Approaches for Learning Classifiers of Drifting Concept

Ivan Koychev[1]
Faculty of Mathematics and Informatics, University of Sofia
5 J. Bouchier Street, Sofia 1164, BULGARIA
ivan.koychev@fmi.uni-sofia.bg

Abstract: Numerous, of the counterterrorist activities on the web have to distinguish between terror-related and the others items. In this case Machine Learning algorithms can be employed to construct classifiers from examples. Machine Learning applications often face the problem that the real-life concepts tend to change over time and some of the learned classifiers from old observations become out-of-date. This problem is known as concept drift. It seems to be doubly valid for terrorist acting on the web, because they presumably wiling to avoid to be tracked.

The paper gives a brief overview of the mechanisms that are aiming to deal with drifting concepts. Further it describes in more detail two mechanisms for detailing with drifting concepts, which are capable to adapt dynamically to changes by forgetting irrelevant data and models. The presented mechanisms are general in nature and can be an add-on to any concept learning algorithm. Results from experiments that give evidences for the effectiveness of the presented approaches are reported and discussed.

1 Introduction

Many of the problems related to the activities against terrorism on the web come to the problem of distinguish between terror-related and the others items (e.g. documents, user's behaviour profiles etc.). For this purpose a classifier needs to be built. Often these systems employ Machine Learning algorithms to construct classifiers from examples (labelled instances of the cases) (e.g. [AbCh05] etc.). While such a system is working, new examples are obtained continuously which have to be added to the training dataset. This requires the classifier to be updated regularly to take into account the new set of examples. We can expect that with the time the classification accuracy will improve, because we are using a larger training set. However, a classifier built on all previous examples can decrease its accuracy on the new data, because the real-life concepts tend to change over time. Hence, some of the old observations that are out-of-date/out-of-context have to be 'forgotten', because they rather producing noise than useful training examples. This problem is known as *concept drift* [ScG86]. A promising example is the systems that learn from observing user's profiles. These systems face the problem that users are inclined to change over time, or when the context is shifting (e.g. [MCFDZ94], [MaM00] and [Koy02]). This is even more valid for terrorist acting on the web, because the terrorist are most likely wiling to hide.

A number of context-aware 'forgetting' mechanisms have been developed to cope with this problem. They can be divided into two major types depending on whether they are able to *forget only* or whether they are also able to keep old data/knowledge and *remember* them if they seem to be useful again.

The first type of mechanisms has the advantage of being simpler and faster and in many case providing good enough solutions. A typical example is the Time Window approach. It performs well with concepts that do not change often and change gradually rather than abruptly, but it performs very poorly on frequently changing and recurring concepts.

[1] Also Senior Researcher at the Institute of Mathematics and Informatics at the Bulgarian Academy of Science

The second type of mechanisms has a clear advantage in the case of recurring concepts (e.g. caused by season changes or other transformation in context), where the relevant old data or acquired knowledge can be very helpful. However it requires larger storage space for the old information, extra time for retrieving and adapting/weighting of the old relevant episodes of data or knowledge.

This paper describes two forgetting mechanisms that belong to the first type of forgetting mechanisms: the first one provides the examples in the time window with weights, which decrease over time. Thus it simulates a forgetting that is gradual, which seems to be a more natural one. The second one adapts dynamically the size of the time window according to the current concept behaviour. The paper reports summary of the results from experiments with this two forgetting mechanisms, reported earlier in [KoL05] and [Koy07]. The aims are to compare the approaches and to explore the possibility to combine them for further improvement of the performance.

The next section gives a short overview of the related work. The compared mechanisms are described in section 3. The experiment design is introduced in section 4. The results from the experiments are reported and discussed in section 5.

2 Related Work

Different approaches have been developed to track changing (also known as shifting, drifting or evolving) concepts. Typically it is assumed that if the concept changes, then the old examples become irrelevant to the current period. The concept descriptions are learned from a collection of most recent examples called a time window. For example, Mitchell et al. [MCFDZ24] developed a software assistant for scheduling meetings, which employs machine learning to acquire assumptions about individual habits of arranging meetings, uses a time window to adapt faster to the changing preferences of the user. Widmer and Kubat [WiK96] developed the first approach that uses adaptive window size. The algorithm monitors the learning process and if the performance drops under a predefined threshold it uses heuristics to adapt the time window size dynamically. Maloof and Michalski [MaM00] have developed a method for selecting training examples for a partial memory learning system. The method uses a time-based function to provide each instance with an age. Examples that are older than a certain age are removed from the partial memory. Delany et al. [DCTC04] employ a case base editing approach that removes noise cases (i.e. the cases that contribute to the classification incorrectly are removed) to deal with concept drift in a spam filtering system. The approach is very promising, but is applicable for lazy learning algorithms only. To manage the problems with gradual concept drift and noisy data, the approach in [LVB04] suggests the use of three windows: a small (with fixed size), a medium and a large (dynamically adapted by simple heuristics).

