Tracking Drifting Concepts by Time Window Optimisation

Ivan Koychev[1] **Robert Lothian**[2]

[1] Department of Information Research
Institute of Mathematics and Informatics, Bulgarian Academy of Science,
Acad. G. Bonchev Street, bl.8, Sofia -1113, Bulgaria
ikoychev@math.bas.bg
[2] School of Computing, The Robert Gordon University
St Andrew Street, Aberdeen AB25 1HG, UK
rml@comp.rgu.ac.uk

Abstract. This paper addresses the task of learning concept descriptions from streams of data. As new data are obtained the concept description has to be updated regularly to include the new data. In this case we can face the problem that the concept changes over time. Hence the old data become irrelevant to the current concept and have to be removed from the training dataset. This problem is known in the area of machine learning as concept drift. We develop a mechanism that tracks changing concepts using an adaptive time window. The method uses a significance test to detect concept drift and then optimizes the size of the time window, aiming to maximise the classification accuracy on recent data. The method presented is general in nature and can be used with any learning algorithm. The method is tested with three standard learning algorithms (kNN, ID3 and NBC). Three datasets have been used in these experiments. The experimental results provide evidence that the suggested forgetting mechanism is able significantly to improve predictive accuracy on changing concepts.

1 Introduction

Many machine learning applications employ algorithms for learning concept descriptions. Usually, as time passes, new examples are obtained and added to the training dataset. Then the concept description is updated to take into account the new examples. These applications often face the problem that real life concepts tend to change over time, i.e. a concept description learned on all previous examples is no longer up-to-date. Hence, some of the old observations that are out-of-date have to be 'forgotten'. This problem is known as *concept drift* [16]. A prominent example of a system that should adapt to changing concepts is a system that helps users to find pieces of information in which they are likely to be interested. These systems use machine learning methods to acquire from observations a model of user's interests. However, user's interests and preferences are inclined to change over time (e.g. [7],

[12] and [13]). Such systems are usually provided with mechanisms that are able to track changing user interests.

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*? If there is no background information about the process, we can 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 (e.g. [18]). However the underlying concept can change with different speeds i.e. some times it can be *abrupt* other times it can be rather *gradual*. The detection mechanism often has difficulty in detecting both types of changes i.e. if the threshold is very sensitive it can mistake noise for concept drift or if it is not sensitive enough it can take too long to discover a gradual drift. Recently some authors suggested the use of a statistical hypothesis test to detect the changes (e.g. [4] and [9]).

The second important question is *how to adapt if a change is detected?* In some applications a fixed size time window optimised for the particular application is used (e.g. [13]). This solution is fast and easy to implement, but it requires a preliminary investigation of the domain to select the window size. Moreover if the type and frequency of the changes in the concept are unpredictable it can lead to a decrease in the classification accuracy. Other approaches use heuristics to decrease the size of the time window when changes in the concept are detected (e.g. [4] and [18]).

The next section gives a short overview of the related work. A novel mechanism that addresses both questions for dealing with the concept drift problem is presented in section 3. It uses a statistical significance test to detect concept drift. Then it employs an efficient optimisation algorithm to adapt the size of the time window. Experiments with one artificial and two real datasets are reported in section 4.

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 set of recent examples called a time window. For example, Mitchell et al. [13] 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 [18] 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 [12] 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. [3] 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 [11] suggests the use of three windows: a small (with fixed size), a medium and a large (dynamically adapted by simple heuristics). The approach presented in this paper also addresses those problems, by a carefully designed statistical test and uses an efficient optimal algorithm instead of heuristics to adapt dynamically the time window size.

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 aging. To deal with this problem it was suggested to weight training examples in the time window according to their appearance over time [6]. 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 [8] and [9]. The mechanism presented in this paper is also based on the time window idea, but it seeks to improve the performance by dynamically optimising the time window size. It seems that gradual forgetting can compliment quite well the time window approach, but we choose to focus on exploring the pure approach in this paper.

Some systems use different approaches to avoid loss of useful knowledge learned from old examples. The CAP system [13] keeps old rules as long as they are competitive with the new ones. The architecture of the FLORA system [18], assumes that the learner 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 approach presented in [7] employs a two-level schema to deal with drifting and recurring concepts. On the first level the system learns a classifier from the most recent observations assuming that it is able to provide description of the current context. 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. The approach presented in this paper, does not assume that old examples or models can be retrieved, because this can be very time consuming and memory intensive.

Widmer [17] 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 [5]. This approach uses batch learning and contextual clustering to detect stable concepts and to extract hidden context. The mechanism in this paper does not assume that the domain provides some clues that can be discovered using a meta-learning level. It rather aims to get the best performance using a single learning level.

