# Dedicated Memory Models for Continual Learning in the Presence of Concept Drift

**Viktor Losing, Barbara Hammer**
Bielefeld University
Universitaetsstr. 25
33615 Bielefeld, Germany
vlosing@techfak.uni-bielefeld.de,
bhammer@techfak.uni-bielefeld.de

**Heiko Wersing**
Honda Research Institute Europe
Carl-Legien-Str. 30
63073 Offenbach, Germany
heiko.wersing@honda-ri.de

## Abstract

Data Mining in non-stationary data streams is gaining more attention recently, especially in the context of Internet of Things and Big Data. It is a highly challenging task since the different types of possibly occurring concept drift undermine classical assumptions such as data independence or stationary distributions.
We propose the Self Adjusting Memory (SAM) model, which can deal with heterogeneous concept drift, i.e different types and rates, using biologically inspired memory models and their coordination. The idea is to construct dedicated models for the current and former concepts and apply them according to the given situation. This general approach can be combined with various classifiers meeting certain conditions, which we specify in this contribution. SAM is easy to use in practice since a task specific optimization of the meta parameters is not necessary.
We recap the merits of our architecture with the k Nearest Neighbor classifier and evaluate it on artificial as well as real world benchmarks. SAM's highly competitive results throughout all experiments underline its robustness as well as its capability to handle heterogeneous concept drift.

## 1 Introduction

An ever growing field of real world applications generates data in streaming fashion at increasing rate, requiring large-scale and real-time processing. Streaming data is prevalent in domains such as health monitoring, traffic management, financial transactions, social networks [1] and is the foundation of the Internet of Things [2] technology. Supplementing streaming data with non-stationary environments leads to one of the recent key areas in data mining research: Learning in streams under concept drift. Here, algorithms are challenged by a variety of possible forms of concept drift under strict limitations in terms of memory consumption and processing time.
In recent years, a few algorithms have been published able to handle specific types of concept drift. Adaptive Sliding Windowing (ADWIN) [3] by Bifet et al. monitors the error history and detects significant changes, which is particularly effective for abrupt drift. In [4] Bifet et al. combine ADWIN with a dynamic sliding window technique, termed Probabilistic Adaptive Windowing (PAW), and the k Nearest Neighbor (kNN) classifier [5]. Whenever a change is detected the window is accordingly shrinked. ADWIN was also coupled with a modified Online Bagging [6] approach named Leveraging Bagging (LVGB) in [7]. Jaber et al. presented with Dynamic Adaption to Concept Changes (DACC) [8] also an ensemble algorithm, which performed well for incremental concept drift. A classifier of the worst half of the pool is randomly removed after a predefined number of examples and replaced by a new one. Predictions are solely done by the best classifier within the pool. Learn++.NSE [9] by Elwell et al. processes incoming examples in chunks with a predefined size. A base classifier is trained for each chunk and added to the ensemble. In contrast to other methods, members are not

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

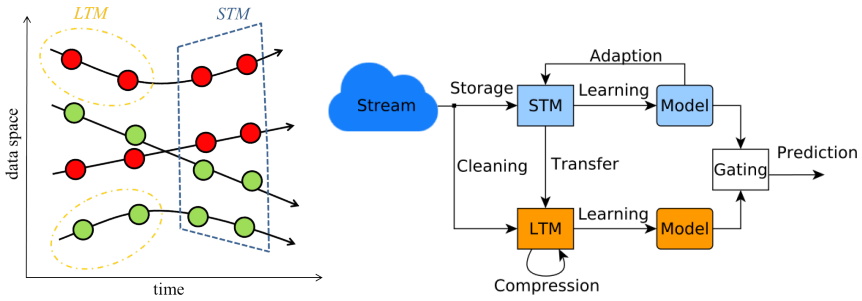


Figure 1: Illustration of the general approach (on the left). The STM contains only the current concept, while the LTM preserves only knowledge which is consistent in regard to the STM. The SAM architecture is depicted on the right. Incoming examples are stored within the STM. The cleaning process keeps the LTM all-time consistent with the STM. Whenever, the STM is reduced in size, its discarded knowledge is transfered into the LTM. Accumulated knowledge within the LTM is compressed each time the available space is exhausted. Both models are considered during prediction.

continuously learning but preserve their initial state. This fact is used to revive former members in the presence of reoccurring concept drift.

Even though some methods can be used for several types of drift by an according setting of their meta parameters, this requires explicit prior knowledge about the task at hand. Furthermore, in real world applications, data usually do not change only in one specific form, but instead multiple, sometimes even concurrent, types of concept drift are taking place at various rates. One example is the field of personalized assistance, in which individual user behavior is taken into account to provide appropriate assistance in various situations [10]. But, individual behavior in particular can change in arbitrary ways. Systems anticipating only certain forms of concept drift, will perform sub-optimal at best, or fail completely at worst, when unexpected forms of change occur.