The above approaches totally forget the examples that are outside the given window, or older than a certain age. The examples which remain in the partial memory are equally important for the learning algorithms. This is an abrupt forgetting of old examples, which probably does not reflect their rather gradual ageing. To deal with this problem it was suggested to weight training examples in the time window according to their appearance over time [Koy00]. These weights make recent examples more important than older ones, essentially causing the system to gradually forget old examples. This approach has been explored further in [Kuk03], [DuN04] and [Kli04].

Some systems use different approaches to avoid loss of useful knowledge learned from old examples. The CAP system [MCFDZ94] keeps old rules as long as they are competitive with the new ones. The FLORA system [WiK96], also maintains a store of concept descriptions relevant to previous contexts. When the learner suspects a context change, it will examine the potential of previously stored descriptions to provide better classification. The COPL approach [Koy02] employs a two-level schema to deal with changing and recurring concepts. On the first level the system learns a classifier from a small set of the most recent examples. The learned classifier is accurate enough to be able to distinguish the past episodes that are relevant to the current context. Then the algorithm constructs a new training set, ‘remembers’ relevant examples and ‘forgets’ irrelevant ones. As we explained in the introduction, the approaches explored in this paper, does not assume that old examples or models can be retrieved.

Widmer [Wid97] assumes that the domain provides explicit clues to the current context (e.g. attributes with characteristic values). A two-level learning algorithm is presented that effectively adjusts to changing contexts by trying to detect (via meta-learning) contextual clues and using this information to focus the learning process. Another two-level learning algorithm assumes that concepts are likely to be stable for some period of time [HaS98]. This approach uses contextual clustering to detect stable concepts and to extract hidden context. The mechanisms studied in this paper do not assume that the domain provides some clues that can be discovered using a meta-learning level. They rather aim to get the best performance using a single learning level.

An adaptive boosting method based on dynamic sample-weighting is presented in [ChZ04]. This approach uses statistical hypothesis testing to detect concept changes. Gama et al., [GMCR04] also use a hypothesis testing procedure, similar to that used in control charts, to detect the concept drift, calculating on all of the data so far. The mechanism gives a warning at 2 standard deviations (approximately 95%) and then takes action at 3 standard deviations (approximately 99.7%). If the action level is reached then the start of the window is reset to the point at which the warning level was reached. However, for this mechanism, it can take quite a long time to react to changes and the examples that belong to the old concept are not always completely useless, especially when the concept drift is rather gradual.

To adapt the size of the window according to current changes in the concept, the approach presented in [Kli04] uses a naïve optimisation approach, which tries all possible window sizes and selects the one with the smallest error rate. This work provides interesting results from experiments with different forgetting mechanisms applied for Support Vector Machines classifier, using a textual data corpus.

To detect concept changes the TWO approach [KoL05] uses a statistical test on selection population from the time window, which excludes its’ beginning and end. If a concept drift is detected an efficient optimisation algorithm is employed to detect the optimal window size. This approach is explained in more details in the next section and studied in the conducted experiments.

3 Two Approaches for Tracking Drifting Concepts

Let us consider a sequence of examples. Each example is classified according to underlying criteria into one of a set of classes, which forms the training set. The task is to learn a classifier that can be used to classify the next examples in the sequence. However, the underlying criteria can subsequently change and the same example can be classified differently according to the time of its appearance, i.e. a concept drift takes place. As we discussed above to deal with this problem machine learning systems often use a time window i.e. the classifier is not learned on all examples, but only on a subset of recent examples. The next section describes forgetting mechanisms that further develop this idea.

