KNN classifier with self adjusting memory for heterogenous concept drift

Viktor Losing*†, Barbara Hammer* and Heiko Wersing†
*Bielefeld University, Universitätsstr. 25, 33615 Bielefeld
†HONDA Research Institute Europe, Carl-Legien-Str. 30, 63073 Offenbach am Main

***Abstract*—Learning from 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 fundamentally different types of possibly occurring drift undermine classical assumptions such as i.i.d data. Incremental drift characterizes a continuous change in the distribution such as the signals of a slowly degrading sensor. A suddenly malfunctioning sensor on the other hand causes a severe shift and is defined as abrupt drift. Available algorithms are able to handle different types of drift, however they target either abrupt or incremental drift and often incorporate hyperparameter requiring a priori knowledge about the task at hand.**

We propose a biological inspired, architecture which partitions the data into a short-term and long-term memory. The former is a window of recently seen data-points whose size is adjusted such that the estimated generalization error is minimized. The latter preserves only those information from previous concepts which are non-conflicting to the current one. These memories are combined according to the demands of the present concept to classify unseen data points. We couple our parameter free approach with the K-Nearest Neighbor classifier, however, any other incremental learning algorithm could be used as well.

New artificial and real datasets are proposed to evaluate performance on specific types of drift. Experiments on these as well as on generally known benchmark datasets compare our approach with state of the art methods. The highly competitive results throughout all experiments underline the robustness of our approach.

I. Introduction

II. Concept Drift

Definition and examples

A. Types of Drift

-abrupt, incremental, gradual, reoccuring, virtual

III. Related Work

-Survey by gamma eventuell Ditzler -Active drift detection work Gamma, Bifet ADWIN for abrupt to minimize delay -Passive drift handling with ensembles DACC, LPPNSE -Windowing but usually throws away information which still can be valid, no LTM -SVM Leave one out -KNNPAW -KNN for streaming like the ICDM paper

IV. Architecture

Our architecture is depicted in figure **??**. It consists of 4 different steps. The goal is to keep as much as possible of the current concept in the Short Term Memory(STM) and at the same time preserve former knowledge in the Long Term Memory (LTM). We assume that the recent information is correct and reflects the current concept. Therefore, it is crucial to make sure that the memory in the LTM is not conflicting.

A. Short Term Memory

-STM, partitioning, O(log n) tests, maximizing generalization accuracy, write down in fancy formal way -LTM, keep not compromising -Transfer -choosing of proper prediction model(STM, or LTM, or Both) -Clustering -Size management

B. Efficiency

-distances have to be calculated anyways once, can be stored and afterwards need only to be sorted -fits with other algorithms as long they can be trained incremental and decremental e.g. NB, incremental SVM, but how to solve validation procedure -approximative KNN -adapt stm not every example but every n examples

V. Algorithms

We compare our method with well-known state of the art algorithms for handling drift in streaming data. Implementations of the original authors or those available in MOA [1] have been used to conduct the experiments. The following algorithms are considered:

Learn++.NSE
Proposed in [2], this algorithm processes incoming samples in chunks with a predefined size. A base classifier is trained for each chunk and added to the ensemble. The loss on recent chunks is averaged with a sigmoidal function to compute the final weight of each base classifier. Similar to AdaBoost [**?**], instances are weighted such that misclassified inputs have a higher impact on the calculated loss. Chunk-wise trained models have by design an adaption delay depending on the chunk size(XXVL chunksize very crucial). The base classifier are in our case Classification and Regression Trees [3].

Leveraging Bagging
Bifet et al. propose in [4] to increase the randomization of Online Bagging [5] and thereby the diversity of the ensemble. This is done by a higher λ value for the poisson distribution and the usage of output detection codes. Additionaly, they use ADWIN as change detector for every classifier within the ensemble such that whenever a change is detected the worst classifier is replaced by a new one. Hoeffding Trees[6] with Gaussian Naive Bayes within the leaves are used as base classifier. XXVL maybe define Hoeffding trees.

Dynamic Adaption of Concept Changes
Within this ensemble algorithm [7] a classifier of the worst

TABLE I. THE COMPARED ALGORITHMS.