Our Self Adjusting Memory (SAM) is able to cope with heterogeneous concept drift and can be easily applied in practice without any parametrization. It exhibits several analogies to the structure of the human memory as knowledge is explicitly partitioned among a short- and long-term memory. Thereby, it omits a common weakness of state of the art approaches which simply discard former knowledge and, therefore, have to relearn old concepts in case of reoccurence.

We recap the benefits of SAM by applying it in combination with the kNN classifier. An extensive evaluation on common benchmarks demonstrates the gain of our approach in comparison to current state of the art methods. It exclusively achieves highly competitive results throughout all experiments, demonstrating its robustness and the capability to handle different types of concept drift.

## 2 Streaming setting

Our focus is data stream classification under supervised learning for incremental / on-line algorithms. A potentially infinite sequence $S = (s_1, s_2, \ldots, s_t, \ldots)$ of tuples $s_i = (\mathbf{x}_i, y_i)$ arrives one after another. As $t$ represents the current time stamp, the learning objective is to predict the target variable $y_t \in \{1, \ldots, c\}$ for a given set of features $\mathbf{x}_t \in \mathbb{R}^n$. The prediction $\hat{y}_t = h_{t-1}(\mathbf{x}_t)$ is done according to the previously learned model $h_{t-1}$. Before proceeding with the next example, the applied learning algorithm generates a new model $h_t = \text{train}(h_{t-1}, s_t)$ based on the current tuple $s_t$ and the previous model $h_{t-1}$. Algorithmic performance is measured via the Interleaved Test-Train error $E(S) = \frac{1}{t}\sum_{i=1}^{t} \mathbb{1}(h_{i-1}(x_i) \neq y_i)$.

## 3 Architecture of the Self Adjusting Memory (SAM)

The basic idea of the SAM architecture is to combine dedicated models for the current concept $P_t(\mathbf{x}, y)$ and all former ones $P_{t-1}(\mathbf{x}, y), \ldots, P_1(\mathbf{x}, y)$ in such a way that the prediction error is minimized. We construct two different memories: The Short-Term Memory (STM), containing data of the current concept and the Long-Term Memory (LTM), maintaining knowledge of past concepts. This approach is illustrated by Figure 1 on the left. We share the general assumption of new data being

more relevant for current predictions. Hence, we remove those information from former concepts which is in conflict with the current one, but we explicitly preserve the rest in compressed fashion. We avoid any parametrization, by exploiting the minimization of the error on recent data at various steps. Our architecture is depicted on the right of Figure 1 and described in the following.

### 3.1 Model definition

Memories are represented by sets $M_{\text{ST}}$, $M_{\text{LT}}$, $M_{\text{C}}$. The STM represents the current concept and is a dynamic sliding window containing the most recent $m$ examples of the data stream

$$M_{ST} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{1, \ldots, c\} \,|\, i = t - m + 1, \ldots, t\}.$$

The LTM preserves all former information which is not contradicting those of the STM in a compressed way. In contrast to the STM the LTM is neither a continuous subpart of the data stream nor given by exemplars of it, but instead a set of $p$ points

$$M_{LT} = \{(\mathbf{x}_i, y_i) \in \mathbb{R}^n \times \{1, \ldots, c\} \,|\, i = 1, \ldots, p\}.$$

The combined memory (CM) is the union of both memories

$$M_C = M_{ST} \cup M_{LT}.$$

Every set induces a classifier $\text{C} : \mathbb{R}^n \mapsto \{1, \ldots, c\}$, $\text{C}_{M_{\text{ST}}}$, $\text{C}_{M_{\text{LT}}}$, $\text{C}_{M_{\text{C}}}$. Weights $w_{\text{ST}}$, $w_{\text{LT}}$, $w_C$ are representing the accuracy of the corresponding model on the current concept. The prediction of our complete model relies on the sub-model with the highest weight.
The hyperparameters of the model consist of the minimal length of the STM $L_{\text{min}}$ as well as the maximum number of stored examples $L_{\text{max}}$ (STM & LTM combined). Furthermore, the meta parameters of the chosen classifier model have to be set, in our case the $k$ for kNN. These can be robustly chosen and do not require a task specific setting[1].

### 3.2 Model adaption

The adaption comprises the size $m$ of the STM, the data points in the LTM and the weights of the sub-models. We denote a data point at time $t$ as $(\mathbf{x}_t, y_t)$, the corresponding memories as $M_{\text{ST}_t}$, $M_{\text{LT}_t}$, $M_{\text{C}_t}$ and the sub-model weights as $w_{\text{ST}_t}$, $w_{\text{LT}_t}$, $w_{\text{C}_t}$.