3.1 Gradual Forgetting

This section describes a gradual forgetting mechanism earlier introduced in [Koy00]. Just as the time window approach it assumes that when a concept tends to drift the newest observations better represent the current concept. Additionally, it aims to make the forgetting of old examples gradual. Let us define a gradual forgetting function $w = f(t)$, which provides weights for each instance according to its location in the course of time. The weights assign higher importance value to the recent examples. Earlier, Widmer [Wid97] suggests an "exemplar weighting" mechanism, which is used for the IBL algorithm in METAL(IB), however it is not exploited for NBC in METAL(B). The researcher in area of boosting also faced the need of weighting examples. There are two ways, in which boosting employs the weights. The first one is known as boosting *by sampling* - the examples are drawn with replacement from the training set with a probability proportional to their weights. However, this approach requires a pre-processing stage where a new set of training examples should be generated. The better the sampling approximates the weights, the larger is the new training set. The second method, boosting *by weighting* is used with learning algorithms that accept weighted training examples directly. In this case the weight is constrained as follows:

$$w_i \geq 0 \text{ and } \sum_{i=1}^{n} w'_i = 1 \tag{1}$$

where, n is the size of the training set. Most of the basic learning algorithms are designed to treat all examples as equally important. For kNN it can be easily implemented by multiplying the calculated distances with the weights of the examples. For other algorithms it does not look so straightforward. When there are no weights (i.e. all weights are the same) the formula (1) will provide weights $\forall w_i = \frac{1}{n}$. However, as the weights are utilized by multiplication, it seems to be better if we have $\forall w_i = 1$ in this boundary case. If we substitute that $w_i = nw'_i$ then the constraints in equation (1) will be transformed as follows:

$$w_i \geq 0 \text{ and } \frac{\sum_{i=1}^{n} w_i}{n} = 1 \tag{2}$$

Weights that obey the constraints (2), can be easily used in almost any learning algorithm requiring minor changes in the code i.e. every time when the algorithm counts an example it should be multiplied by its weight.

Various functions that model the process of forgetting and satisfy constraints (2) can be defined. For example, the following linear gradual forgetting function has been defined:

$$w_i = -\frac{2k}{n-1}(i-1) + 1 + k \tag{3}$$

where: i is a counter of observations starting from the most recent observation and it goes back in time (as the function is forgetting $w_i \geq w_{i+1}$); $k \in [0,1]$ is a parameter that represents the percent of decreasing the weight of the first and respectively increasing the weight of the last observation in comparison to the average. By varying *k* the slope of the forgetting function can be adjusted to reach better predictive accuracy.

The presented gradual forgetting mechanism is naturally integrated into a time window, by weighting the examples in it (i.e. $n = l$, where l is the size of the time window). The system will forget gradually inside the window. When $k = 0$ then all weights will be equal to 1, which means that we will have a "standard" time window. When k is approaching 1 then the weights of the examples in the end on the window approach 0 and there is not an abrupt forgetting after the window's end.

3.2 Time Window Optimisation

This section presents an approach that learns an up-to-date classifier on drifting concept by dynamic optimisation of the time window size to gain maximum accuracy of prediction [KoL05].

In case of dynamical adaptation of window size there are two important questions that have to be addressed in case of concept drift. The first one is *how to detect a change in the concept*? We are assuming that there is no background information about the process and we use the decrease in predictive accuracy of the learned classifier as an indicator of changes in the concept. Usually the developed detection mechanism uses a predefined threshold tailored for the particular dataset. However the underlying concept can change with different speeds and extend. To detect the changes the approach uses a statistical hypothesis test, which adapts the thresholds according to the recently observed concept deviation. The second important question is *how to adapt if a change is detected?* Other approaches use heuristics to decrease the size of the time window when changes in the concept are detected (e.g. [DuN04] and [WiK96]). This approach employs a fast 1-D optimisation to get the optimal size of the time window if a concept drift is detected. The next two subsections describe the algorithm in more details.

3.2.1 Detecting the Concept Drift

To detect concept changes the approach monitors the performance of the algorithm. For the presentation below we choose to observe the classification accuracy as a measure of the algorithm performance, but other measures, such as error rate or precision could have been used. The approach process the sequence of examples on small episodes (a.k.a. batches). On each step, a new classifier is learned from the current time window then the average accuracy of this classifier is calculated on the next batch of examples [MCFDZ94]. Then the presented approach suggests using a statistical test to check whether the accuracy of classification has significantly decreased compared with its historical level. If the average prediction accuracy of the last batch is significantly less than the average accuracy for the population of batches defined on the current time window, then a concept drift can be assumed.

The significance test is done using a population from the time window that does not include the first one or a few batches from the beginning of the time window, because the predictive accuracy can be low caused by a previous concept drift. Also one or a few most recent batches are not included in the test window, because if the drift is gradual the accuracy will be dropping slowly. Such a test that uses a test population from the core of the time window works well for both abrupt and gradual drift.