Abbr.	Classifier	Parameter
HT_A	Hoeffding Tree with ADWIN	
LPPNSE	Learn++.NSE with CART	chunk-Size = various
DACC	Dynamic with HT	n=10
LVGB	Leveraging Bagging with HT	n=10
KNN_S	KNN with sliding window	w=5000, k=5
KNN_{W_A}	NN with PAW+ADWIN	w=5000, k=5
KNN_M	KNN with STM+LTM memory	w=5000, k=5

TABLE II. EVALUATED ARTIFICIAL DATASETS.

Dataset	#Samples	#Feat.	#Class	Drift type
SEA Concepts	50K	3	2	abrupt
Rotating Hyperplane	200K	10	2	incremental
Moving RBF	200K	10	5	incremental
Interchanging RBF	200K	2	10	abrupt
Moving Squares	200K	2	4	incremental
Transient Chessboard	200K	2	8	virtual
Mixed Drift	600K	2	8	abr/incr/virt

half of the pool is removed randomly after a predefined number of examples and replaced by a new one. Newly generated classifier are excluded for a predefined time from the elimination process. Predictions for incoming examples are solely done by the classifier within the pool which achieved the highest accuracy in the past. As in Leavarging Bagging, Hoeffding Trees are chosen as base classifier.

kNN sliding window

A distance weighted kNN classifier is combined with a fixed window size containing the most recent samples. This so called sliding window approach is a common way for drift handling, since the window contains usually the most relevant examples for the future predictions. However, the choice of the static size is crucial and usually has to be chosen according to the given problem. A too small window hinders the classifier to gain a low error rate, whereas a too large window may be very slow in adapting to new concepts.

kNN with Probabilistic Adaptive Window and ADWIN

In contrast to the approach FIFO principle of the sliding window approach, examples here are removed randomly leading to a mix of recent and older samples in the window. The window size is not strictly bounded and varies around a target size (see [8]). This approach also uses ADWIN as change detector and clears the window (XXVL gegebenenfalls) accordingly.

Table I gives an overview of the algorithms as well as the chosen hyperparameter. A maximum of 5000 samples was allowed as size for the window based approaches. However, we limited it to at most 10% of the whole dataset for those with rather few examples. LPPNSE demands a chunk size which is critical for its performance. To avoid any disadvantage we evaluated several sizes and report the best result. No further dataset specific hyperparameter tuning was done, since we wanted to use(XXVL reinstecken)as little prior knowledge as possible.

VI. DATASETS

We used own and well known artificial as well as real world datasets for the experiments. Links to all datasets and algorithms are available at https://github.com/vlosing/Online-learning. In the following we describe the data more detailed.

Artificial datasets have the advantage that any desired drift behavior can be explicitly simulated. They are often 2-dimensional to enable a straightforward visualization. We evaluated ready to use published benchmark datasets as well as MOA generated ones using common parametrization in the literature. We also added four new datasets allowing the evaluation of particular algorithm properties which are in our opinion not yet considered enough in the community. Table II shows their main characteristics.

SEA Concepts

This dataset was proposed in [9] and consists of 50000 samples with three attributes of which only two are relevant. The two class decision boundary is given by $f_1 + f_2 = \theta$, where f_1 and f_2 are the two relevant features and θ a predefined threshold. Abrupt drift is simulated with four different concepts, by changing the value of θ every 12500 samples. This dataset includes also 10% of noise.

Rotating Hyperplane

A hyperplane in d-dimensional space is defined by the set of points x that satisfy $\sum_{i=1}^{d} w_i x_i = w_0$. The position and orientation of the hyperplane are changed incrementally by continuously adding a term δ to the weights $w_i = w_i + \delta$. We used the Random Hyperplane generator in MOA with the same parameters as in [8] (10 dimensions, 2 classes, δ=0.001).

Moving RBF

Gaussian distributions with random initial positions, weights and standard deviations are moved with constant speed v in d-dimensional speed. The weight controls the partitioning of all samples among the Gaussians. We used the Random RBF generator in MOA with the same parameters as in [8] (10 dimensions, 50 Gaussians, 5 classes, v=0.001).

Interchanging RBF

Ten Gaussians with random covariance matrix are exchanging positions every 2000 samples and generate thereby a total of 100 abrupt drifts.