**Adaption of the Short Term Memory** The STM is a dynamic sliding window, whose role is to exclusively contain data of the current concept. Therefore, its size has to be reduced, whenever the concept changes such that examples of the former concept are dropped. This is done by adjusting the size such that the Interleaved Test-Train error of the remaining STM is minimized. This approach relies on the fact that a model trained on internally consistent data yields less errors and we assume the remaining instances to represent the current concept or being sufficiently "close" to it.
To compare only a logarithmic number, we use bisection for the selection of the window size. Tested windows are $W_l = \{(\mathbf{x}_{t-l+1}, y_{t-l+1}), \ldots, (\mathbf{x}_t, y_t)\}$ where $l \in \{m, m/2, m/4, \ldots\}$ and $l \geq L_{\text{min}}$. We adopt the window with the minimum error

$$M_{\text{ST}_{t+1}} = \underset{S \in \{W_m, W_{m/2}, \ldots\}}{\arg\min} E(S).$$

**Cleaning and Transfer** The dropped data, resulting from the shrinking of the STM, is transferred into the LTM. However, to keep the LTM consistent with the STM we "clean" the dropped data beforehand. This process filters instances which are contradicting those in the STM. Instances contradict each other when they have different labels but their Euclidean distance is smaller than an adaptive distance threshold. To provide all-time consistency we further "clean" the LTM according to every incoming example. A formal definition is given in [11].

**Compression of the LTM** In contrast to the FIFO principle of the STM, instances are not fading out as soon as the size limit of the LTM is reached. Instead, we condense the available information to a sparse knowledge representation via clustering to preserve information as long as possible. For every class label $\hat{c}$ we group the corresponding data points in the LTM to $|M_{LT_{\hat{c}}}|/2$ clusters[2]. This process is repeated each time the size limit is reached leading to a self-adapting level of compression.

[1] We used for all experiments $k = 5$, $L_{\text{min}} = 50$, $L_{\text{max}} = 5000$.

[2] We used kMeans++ [12] because of its efficiency and scalability to larger datasets.

**Model weight adaption** The weight of a memory is its accuracy averaged over the last $m_t$ samples, where $m_t = |M_{\mathrm{ST}_t}|$ is the size of the current STM. The weight of the LTM at time stamp $t$ equals

$$w_{\mathrm{LT}_t} = \frac{|\{i \in \{t - m_t + 1, \ldots, t\} \,|\, \mathrm{C}_{M_{\mathrm{LT}_i}}(\mathbf{x}_i) = y_i\}|}{m_t}$$

and analogously for STM and CM.

**Classifier choice** Our approach repeatedly modifies the memories by adding new instances or selectively removing contradicting ones. Hence, a model is required which allows an efficient incremental and decremental adaptation. The kNN algorithm is a natural fit to SAM but also other methods such as the Naive Bayes [13] or the incremental and decremental SVM [14] are valid choices. We already conducted some experiments with the Naive Bayes algorithm using non-stationary data in the domain of text classification and the first results are very promising.

## 4 Experiments

We couple the SAM architecture with the kNN classifier and evaluate it in comparison to well-known state of the art algorithms for handling concept drift in streaming data. Next to the methods already discussed in section 1, we compare against a distance weighted kNN classifier with a sliding window of fixed size ($\mathrm{kNN_S}$). The evaluation was done with common artificial and real world benchmarks, which are extensively described in [11]. Window based approaches were allowed to store up to $5000$ samples but never more than $10\%$ of the whole dataset. No dataset specific hyperparameter tuning was done.

The error rates of all experiments are shown in Table 1. SAM significantly outperforms the others by having nearly half the error rate in average compared to the second best method LVGB. Even more important is the fact that whereas other methods struggle at some datasets our approach delivers robust results without any hiccup. All concept drift types are handled better or at least competitive.

Our results confirm the fact that kNN is in general a very competitive algorithm in the streaming setting. It is quite surprising that the simple sliding window approach $\mathrm{kNN_S}$ performs comparably well or even better than more sophisticated methods such as DACC or L++.NSE.