The confidence level for the significance test should be sensitive enough to discover concept drift as soon as possible, but not to mistake noise for changes in the concept. The experience from the conducted experiments shows that the “standard” confidence level of 95% works very well in all experiments. This drift detection level is rather sensitive and it assists the algorithm to detect the drift earlier. If a false concept drift alarm is triggered, it will activate the window optimising mechanism, but in practice, this only results in an insignificant decrease in the time window size.

This mechanism works as follows: ***If*** *concept drift is detected* ***then*** *the optimisation of the size of the time window is performed (see the next section)* ***otherwise,*** *the time window size is increased to include the new examples.*

3.2.2 Optimising the Time Window Size

In general, if the concept is stable, the bigger the training set is (the time window), the more accurately classifier can be learned. However when the concept is changing, a big window will probably contain a lot of old examples, which will result in a decrease of the classification accuracy. Hence, the window size should be decreased to exclude the out-of-date examples and in this way to learn a more accurate classifier. But if the size of the window becomes too small, it will also lead to a decrease in accuracy. The shape of curve that demonstrates the relationship between the size of the time window and the accuracy of the classification is shown in Figure 1.

To adapt the size of the window according to current changes in the concept, the presented mechanism suggests using the Golden Section algorithm for one-dimensional optimization. The algorithm looks for an optimal solution in a closed and bounded interval $[a,b]$ - in our case the possible window sizes $X = [x_{\min}, x_c]$, where $x_{\min}$ is a predefined minimum size of the window and x_c is the current size of the time window. It assumes that the function $f(x)$ is unimodal on X (i.e. there is only one max x^*) and it is strictly increasing on $(x_{\min}, x^*)$ and strictly decreasing on (x^*, x_c), which is the shape that can be seen in Figure 1. In our case the function $f(x)$ calculates the classification accuracy of the learned model using a time window with size x.

The basic idea of this algorithm is to minimize the number of function evaluations by trapping the optimum solution in a set of nested intervals. On each step the algorithm uses the golden section ($\tau = 0.618$) to split the interval into three subintervals, as shown in Figure 1, where $l = b - \tau(b-a)$ and $r = a + \tau(b-a)$. If $f(l) > f(r)$ then the new interval chosen for the next step is $[a,r]$ else $[l,b]$. The length of the interval for the next iteration is $\tau(b-a)$. Those iterations continue until the interval containing the maximum reaches a predefined minimum size. x^* is taken to lie at the centre of the final interval.

The Golden Section algorithm is a very efficient way to trap the x^* that optimizes the function $f(x)$. After n iterations, the interval is reduced to 0.618^{n} times its original size. For example if $n = 10$, less than 1% of the original interval remains. Note that, due to the properties of the golden section, each iteration requires only one new function evaluation.

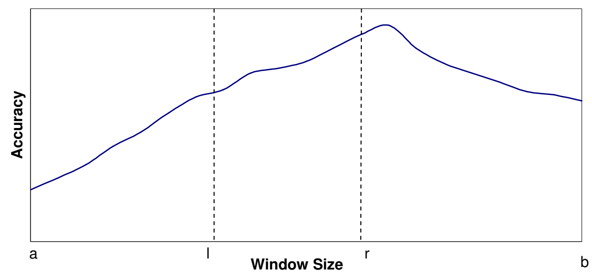

Figure 1 A sample shape of the correlation between the window size and accuracy of the learned classifier

In conclusion, if we can assume that the classification accuracy in relation to the time window is a unimodal function then the golden section algorithm can be used as an efficient way to find the optimal size of the time window. It is possible to find datasets for which the unimodal assumption is not true – e.g. when the concept changes very often and abruptly. In such cases, we can use other optimization methods that do not assume a unimodal distribution, however they are much more expensive in time. The trade-off that we have to take into consideration is to accept that we can occasionally be trapped in a local maximum, but have a fast optimization; or find a global maximum, but have significantly slower optimization.

4 Experiment Design

All experiments were designed to run iteratively, in this way simulating the process of mechanisms' utilisation. For this reason the data streams were chunked on episodes/batched. On each iteration, a concept description is learned from the examples in the current time window. Then the learned classifier is tested on the next batch (see Figure 2).