Moving Squares

Four equidistantly seperated squared uniform distributions are moving in horizontal direction with constant speed. The direction is inverted whenever the leading squares reaches a predefined boundary. Each square is representing a different class. The nice property of this dataset is that the upper bound of stored recent examples before old samples may start to overlap current ones can be easily calculated (120) and facilitates the analysation of algorithms, especially those using a sliding window approach, for incremental drift.

Transient Chessboard

Virtual drift is generated by revealing successively parts of a chessboard. This is done square by square randomly chosen from the whole chessboard such that each each square is representing an own concept. Every four revealed fields, samples covering the whole feature space are presented. This alternation penalizes algorithms simply forgetting former concepts. To reduce the impact of classification by chance we used eight classes per chessboard row.

Mixed Drift

The datasets interchanging RBF, moving squares and transient chessboard are simply positioned next to each other and samples of these are alternately introduced. Therefore, incremental, abrupt and virtual drift are occurring at the same time, requiring a local adaptation of an classifier to different drift types.

TABLE III. CONSIDERED REAL WORLD DATASETS. THE DRIFT TYPE WAS DETERMINED AS DESCRIBED IN VI-A.

Dataset	#Samples	#Feat.	#Class	Drift type
Weather	18159	8	2	virtual
Elec2	27549	6	2	real
Cover Type	581012	54	784	real
Outdoor	4000	21	40	real
Railto	40000	27	10	real

Unfortunately, only a few real world drift benchmarks are available, of which we used the largest ones. Additionally, we contribute two new challenging datasets obtained from visual data. The characteristics of all considered real world datasets are given in Table III.

Weather

Elwell et al. introduced this dataset in [2]. Using eight different features such as temperature, pressure wind speed etc. the target is to predict whether it is going to rain on a certain day or not at the Offutt Air Force Base in Bellevue, Nebraska. A period of 50 years is covered (1949-1999) summing up to 18159 samples with an imbalance towards no rain (69%).

Electricity market dataset (Elec2)

This problem is often used as a benchmark for concept drift classification. Initially described in [10], it was used thereafter for several performance comparisons [11], [12], [8], [13]. A critical note to its suitability as a benchmark can be found in [14]. The dataset holds information of the Australian New South Wales Electricity Market, whose prices are affected by supply and demand. Each sample characterized by attributes such as day of week, timestamp, market demand etc. refers to a period of 30 minutes and the class label identifies the relative change (higher or lower) compared to the last 24 hours. The dataset is often termed Elec2 and contains 45312 samples. However, we removed those with missing values such that 27449 points remained.

Forest Cover Type

Assigns cartographic variables such as elevation, slope, soil type asf of 30×30 meter cells to different forest cover types. Only forests with minimal human-caused disturbances were used, so that resulting forest cover types are more a result of ecological processes. It is often used as a benchmark for drift algorithms [8], [15], [16].

Outdoor Objects

This visual dataset was obtained from images recorded by a mobile robot approaching 40 different objects in a garden environment [17]. The lighting conditions between the approaches are varying significantly caused by different weather conditions and/or cast shadows as can be seen in Figure **??**. Each approach consists of 10 images and is represented in temporal order within the dataset. The objects are encoded in a normalized 21-dimensional rg-chromaticity histogram.

Railto Bridge Timelapse

Ten of the colorful buildings next to the famous rialto bridge in Venice are encoded in a normalized 27-dimensional rgb histogram. The images were obtained from timelapse videos of 10 consecutive days recorded in may 2016 by a webcam of XXVL. Continuously changing weather and lighting conditions affect the representation XXVL see figures. We excluded overnight recordings since they were too dark for being useful. Figure **??** shows some examples recorded at different times of day.

A. Assessing the drift type in real world data

While drift is explicitly generated in artificial datasets, it is rather difficult to identify the drift type or whether drift is present at all in real world data. Therefore, it is usually assumed in the comunity that there is some drift contained in the commonly evaluated datasets. We propose in this section a method which is able to determine the present drift type in a given dataset. To the best of our knowledge this has not been done so far in literature. Our two staged approach can distinguish between real, virtual or no drift at all. In the first step we conclude whether real drift is present. If that is not the case, we continue with the second step which detects virtual drift. No drift is present at all whenever both tests are negative. We use sliding windows of various sizes for this analysis. Even though the type of classifier is interchangeable, we utilize once again KNN.