An adaptive boosting method based on dynamic sample-weighting is presented in [9]. The approach uses statistical hypothesis testing to detect concept changes. Gama

et al., [4] 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. The approach presented in this paper also uses a statistical test to detect concept changes, but some limitations of the plain test are addressed by a more careful selection of the test population. If a concept description is detected then the mechanism uses a fast optimisation algorithm, to find out the optimal size of the window that achieves the maximum accuracy of classification.

3 A Scheme 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. This section introduces an approach for learning up-to-date descriptions of drifting concept based on this idea. It addresses the two major tasks involved in dealing with drifting concepts: by suggesting an effective mechanism for detecting the concept drift and if a drift is detected optimises the size of the time window to gain maximum accuracy of prediction.

3.1 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 presented 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 [13]. Then the presented approach suggests using a statistical test to check whether the accuracy of classification has significantly decreased compared with its historical level. We assume that the concept has changed if the current accuracy exceeds the appropriate test level for the normal distribution at the required confidence level. To be precise, 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.

As the performance level of the algorithm can vary for the different concepts and the noise level can also change over the time, we must carefully select the test population for the current time window. We can use the whole window as in [9], but the predictive accuracy at the beginning of the time window can be low because of previous concept drift. For example, you can see on Figure 2 in section Experiments, how the classification accuracy slowly increases after a concept drift. Therefore we suggest that the test is done on a sub-window that does not include the first few batches from the beginning of the time window. Clearly this mechanism will work well when the concept is shifting i.e. the accuracy is dropping abruptly. In the case where the changes in the concept are gradual the mechanism should work if the significance test is done on a relatively old population, i.e. one or a few most recent batches are not included in the test window. Such a test that uses a test population from the core of the time window will work well for both abrupt and gradual drift.

From the central limit theorem, it follows that if we want to be confident about the required test level, the test should be supplied to batches of at least 30 examples. If it appears that the current batch size is less than 30 then the algorithm can easily resize the batches to satisfy this guidance. In cases where the data set is expected to be noisier, then a larger batch size is recommended, because this will smooth changes caused by noise.

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.

The presented 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 Optimising the Time Window Size

In general, if the concept is stable, the bigger the training set is (the time window), the more accurately the concept description 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, Widmer and Kubat [18] pioneer the use of heuristics. However, it would be ideal if we

were able to find the optimal size of the window to ensure the best classification accuracy. The approach presented in [8] tries all possible window sizes and selects the one with the smallest error rate. This brute force optimization is, of course, inefficient.

The presented mechanism suggests using the Golden Section algorithm for one-dimensional optimization [2]. 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.

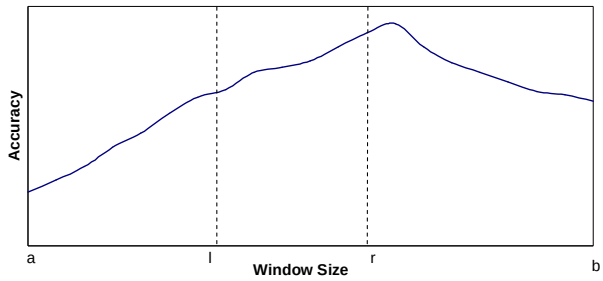

Fig. 1. A sample shape of the correlation between the window size and accuracy of the learned classifier

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 Experiments

The aim of the experiments reported in this section is to explore whether the present forgetting mechanism is able to improve the performance of different learning algorithms on drifting concepts.

All experiments were designed to follow the natural scenario of using such mechanisms [13]. For this reason the data streams were chunked on episodes/batched. The algorithm was run on this data set iteratively - 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.

The experiments were conducted with three popular learning algorithms:

- k Nearest Neighbours (kNN) - also known as Instance Based Learning (IBL) [1]. 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) [15] (using an attribute selection criteria based on the χ^2 statistics);
- Naïve Bayesian Classifier (NBC) [14].

The first experiments were conducted with an artificial learning problem that was defined and used by Schlimmer and Granger [16] 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. [1], [4], [5], [6], [7], [12], [17] and [18]). This allows comparison of our approach with similar approaches on this data set. Those results are presented in the next subsection. Experiments also were conducted with two datasets from the UCI machine learning repository[1], which are presented in subsections 4.2 and 4.3 below.