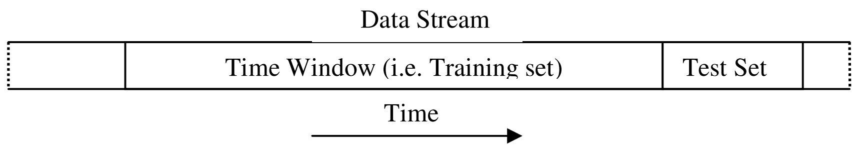

Figure 2 Iterations' cross-validation design

To facilitate the presentation below the basic forgetting mechanisms are abbreviated as follows:

FTW - Fixed-size Time Window
GF - Gradual Forgetting
TWO - Time Window Optimisation

For each data set, the window size for the FTW was chosen to approximate the average time window size obtained in the experiments with the TWO mechanism on the same dataset. This is extra help for the FTW that would not be available in a real situation where the forthcoming sequence of events is unknown. The aim here is to allow the FTW approach to show its best performance.

Each hypothesis was tested on four experimental datasets; using three basing Machine Learning algorithms. The rest of this section describes them in more details. The results from the experiments are reported and discussed in the next subsection.

4.1 Datasets

This subsection explains how the data streams used in the experiments were generated.

4.1.1 STAGGER problem

The first experiments were conducted with an artificial learning problem that was defined and used by Schlimmer and Granger [ScG86] for testing STAGGER, probably the first concept drift tracking system. Much of the subsequent work dedicated to this problem used this dataset for testing purposes (e.g. [Aha91], [DuN04], [HaS98], [Koy00], [Koy02], [MaM00], [SpB04] and [WiK96]).

The STAGGER problem is defined as follows: The instance space of a simple blocks world is described by three attributes: ***size*** = *{small, medium, large}*, ***color*** = *{red, green, blue}*, and ***shape*** = *{square, circular, triangular}*. There is a sequence of three target concepts: (1) - ***size*** = *small and* ***color*** = *red*; (2) - ***color*** = *green or* ***shape*** = *circular;* and (3) - ***size*** = *(medium or large)*. 120 training instances are generated randomly and classified according to the current concept. The underlying concept is forced to change after every 40 training examples in the sequence: (1)-(2)-(3). The setup of the experiments with the STAGGER dataset was done exactly in the same way as in other similar works. The retraining step is 1, however there is a test set with size 100, generated randomly and classified according to the current concept. This differs from the other experiments where the retraining step and the test set are the same - a batch. The size of the FTM is set up to 25, which approximates the average size of the optimised windows.

4.1.2 German Credit Dataset

This subsection presents the results from the experiments conducted with the German credit dataset, from the UCI Machine Learning Repository[2]. The dataset contains 1000 Instances of bank credit data which are described by 20 attributes. The examples are classified in two classes as either "*good*" or "*bad*". To simulate hidden changes in the context the dataset was sorted by an attribute then this attribute was removed from the dataset for the experiments. Using an attribute to sort the data set and in this way simulate changing context is a commonly used approach to set up experiments to study concept drift. Two sorted datasets were created using the German credit dataset: The first one was sorted by a continuous attribute: "*age",* which would produce a gradual drift of the "*class"* concept. The second one was sorted by the attribute "*checking_status",* which has three discrete values. We aimed in this way to create abrupt changes of the "*class"* concept. Thus using this source dataset, two data streams are generated for the experiments. The datasets were divided into a sequence of batches, each of them containing 10 examples. The size of the FTM is set to 200, which approximates the average size of the optimised windows.

4.1.3 Spam dataset

Experiments have also been conducted with the Spam dataset from the UCI Machine Learning Repository. Spam is an unsolicited email message. The dataset consists of 4601 instances, 1813 (39.4%) of which are spam messages. The dataset is represented by 54 attributes that represent the occurrence of a pre-selected set of words in each of the documents plus three attributes representing the number of capital letters in the e-mail. To simulate he changing hidden context the examples in the dataset are sorted according to the *"capital_run_length_total",* which is the total number of capital letters in the e-mail. This attribute and the related two attributes "*capital_run_length_average"* and "*capital_run_length_longest"* are removed from the dataset, because they can provide explicit clues for the concept changes. The sorted dataset was divided into a sequence of batches with a length of 10 examples each. For this dataset the fixed window size was set to 400 - an approximation of the average window size for this dataset used by the TWO mechanism.