1) Test for real drift: The idea for the first step is inspired by the observation that whenever real drift is present, sliding windows of smaller sizes tend to deliver higher accuracies than larger ones. This contradicts the classical assumption for stationary datasets as stated in the PAC model (XXVL reference!), that the error rate of a learning algorithm decreases with increasing number of examples. However, streaming data containing real drift is not stationary and more mistakes are done whenever outdated data is in conflict with samples of the current concept. A window which contains less of the outdated samples, delivers therefore a better result. We test whether the accuracies achieved with sliding windows of different sizes are significantly higher than the one obtained without any size restriction. To obtain multiple accuracies for a given window size bootstrapped samples of the dataset maintaining the initial inherent order are generated. Afterwards, we perform a one sided hypothesis test with the a p-value of XXVL%. The method is described exemplary for one window size s1 in the following.

XXVL make more general and analytic! Given a dataset X with n examples, we generate b bootstrap samples $\Psi = \hat{X}_1, \hat{X}_2 ... X_b$of the dataset and sort them such that the initial order in the dataset is preserved. We train a classifier with window size s_1 and another one with s_n using Ψ and asses their accuracies $a_1, a_2, ...a_b$ and $\tilde{a}_1, \tilde{a}_2, ...\tilde{a}_b$ respectively. The pth percentile of all accuracy differences $d_i = a_i - \tilde{a}_i$ is calculated. The null hypothesis, accuracy $a_i <= \tilde{a}_i$, is rejected if its value is smaller 0.

2) Test for virtual drift: In case of no real drift, we proceed with the second step and test for virtual drift. Here we permute the dataset each time before generating the bootstrapped samples and compare the resulting accuracies with those obtained in the first stage using the same window size. The permutation mixes data of all concepts. If P(X) is changed over time an algorithm should achieve higher accuracies, since it is easier to classify only one concept at a time instead of all of them combined.

We do know that the i.i.d. precondition, necessary for the validity of statistics obtained from bootstrapped samples, is violated in the setting of streaming data and that there are some rather theoretical cases in which our approach does not

work. Nonetheless, we see it as a strong indicator for the present drift type in a dataset.

3) Results of the applied tests: Next to the real world data we also analyze the drift type in some artificial datasets as a proof of concept. Included as well, is a dataset without any drift named Chessboard i.i.d.. We used 200 bootstrap samples per window size. The accuracy is obtained in the standard online learning setting in which each incoming example is first classified and afterwards used for training (often called "Interleaved test-train").
Table IV shows the mean accuracies of the bootstrapped samples with corresponding standard deviations and the inferred drift type after the respective tests. Our approach correctly classifies the drift type of all artificial datasets emphasizing its practical relevance. In the case of artificial datasets the increase / decrease of performance with shrinking window sizes is quite significant for datasets with / without real drift. This is generally less pronounced for real real world data. Nonetheless, the datasets Elec2, Outdoor and Rialto clearly benefit from smaller window sizes. Covtype contains little real drift, while the Weather task incorporates only virtual drift.

VII. Results

We evaluated the methods by measuring the Interleaved test-train accuracy. The error rates of all experiments are shown in Table V.
(XXVL name of algorithm) outperforms the other algorithms quite significantly by having nearly half the error rate in average compared to the second best method LVGB. Even more important is in our eyes the fact that while all other methods are struggling at some datasets our approach delivers very robust results without any hiccup. All drift types are handled better or at least competitive compared to the other algorithms. This is particularly clarified in the large accuracy gap achieved with the MIX dataset, which contains incremental, abrupt and virtual drift at the same time.
The methodically most similar method to ours is kNN_{W_A}, since it uses also kNN as classifier and actively manages its window. However, it performs in all experiments worse, even in those containing only abrupt drift.
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 of fixed window size kNN_S performs comparably well or even better than more sophisticated methods such as HTAdaptive or L++.NSE.

