, Menlo Park, CA, USA, 1998. AAAI Press.

[Klinkenberg and Rüping, 2003] 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, Heidelberg, Germany, 2003.

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

[Klinkenberg, 2003] 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*, 2003. To appear.

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

[Kunisch, 1996] Gerhard Kunisch. Anpassung und Evaluierung statistischer Lernverfahren zur Behandlung dynamischer Aspekte in Data Mining. Masters thesis, Computer Science Department, University of Ulm, Germany, June 1996.

[Lang, 1995] 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.

[Lanquillon, 1997] Carsten Lanquillon. Dynamic neural classification. Masters thesis, Computer Science Department, University of Braunschweig, Germany, October 1997.

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

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

[Mierswa *et al.*, 2003b] Ingo Mierswa, Ralf Klinkenberg, Simon Fischer, and Oliver Ritthoff. 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, August 2003. ISSN 1433-3325. Long version of [Mierswa et al., 2003a]. http://yale.cs.uni-dortmund.de/.

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

[Morik and Rüping, 2002a] Katharina Morik and Stefan Rüping. 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, 2002.

[Morik and Rüping, 2002b] 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, Germany, 2002. Springer.

[Ritthoff *et al.*, 2001] 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, pages 84–92, University of Dortmund, Dortmund, Germany, October 2001. Technical Report 763, ISSN 0933-6192. http://yale.cs.uni-dortmund.de/.

[Rüping, 2000] Stefan Rüping. *mySVM Manual*. Artificial Intelligence Unit, Computer Science Department, University of Dortmund, Germany, 2000. http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.

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

[Sondhauß and Weihs, 2001] Ursula Sondhauß and Claus Weihs. Incorporating background knowledge for better prediction of cycle phases. In Miroslav Kubat and Katharina Morik, editors, *Workshop notes of the IJCAI-01 Workshop on* Learning from Temporal and Spatial Data, pages 38–44, Menlo Park, CA, USA, 2001. IJCAI, AAAI Press. Held in conjunction with the International Joint Conference on Artificial Intelligence (IJCAI).

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

[Theis and Weihs, 1999] Winfried Theis and Claus Weihs. 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, 1999.

[Vapnik, 1998] Vladimir N. Vapnik. *Statistical Learning Theory*. Wiley, Chichester, UK, 1998.

[Veltmann, 1997] Georg Veltmann. Einsatz eines Multiagentensystems zur Erstellung eines persönlichen Pressespiegels. Master's thesis, Computer Science Department, University of Dortmund, Germany, May 1997.

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