4.2 Machine Learning Algorithms

The experiments were conducted using three basic machine learning algorithms:

- k Nearest Neighbours (kNN) - also known as Instance Based Learning (IBL) [Aha91]. k=3 was the default setting for the experiments reported below except for experiments with STAGGER dataset, where k=1 was chosen, because it produces a more accurate classification than k=3;
- Induction of Decision Trees (ID3) [Qui86] (using an attribute selection criteria based on the χ^2 statistics);
- Naïve Bayesian Classifier (NBC) [Mit97].

[2] http://www.ics.uci.edu/~mlearn/MLRepository.html

5 Experimental Results and Discussion

This section presents a summary of the results reported in the [KoL05] and [Koy07].

The results from the conducted experiments are presented in Tables 1, 2 and 3 below. In all these tables, rows present the used learning algorithms: kNN, ID3 and NBC. The columns present the results from the experiments with different datasets. For a more compact presentation in the tables the dataset are referred with numbers as follows:

1 – STAGGER dataset

2 – German dataset - sorted by the attribute *"checking_status"*

3 – German dataset - sorted by the attribute *"age"*

4 – Spam dataset - sorted by the attribute *"capital_run_length_total"*

We used the paired t-tests with 95% confidence level to see whether there is significant difference in the accuracy of learned classifiers. The pairs are formed by comparing the algorithms' accuracies on the same iteration. In the tables below, we are reporting the results from the experiments testing the above formulated hypothesis. The cells represent the results from the significance test comparing the results form the runs using the different forgetting mechanisms. The used basic learner is presented in the row and the used dataset stated in the column. The notation is as follows:

✓ - denotes that the second algorithm performs significantly better then the first one;

✗ - denotes that the first algorithm performs significantly better then the second one;

÷ - denotes that there is no difference in the performance of the compared algorithms.

For example: the sign ✓ in the first column and first row in Table 1 denotes that the experiments conducted using kNN as a basic algorithm and using the STAGGER dataset (corresponds to the column 1), shows that FTW with GF is significantly better than the plain FTW, measured in accuracy of classification.

algorithm	***dataset***	*1*	*2*	*3*	*4*
kNN		✓	✓	✓	✗
ID3		✓	✓	✓	÷
NBC		✓	✓	✓	÷

Table 1 Comparison of FTW versus FTW+GF forgetting mechanisms

The results from the experiments presented in Table 1 show that for three of the four datasets, for any of the Gradual Forgetting is able to improve significantly the average classification accuracy in comparison of the Fixed Time Window, however the Gradual Forgetting fails to improve the accuracy on the fourth data set. In comparison with other dataset, the main difference of this dataset is its sparseness. Therefore, we can assume that probably this is the reason for the lack of improvement. Perhaps, using feature selection and adapted for text similarity measure for kNN, will diminish this problem. Further studies are needed to find the correct answer to these questions.

algorithm	***dataset***	*1*	*2*	*3*	*4*
kNN		✓	✓	✓	✓
ID3		✓	✓	✓	✓
NBC		✓	✓	✓	✓

Table 2. Comparison of FTW versus TWO forgetting mechanisms

The experiment results presented in Table 2 provided very strong evidence that the Time Window Optimisation mechanism is able to improve the prediction accuracy significantly in comparison with Fixed-size Time Window.

algorithm	*dataset*	*1*	*2*	*3*	*4*
kNN		✓	÷	✓	✓
ID3		✓	✓	✓	✓
NBC		✓	✓	✓	✓

Table 3 Comparison of FTW+GF versus TWO forgetting mechanisms

The results presented in Table 3 shows that in all experiments the Time Window Optimisation mechanisms achieve significantly better classification accuracy than Gradual Forgetting, except one where there is no significant difference in performance.

algorithm	*dataset*	*1*	*2*	*3*	*4*
kNN		÷	✓	✓	×
ID3		✓	✓	÷	×
NBC		×	÷	✓	÷

Table 4 Comparison of TWO versus TWO+GF forgetting mechanisms

Table 4 shows the result from the experiment testing the hypothesis that applying Gradual Forgetting together with TWO can improves the prediction accuracy in comparison to pure TWO, i.e. both mechanisms are having synergetic effect. The results from the experiments do not provide enough evidences in support of this hypothesis.