The results from the conducted experiments are presented in Tables 1, 3, 4 and 5 below. In all these tables, rows present the used learning algorithms: kNN, ID3 and NBC. The first column shows the predictive accuracy of the algorithms using Full Memory (FM) learning – all data available up to the current moment are used for learning the concept. The second column presents the results from the experiments with Fixed-size Time Window (FTW). The third column shows the results from the experiments with the algorithms using this paper's Time Window Optimisation (TWO) mechanism. 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

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

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.

We used the paired t-tests with 95% confidence level to see whether the presented approach significantly changes the accuracy of learned classifiers. The pairs are formed by comparing the algorithms' accuracies on the same iteration. In the tables below, reporting the results from the experiments, the sign * denotes that the TWO approach achieves a significantly better classification accuracy than the FM and the sign ^ - that TWO is significantly better than the FTW approach.

4.1 STAGGER problem

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 us 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.

Memory: / Algorithm:	FM	FTW	TWO
kNN	63.03	80.47	86.56*^
ID3	69.05	83.73	89.03*^
NBC	69.97	82.99	90.26*^

Table 1. The improvement of the classification accuracy when the TWO mechanism is applied to the STAGGER dataset

Table 1 shows the results from the experiments with this dataset. In this dataset we have two abrupt changes in the underlying concept and the fixed size time window is able to improve the classification accuracy significantly compared with the full memory. The TWO mechanism additionally improves the classification accuracy significantly compared to FTW.

Figure 2 shows a plot of the results from the experiments with NBC, which illustrate the behaviour of the three mechanisms. It can be seen from the chart that when the algorithm uses the TWO mechanism it adapts faster to the changes.

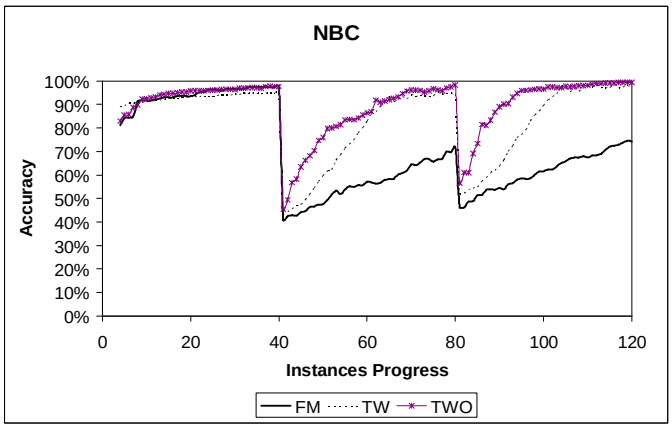

Fig. 2. Classification accuracy of the NBC using Full Memory, Time Window and Time Window Optimisation.

Figure 3 shows the average size of the time window for each step in one of the experiments with NBC-TWO. The experiments with other algorithms produce very similar graphs.

As we mentioned above the STAGGER problem was used by a number of other systems that deal with concept drift. To compare the present approach with the previous approaches, we summarised all available results from experiments using this dataset in Table 2. The first ten rows of this table present the results from experiments with other similar approaches. The last three rows present the results from the experiments with the algorithms using the Time Window Optimisation (TWO) mechanism. The results are shown in more detail to facilitate a better comparison of the systems' performance on different concepts: (1), (2) and (3), shown in separate columns. The last column shows the average performance on the whole dataset. We will look more closely at the results from the second and third concepts, where actually the systems adapt to changes in the underlying concept. The performance of the systems on this dataset depends on the basic learning algorithms, which seem to perform differently on this dataset. Therefore we will mainly be interested in comparing the systems that use the same basic learning algorithm.

The algorithms that use the present TWO mechanism (rows 11-13) significantly outperform the first three systems (rows 1 to 3). The results presented in row 4, are

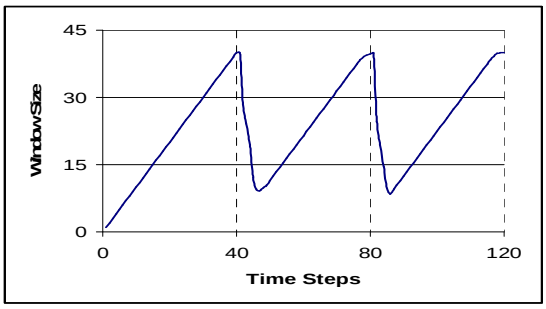

Fig. 3. The average window size for each step in the experiment with NBC-TWO

from the experiments with the COPL mechanism, using NBC as the basic learning algorithm. Therefore, we compared it with NBC-TWO, which achieves a significantly better performance for this dataset.