A. Memory behaviour

In this section we illustrate the adaptation of the STM and LTM as well as the different roles they take on depending on the demands of a given task. Figure 1 depicts on the left the size adjustment of the STM in case of abrupt drift during the Interchanging RBF experiment. The algorithm reliably reduces the window size each time after an abrupt drift occurred. However, we also observe a certain delay during which wrong predictions are likely to be done by the classifier. This delay is due to two reasons. Firstly, a certain amount of examples is required to construct a sufficiently well performing classifier for the new concept. Secondly, the more examples of the former concept are contained in the short term memory,

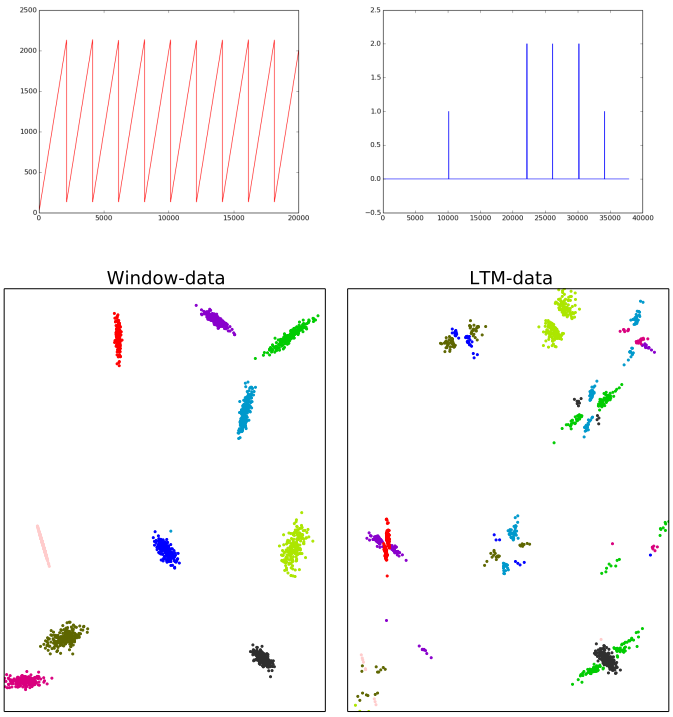

Fig. 1. Figures for Interchanging RBF

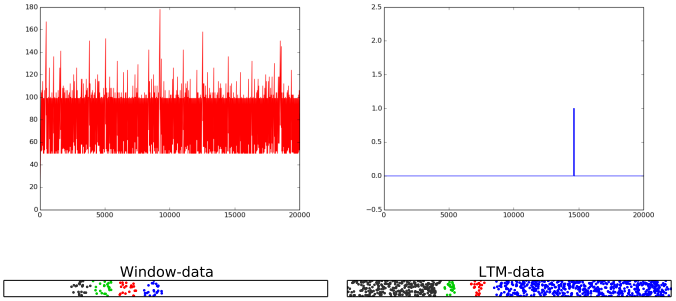

Fig. 2. Figures for Moving Squares

the more stable is the corresponding classifier and the more examples of the new concept are required to deteriorate its overall accuracy. Hence, the delay illustrates a self-adjusting natural trade-off between adaption speed to new concepts and the robustness against noise, governed by the stability of both concepts.
The consolidation of the LTM is also visible in Figure 1. All samples which were interfering with the STM were deleted, and therefore the feature space is empty there. Whereas the harmless samples around the empty spots are kept in the memory.
As already mentioned the Moving Squares dataset is constructed in such a way that the squares may overlap each other if more than 120 of the recent samples are kept in memory. Hence, the best strategy for this problem is to keep the memory as small as possible and to use only the most recent examples for prediction.

TABLE IV. MEAN ACCURACIES AND CORRESPONDING STANDARD DEVIATIONS OBTAINED FROM BOOTSTRAPPED SAMPLES OF THE DATASETS WITH VARYING SLIDING WINDOW SIZES.

Dataset	$KNN_{S\text{-}100}$	$KNN_{S\text{-}500}$	$KNN_{S\text{-}1000}$	$KNN_{S\text{-}5000}$	$KNN_{S\text{-}10000}$	KNN	Concluded drift type
Weather	82.03(0.12)	84.05(0.24)	84.36(0.25)	84.95(0.18)	84.92(0.20)	84.86(0.16)	not real
Elec2	**89.08(0.18)**	**87.46(0.17)**	**86.61(0.16)**	**85.07(0.11)**	**84.22(0.14)**	83.43(0.13)	real
Cover Type	94.65(0.01)	96.15(0.01)	**97.11(0.01)**	**96.95(0.01)**	**96.92(0.01)**	5	real
Outdoor	89.08(0.18)	87.46(0.17)	86.61(0.16)	85.07(0.11)	84.22(0.14)	83.43(0.13)	real
Railto	89.08(0.18)	87.46(0.17)	86.61(0.16)	85.07(0.11)	84.22(0.14)	83.43(0.13)	real
Chessboard i.i.d.	63.23(0.07)	83.63(0.07)	88.47(0.06)	94.67(0.03)	96.15(0.04)	98.37(0.02)	not real
Transient Chessboard	81.00(0.05)	88.56(0.04)	88.85(0.04)	94.81(0.02)	95.89(0.03)	98.36(0.03)	not real
Moving Squares	**97.73(0.02)**	**67.22(0.05)**	**63.30(0.05)**	56.87(0.11)	57.22(0.04)	57.12(0.10)	real
Interchanging RBF	97.73(0.02)	67.22(0.05)	63.30(0.05)	56.87(0.11)	57.22(0.04)	57.12(0.10)	real