Figure 4 shows on the same diagram: the classification accuracy in one of the experiments with TWO mechanism (the thin line) and the size of the optimised time window (the thick line) on each step. It can be seen that a drop in the accuracy normally leads to a decrease of the time widow size. However, a sudden decrease in the classification accuracy does not always indicate a concept drift, it can be caused by noise in the data stream. It can be seen that TWO mechanism is very robust to noise, merely decreasing the window size insignificantly, e.g. see the arrow 1 on Figure 4. However, it remains sensitive enough to detect genuine concept drifts that decrease the accuracy by a relatively small value – e.g. see the arrow 2 on Figure 4. The detection mechanism

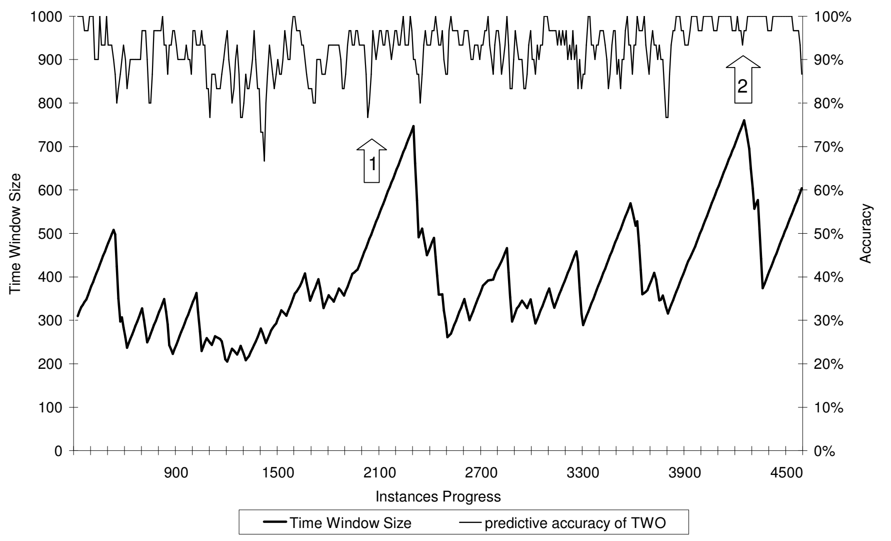

Fig. 1. The relationship between the accuracy of prediction measured on the test set and the time window size.

flags both real and false concept drifts, but the window size optimizer responds very differently to the two possibilities.

The STAGGER problem was used by a number of other systems that deal with concept drift. The results from the experiments on this dataset using the presented forgetting mechanisms (GF and TWO) where compared to similar approves (i.e. [Aha91] [WiK96] [MaM00] and [Koy02]) in KoL05. In summary: Gradual Forgetting is simple approach that usually performs compatibly to other approached even in some cases significantly outperforms them mainly when concepts are drifting; Time Window Optimisation significantly outperform the other forgetting mechanisms on drifting concepts.

6 Conclusion

The paper presents an experimental study of two forgetting mechanism for dealing with the concept drift problem. Both are general in nature and can be added to any relevant algorithm. Experiments are conducted with three learning algorithms using four datasets. The results from the experiments show that: applying Gradual Forgetting inside a fixed-size Time Window usually leads to a significant improvement of the classification accuracy on drifting concept; when comparing both forgetting mechanisms the advantage clearly is on the side of the self-adapting algorithm - Time Window Optimisation; finally, applying both mechanisms together usually does not lead to an improvement in the classification accuracy.

Further controlled experiments using common patterns from the space of different type of drift (i.e. frequent – rare; abrupt - gradual; slow – fast; permanent – temporary; etc.) will provide clearer ideas for which pattern, what kind of forgetting mechanism is most appropriate.

The importance of taking in to account the concept drift in case of the counter-terrorists mining of the web springs from the nature of the typical task: a limited data history and often changing concepts. The importance of the problem was also raised by other participants in their formal talks and informal discussion.

References

[AbCh05] Abbasi A. and Chen H., "Applying Authorship Analysis to Extremist-Group Web Forum Messages", Published by the IEEE Computer Society, september/october 2005, Pages 67-75

[Aha91] Aha, D., D. Kibler and M. Albert. Instance-Based Learning Algorithms. Machine Learning 6, (1991) 37-66

[ChZ04] Chu, F. and Zaniolo, C.: Fast and light boosting for adaptive mining of data streams. In: Proc. of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining. Lecture Notes in Computer Science, Vol.3056, Springer-Verlag, (2004) 282-292

[DCTC04] Delany, SJ., P. Cunningham, A. Tsymbal and L. Coyle. A Case-Based Technique for Tracking Concept Drift in Spam Filtering. In: Macintosh, A., Ellis, R. & Allen T. (eds.) Applications and Innovations in Intelligent Systems XII, Proceedings of AI2004, Lecture Notes in Computer Science, Springer (2004) 3-16

[DuN04] Ducatel, G. and A. Nürnberger. iArchive: An Assistant To Help Users Personalise Search Engines. In Chapter 2. User Profiles in Information Retrieva, Enhancing the Power of the Internet, Series: Studies in Fuzziness and Soft Computing , Vol. 139, Nikravesh, M.; Azvine, B.; Yager, R.; Zadeh, L.A. (Eds.) 2004, VIII, 406 p., Stinger-Verlag.