The FLORA systems reach significantly higher accuracy, than all other approaches on the first concept, which is actually a stable one. However, NBC-TWO and ID3-TWO significantly outperform all FLORA systems on the concepts (2)-(3), i.e. after concept drift has occurred. In general, kNN does not perform very well on this dataset, despite this kNN-TWO significantly outperforms the FLORA2 and FLORA3 algorithms on the concepts (2)-(3).

Concept: / Algorithm:	(1)	(2)	(3)	(1)-(2)-(3)
1. IB2 [1]	80.4	51.9	55.6	63.9
2. AQBL [12]	89.6	57.2	55.7	67.0
3. AQPM [12]	89.7	70.5	75.1	79.9
4. COPL (NBC) [7]	91.2	78.9	85.9	85.3
5. FLORA 2 [18]	98.8	80.4	80.3	86.5
6. FLORA 3 [18]	98.7	80.7	79.5	86.0
7. FLORA 4 [18]	98.0	82.5	82.7	87.7
8. kNN – GF [6]	91.8	79.5	83.2	84.8
9. ID3 – GF [6]	93.9	78.3	89.0	87.07
10. NBC – GF [6]	92.4	83.9	88.9	88.4
11. kNN – TWO	*91.9*	*81.4*	*86.3*	*86.5*
12. ID3 – TWO	*93.1*	*84.2*	*89.8*	*89.0*
13. NBC – TWO	*93.9*	*85.4*	*91.5*	*90.3*

Table 2. Average classification accuracy of systems with embedded concept drift tracking mechanisms on the STAGGER dataset

The results reported in rows 8 to 11 are from algorithms that use the Gradual Forgetting (GF) mechanism [6]. For this dataset the GF mechanism uses a fixed-size (30) time window in which the examples are assigned gradually decreasing weights using a linear forgetting function. We are comparing TWO and GF mechanisms on pairs that use the same learning algorithm (e.g. rows 9 and 12). We can see that there is no difference in average accuracy on the first concept. There is a small, but significant improvement of the accuracy obtained by TWO for the changed concepts (2)-(3).

4.2 German Credit Dataset

This subsection presents the results from the experiments conducted with the German credit dataset, from the UCI machine learning Repository. 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. The dataset was 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.

Memory: / Algorithm:	FM	FTW	TWO
kNN	68.25	72.37	77.75*^
ID3	77.00	75.50	79.00*^
NBC	78.37	75.87	78.63 ^

Table 3. The improvement of the classification accuracy when the TWO mechanism is applied to the Credit dataset (sorted by *"age"* attribute).

Table 3 shows the results from the experiments with the Credit dataset (sorted by *"age"* attribute). With all algorithms an improvement in classification accuracy was achieved when the algorithms using TWO mechanism were applied. All these improvements are significant except the comparison with NBC-FM.

Table 4 shows the results from the experiments with the Credit dataset (sorted by "*checking_status"* attribute). The results show that the TWO mechanism improves the classification accuracy of the algorithms and these improvements are significant for all algorithms.

Memory: / Algorithm:	FM	FTW	TWO
kNN	64.00	71.75	77.13*^
ID3	64.87	71.75	75.25*^
NBC	74.37	74.14	77.76*^

Table 4. The improvement of the classification accuracy when the TWO mechanism is applied to the Credit dataset (sorted by "*checking_status"* attribute).

The results also show that a fixed time window does not always provide an improvement of the accuracy and can even be destructive compared to the full memory learning algorithm. The problem with it is that we do not know in advance how the concept will change and what will be the best size in the future. Even with some "cheating", by using a time window approximating the average optimal window size in the experiments with the TWO mechanism, an improvement is achieved in only half of the cases with this dataset.

4.3 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 the 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.

Memory: / Algorithm:	FM	FTW	TWO
kNN	90.12	90.10	92.48*^
ID3	87.08	86.56	89.51*^
NBC	90.61	90.78	91.56*^

Table 5. The improvement of the classification accuracy when the TWO mechanism is applied to the Spam dataset, sorted by *"capital_run_length_total"* attribute (ster 10).

Table 5 presents the results from the experiments with the Spam dataset comparing full memory learning, time window with fixed size and the time window with optimized size. 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. For this dataset for two of the algorithms (kNN and NBC) the fixed time window improves the classification accuracy, but not significantly in either case. For ID3 we can even see a slight decrease in the accuracy. An improvement of the classification accuracy for all algorithms was achieved when the TWO mechanisms were applied and all those improvements are significant compared to FM and FTW as well.

Figure 4 shows on the same diagram: the classification accuracy on the test step (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. The presented algorithm is very robust to such 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 flags both real and false concept drifts, but the window size optimizer responds very differently to the two possibilities.