(a) Stage 1: Mean accuracies resulted by evaluating bootstrapped samples with various different window sizes. Values achieved with restricted windows which are significantly higher compared to those without restriction are marked bold. A dataset contains real drift if this is the case for at least one size.

Dataset	$KNN_{S\text{-}100}$	$KNN_{S\text{-}500}$	$KNN_{S\text{-}1000}$	$KNN_{S\text{-}5000}$	$KNN_{S\text{-}10000}$	KNN	Concluded drift type
Weather	**81.14(0.30)**	**83.20(0.23)**	83.89(0.28)	84.42(0.17)	84.77(0.19)	84.63(0.18)	virtual
Chessboard i.i.d.	63.25(0.05)	83.63(0.06)	88.44(0.06)	94.71(0.04)	96.18(0.02)	98.36(0.01)	None
Transient Chessboard	**63.30(0.09)**	**83.77(0.06)**	**88.48(0.05)**	94.77(0.03)	96.20(0.05)	98.38(0.02)	virtual

(b) Stage 2: Mean accuracies resulted by evaluating permuted, bootstrapped samples with various different window sizes. Values which are significantly higher compared to those achieved in a) with the same window size are marked bold. Virtual drift is incorporated within a dataset when this is the case for at least one size.

TABLE V. ERROR RATES OF ALL EXPERIMENTS EVALUATED WITH THE INTERLEAVED TEST-THEN-TRAIN METHOD.

Dataset	HT_A	LPPNSE	DACC	LVGB	kNN_S	kNN_{W_A}	kNN_M
SEA Concepts	13.8	10.00	10.00	10.00	10.00	10.00	10.00
HYP	10.00	10.00	10.00	10.00	10.00	10.00	10.00
MRBF	10.00	10.00	10.00	10.00	10.00	10.00	10.00
IRBF	10.00	10.00	10.00	10.00	10.00	10.00	10.00
SQR	10.00	10.00	10.00	10.00	10.00	10.00	10.00
CHESS	10.00	10.00	10.00	10.00	10.00	10.00	10.00
MIX	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Art. avg	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Art. rank	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Weather	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Elec	10.00	10.00	10.00	10.00	10.00	10.00	10.00
CovType	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Outdoor	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Railto	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Real avg	10.00	10.00	10.00	10.00	10.00	10.00	10.00
Real rank	10.00	10.00	10.00	10.00	10.00	10.00	10.00
total avg	10.00	10.00	10.00	10.00	10.00	10.00	10.00
total rank	10.00	10.00	10.00	10.00	10.00	10.00	10.00

As can be seen in figure 2, the size of the STM is most of the time kept between 50 and 100 samples, allowing a nearly perfect prediction. As desired, the samples do not overlap each other within the STM as can be seen in its snapshot. The reason that the window is usually reduced precisely at 100 examples is that the default parameter for the minimal size of the STM is set to 50. Therefore, even better results are possible if the minimal size is further reduced.

In contrast to the previous two datasets which basically do not require the LTM, we see in figure 3 its importance during the prediction of the Transient Chessboard task. The STM is used alone whenever the data is presented square by square XXVL formulate reason. But in the phases introducing samples distributed over the whole board, the LTM is heavily used since it contains beneficial information from the past. The LTM snapshot reveals that the whole chessboard is preserved in a compressed fashion.

A further example for the task dependent relevance of the LTM and STM, this time considering real world data, is depicted in Fig.4. While the LTM is heavily used in the virtual drift dataset Weather, it is mainly neglected in the Rialto problem.