[GMCR04] Gama, J., P Medas, G. Castillo and P. Rodrigues. Learning with Drift Detection. In: Ana, C., Bazzan, S. and Labidi (Eds.): Proceedings of the 17th Brazilian Symposium on Artificial Intelligence. Lecture Notes in Computer Science, Vol. 3171, Springer, (2004) 286-295

[HaS98] Harries, M. and C. Sammut. Extracting Hidden Context. Machine Learning 32 (1998) 101-126

[Kli04] Klinkenberg, R. Learning Drifting Concepts: Example Selection vs. Example Weighting. In Intelligent Data Analysis, Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift, Vol. 8, No. 3, (2004) 281-300

[KoL05] Koychev, I., Lothian R.: Tracking Drifting Concepts by Time Window Optimisation. In: Research and Development in Intelligent Systems XXII Proc. of AI-2005, the 25-th SGAI Int. Conference on Innovative Techniques and Applications of Artificial Intelligence. Bramer, Max; Coenen, Frans; Allen, Tony (Eds.), Springer-Verlag, London p.46-59

[KoS00] Koychev, I. and I. Schwab. Adaptation to Drifting User's Interests. In proc. of ECML2000 Workshop: Machine Learning in New Information Age, Barcelona, Spain, p. 39-46

[Koy00] Koychev, I. Gradual Forgetting for Adaptation to Concept Drift. Proceedings of ECAI 2000 Workshop on Current Issues in Spatio-Temporal Reasoning, Berlin, (2000) 101-107

[Koy02] Koychev, I. Tracking Changing User Interests through Prior-Learning of Context. In: de Bra, P., Brusilovsky, P., Conejo, R. (eds.): Adaptive Hypermedia and Adaptive Web Based Systems. Lecture Notes in Computer Science, Vol. 2347, Springer-Verlag (2002) 223-232

[Koy07] Koychev I. Experiments with two approaches for tracking drifting concepts. Serdica Journal of Computing 1 (2007), 27-44.

[Kuk03] Kukar, M. Drifting Concepts as Hidden Factors in Clinical Studies. In Dojat, D., Elpida T. Keravnou, Pedro Barahona (Eds.): Proceedings of 9th Conference on Artificial Intelligence in Medicine in Europe, AIME 2003, Protaras, Cyprus, October 18-22, 2003, Lecture Notes in Computer Science, Vol. 2780, Springer-Verlag (2003) 355-364

[LVB04] Lazarescu, M., Venkatesh, S. and Bui, H.: Using Multiple Windows to Track Concept Drift. In the Intelligent Data Analysis Journal, Vol 8 (1), (2004) 29-59

[MaM00] Maloof, M. and R. Michalski. Selecting examples for partial memory learning. Machine Learning 41 (2000) 27-52

[MCFDZ94] Mitchell, T., R. Caruana, D. Freitag, J. McDermott and D. Zabowski. Experience with a Learning Personal Assistant. Communications of the ACM 37(7) (1994) 81-91

[Mit97] Mitchell T. Machine Learning. McGraw-Hill (1997)

[Qui86] Quinlan, R. Induction of Decision Trees. Machine Learning 1 (1986) 81-106

[ScG86] Schlimmer, J. and R. Granger. Incremental Learning from Noisy Data. Machine Learning 3, (1986), 317-357

[SpB04] Spiliopoulou M., S. Baron. Temporal Evolution and Local Patterns. In Morik, K., Boulicaut, JF., and Siebes A. (Eds.): Local Pattern Detection, International Seminar, Dagstuhl Castle, Germany, April 12-16, 2004, Revised Selected Papers. Lecture Notes in Computer Science 3539, Springer 2005

[Wid97] Widmer, G. Tracking Changes through Meta-Learning. Machine Learning 27 (1997) 256-286

[WiK96] Widmer, G. and M. Kubat. Learning in the presence of concept drift and hidden contexts: Machine Learning 23 (1996) 69-101