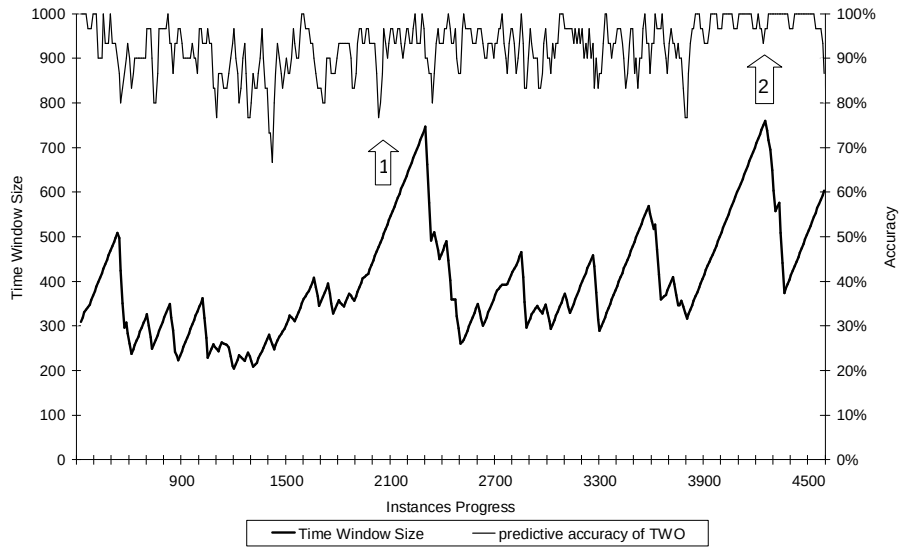

Fig. 4. The relationship between the accuracy of prediction measured on the test set and the time window size.

5 Conclusion

The paper presents a mechanism for dealing with the concept drift problem, which uses a statistical test to detect whether the current concept is changing. If a concept drift is detected, then the mechanism optimizes the time window size to achieve maximum accuracy of prediction. The algorithm is self-adapting and it can be used in many datasets without any predefined domain-dependent heuristics or parameter. Moreover, the developed mechanism is not attached to a particular learning algorithm. It is general in nature and can be added to any relevant algorithm.

The results from experiments with three learning algorithms using three datasets provide strong evidence that the mechanism is able significantly to improve the classification accuracy on drifting concepts.

Acknowledgements

We would like to thank Gerhard Widmer, and Marcus Maloof for providing data from their experiments. Thanks also to David Harper, Dietrich Wettschereck and Nirmalie Wiratunga for their very helpful comments on an early draft of the paper. This research was partially supported by EU FP6 Marie Curie grant KT-DigiCult-Bg, MTKD-CT-2004-509754.

References

1. Aha, D., Kibler, D. and Albert, M.: Instance-Based Learning Algorithms. Machine Learning 6, (1991) 37-66
2. Burghes, D. and Graham, A.: Introduction to Control Theory including Optimal Control: Ellis Horwood Series Mathematics and its Applications. John Wiley & Sons (1980)
3. Delany, SJ., Cunningham. P., Tsymbal, A. and Coyle, L.: 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
4. Gama, J., Medas, P., Castillo, G. and Rodrigues, P.: 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
5. Harries, M. and Sammut, C.: Extracting Hidden Context. Machine Learning 32 (1998) 101-126
6. 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
7. 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
8. 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
9. 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
10. 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
11. Lazarescu, M., Venkatesh, S. and Bui H. H.: Using Multiple Windows to Track Concept Drift. In the Intelligent Data Analysis Journal, Vol 8 (1), (2004) 29-59
12. Maloof, M. and Michalski, R.: Selecting examples for partial memory learning. Machine Learning 41 (2000) 27-52
13. Mitchell, T., Caruana, R., Freitag, D., McDermott, J. and Zabowski, D.: Experience with a Learning Personal Assistant. Communications of the ACM 37(7) (1994) 81-91
14. Mitchell T. Machine Learning. McGraw-Hill (1997)
15. Quinlan, R.: Induction of Decision Trees. Machine Learning 1 (1986) 81-106
16. Schlimmer, J. and Granger, R.: Incremental Learning from Noisy Data. Machine Learning 3, (1986), 317-357
17. Widmer, G.: Tracking Changes through Meta-Learning. Machine Learning 27 (1997) 256-286
18. Widmer, G. and Kubat, M.: Learning in the presence of concept drift and hidden contexts: Machine Learning 23 (1996) 69-101