VIII. CONCLUSION

REFERENCES

[1] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "Moa: Massive online analysis," *The Journal of Machine Learning Research*, vol. 11, pp. 1601–1604, 2010.

[2] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, Oct 2011.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984.

[4] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Machine learning and knowledge discovery in databases*. Springer, 2010, pp. 135–150.

[5] N. C. Oza, "Online bagging and boosting," in *Systems, man and cybernetics, 2005 IEEE international conference on*, vol. 3. IEEE, 2005, pp. 2340–2345.

[6] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2000, pp. 71–80.

[7] G. Jaber, A. Cornuéjols, and P. Tarroux, "Online learning: Searching for the best forgetting strategy under concept drift," in *Neural Information Processing*. Springer, 2013, pp. 400–408.

[8] A. Bifet, B. Pfahringer, J. Read, and G. Holmes, "Efficient data stream classification via probabilistic adaptive windows," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ser. SAC

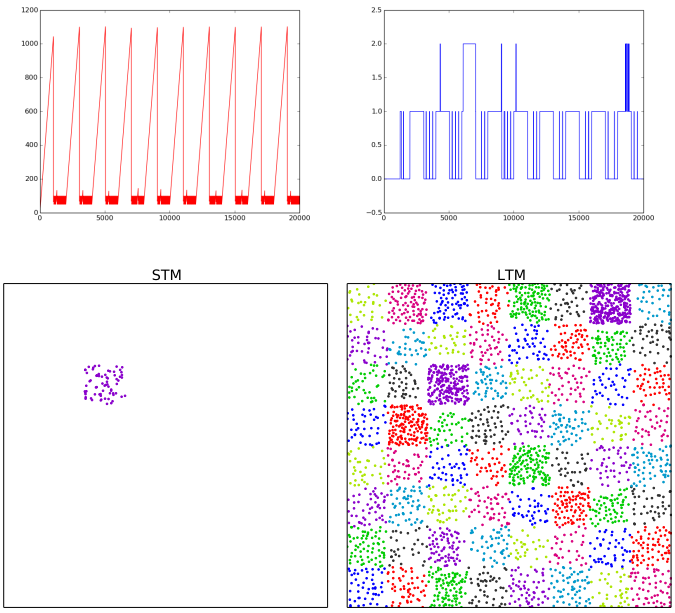

Fig. 3. Figures for T

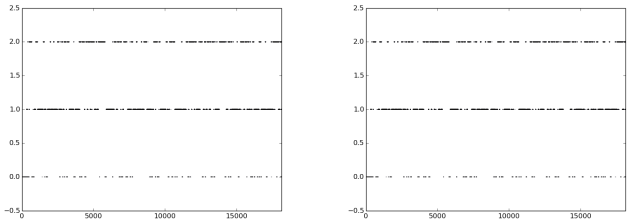

Fig. 4. Figures for T

'13. New York, NY, USA: ACM, 2013, pp. 801–806. [Online]. Available: http://doi.acm.org/10.1145/2480362.2480516

[9] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01. New York, NY, USA: ACM, 2001, pp. 377–382. [Online]. Available: http://doi.acm.org/10.1145/502512.502568

[10] M. Harries and N. S. Wales, "Splice-2 comparative evaluation: Electricity pricing," 1999.

[11] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Fourth international workshop on knowledge discovery from data streams*, vol. 6, 2006, pp. 77–86.

[12] L. I. Kuncheva and C. O. Plumpton, "Adaptive learning rate for online linear discriminant classifiers," in *Structural, Syntactic, and Statistical Pattern Recognition*. Springer, 2008, pp. 510–519.

[13] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in artificial intelligence–SBIA 2004*. Springer, 2004, pp. 286–295.

[14] I. Zliobaite, "How good is the electricity benchmark for evaluating concept drift adaptation," *arXiv preprint arXiv:1301.3524*, 2013.

[15] J. Gama, R. Rocha, and P. Medas, "Accurate decision trees for mining high-speed data streams," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 523–528.

[16] N. C. Oza and S. Russell, "Experimental comparisons of online and batch versions of bagging and boosting," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2001, pp. 359–364.

[17] V. Losing, B. Hammer, and H. Wersing, "Interactive online learning for obstacle classification on a mobile robot," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.