Table 1: Error rates of all experiments. The lowest rate for each dataset is marked bold.

| Dataset | L++.NSE | DACC | LVGB | $\mathrm{kNN_S}$ | PAW | SAM |
|---|---|---|---|---|---|---|
| SEA Concepts | 14.48 | 15.68 | **11.69** | 13.83 | 13.39 | 12.50 |
| Rotating Hyperplane | 15.58 | 18.20 | **12.53** | 16.00 | 16.16 | 13.31 |
| Moving RBF | 44.50 | 54.34 | 44.84 | 20.36 | 24.04 | **15.30** |
| Interchanging RBF | 27.52 | **1.40** | 6.11 | 45.92 | 8.56 | 5.70 |
| Moving Squares | 65.90 | **1.17** | 12.17 | 68.87 | 61.01 | 2.30 |
| Transient Chessb. | **1.98** | 43.21 | 17.95 | 7.36 | 14.44 | 6.25 |
| Mixed Drift | 40.37 | 61.06 | 26.29 | 31.00 | 26.75 | **13.33** |
| Artificial ∅ | 30.05 | 27.87 | 18.80 | 29.05 | 23.48 | **9.81** |
| Artificial ∅ Rank | 4.00 | 4.57 | 2.86 | 4.29 | 3.57 | **1.71** |
| Weather | 22.88 | 26.78 | 21.89 | **21.53** | 23.11 | 21.74 |
| Electricity | 27.24 | 16.87 | **16.78** | 28.61 | 26.13 | 17.52 |
| Cover Type | 15.00 | 10.05 | 9.07 | **4.21** | 6.76 | 4.8 |
| Poker Hand | 22.14 | 20.97 | **13.65** | 17.08 | 27.94 | 18.45 |
| Outdoor | 57.80 | 35.65 | 39.97 | 13.98 | 16.30 | **11.25** |
| Rialto | 40.36 | 28.93 | 39.64 | 22.74 | 24.96 | **18.58** |
| Real world ∅ | 30.90 | 23.21 | 23.50 | 18.03 | 20.87 | **15.40** |
| Real word ∅ Rank | 5.33 | 4.17 | 3.17 | 2.33 | 4.00 | **2.00** |
| Overall ∅ | 30.44 | 25.72 | 20.97 | 23.96 | 22.27 | **12.39** |
| Overall ∅ Rank | 4.62 | 4.38 | 3.00 | 3.38 | 3.77 | **1.85** |

## 5 Conclusion

In this paper we presented the Self Adjusting Memory (SAM) architecture, especially designed to handle heterogeneous concept drift within streaming data. It explicitly separates the current concept from former ones and preserves both in dedicated memories. Thereby, it omits a common weakness of available methods which simply discard former knowledge and, therefore, have to relearn concepts in case of reoccurring concept drift. Our method is easy to use in practice since it requires no task-specific meta-parameterization.

We compared SAM with current state of the art methods on various artificial and real world benchmarks. As the only algorithm in the field, it demonstrated consistently accurate results for heterogeneous concept drift.

## References


[1] Min Chen, Shiwen Mao, and Yunhao Liu. Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209, 2014.

[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A Survey. *Computer networks*, 54(15):2787–2805, 2010.

[3] Albert Bifet and Ricard Gavalda. Learning from Time-Changing Data with Adaptive Windowing. In *SIAM International Conference on Data Mining (SDM)*, volume 7, page 2007. SIAM, 2007.

[4] Albert Bifet, Bernhard Pfahringer, Jesse Read, and Geoff Holmes. Efficient Data Stream Classification via Probabilistic Adaptive Windows. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, SAC '13, pages 801–806, New York, NY, USA, 2013. ACM.

[5] Sahibsingh A Dudani. The Distance-Weighted k-Nearest-Neighbor Rule. *Systems, Man and Cybernetics, IEEE Transactions on*, (4):325–327, 1976.

[6] Nikunj C. Oza. Online Bagging and Boosting. In *IEEE International Conference on Systems, Man and Cybernetics 2005*, volume 3, pages 2340–2345. IEEE, 2005.

[7] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging Bagging for Evolving Data Streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 135–150. Springer, 2010.

[8] Ghazal Jaber, Antoine Cornuéjols, and Philippe Tarroux. *Online Learning: Searching for the Best Forgetting Strategy under Concept Drift*, pages 400–408. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[9] Ryan Elwell and Robi Polikar. Incremental learning of concept drift in nonstationary environments. *IEEE Transactions on Neural Networks*, 22(10):1517–1531, 2011.

[10] Silvia Schiaffino, Patricio Garcia, and Analia Amandi. eTeacher: Providing personalized assistance to e-learning students. *Computers & Education*, 51(4):1744–1754, 2008.

[11] Viktor Losing, Barbara Hammer, and Heiko Wersing. KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift. In *16th International Conference on Data Mining (ICDM)*. IEEE, 2016.

[12] David Arthur and Sergei Vassilvitskii. k-means++: The Advantages of Careful Seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 1027–1035. Society for Industrial and Applied Mathematics, 2007.

[13] Harry Zhang. The Optimality of Naive Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, Miami Beach, Florida, USA*, pages 562–567, 2004.

[14] Tomaso Poggio and Gert Cauwenberghs. Incremental and Decremental Support Vector Machine Learning. *Advances in Neural Information Processing Systems*, 13:409, 2001.