ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

Reactive Soft Prototype Computing for frequent reoccurring Concept Drift

Christoph Raab[1], Moritz Heusinger[1] and Frank-Michael Schleif[1]

1- University of applied Science Würzburg-Schweinfurt - Department of Computer Science, Sanderheinrichsleitenweg 20, Würzburg - Germany

Abstract. Todays datasets, especially in streaming context, are more and more non-static and require algorithms to detect and adapt to change. Recent work shows vital research in the field, but mainly lack stable performance during model adaptation. In this work, a concept drift detection strategy followed by a prototype based insertion strategy is proposed. Validated through experimental results on a variety of typical non-static data, our solution provides stable and quick adjustments in times of change.

1 Introduction

Recent years showing a rapidly increasing amount of data, generated by systems like social media or internet of things. In particular, data is streamed and exceeds the memory and processing capabilities of analyzing systems by far. Hence streaming algorithms are designed with objective to process data as fast as they arrive, in an online manner and without storing large portions of data in main memory. In supervised setting, streams are often affected by change of underlying class distributions, known as concept drift. This results in drops in prediction performance of prior models making them unusable. The detection and handling of these events is one key area in the field of streaming research. Drifts appear differently by means of speed, intensity and frequency, i. e. incremental, abrupt, gradual or reoccurring [1].

The stability-plasticity dilemma [1] defines the trade-off between incorporating new knowledge into models (plasticity) and preserve prior knowledge (stability). This prevents stable performance over time, because on the edge of a drift, major efforts going into learning and testing against new distributions.

In this work we propose a streaming algorithm able to maintain stability during drift while learning new concepts. We discuss prior work on modification of LVQ-prototypes and concept drift detectors in Sec. 2. In Sec. 3, the RSLVQ [2], recently considered as stream classifier [3], is enhanced by a prototype insertion technique and the Kolmogorov-Smirnov (KS)-Test. Subsequently, we refer to this method as Reactive Robust Soft Learning Vector Quantization (RRSLVQ), which is validated by a study in Sec 4. In this study, we focus on frequent reoccurring abrupt drifts streams, not yet widely considered in the literature.

2 Related Work

Concept drift detectors trying to detect change in streams by monitoring their distribution or performance of a classifier. A popular approach for observing

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

performance values [1], the Adaptive Sliding Window (ADWIN)[4], identifies changes in distributions using a window W. ADWIN splits W into two adaptive subwindows and compares underlying statistics. The main window grows as there is no change detected and shrinks if a change between statistics of subwindows is detected. The change is recognized via Hoeffding Bound.

Others methods identify drift not only but mainly via statistical tests on distributions of streams. In [5] two windows are maintained by a randomized search tree, which keeps recent data and the last concept. The KS-Test detects concept drift between windows. Further research is done in [6], where one window stores a snapshot of the concept since last drift and another stores a recent surveyed concept. These KS based approaches are lacking a factor of detecting distribution change at a certain rate.

In context of concept drift handling, techniques can be roughly divided into active and passive [7]. The latter ones use no explicit detection strategy, hence constantly updating the model without awareness of concept drift. Active adaptation changes a model noticeable. By means of LVQ, prototype adjustment is common [8]. In [9], a new prototype is inserted as mean of a set of misclassified examples. The authors of [10] select one sample from given samples as new prototypes which minimizing the error. A near-mean technique is proposed in [8]. It uses the sample as new prototype, which has the smallest Euclidean distance to the mean of a set of misclassified examples. Besides great results in the respective domains, they share the problem of assuming the existence of all classes in a given batch of samples or apply asymmetric prototype insertion. Due to the composition of streams, with potentially unbalanced classes, this assumption is not guaranteed.

3 Reactive Robust Soft Learning Vector Quantization

In supervised classification a stream with potentially infinite length is a sequence $S = \{s_1, \dots, s_t, \dots\}$ of tuples $s_i = \{\mathbf{x_i}, y_i\}$, arriving one s_t at time t. A classifier predicts labels $y_t \in \{1, \dots, C\}$ of unseen data $\mathbf{x}_t \in \mathbb{R}$ by prior model $\hat{y} = h_{t-1}(\mathbf{x}_t)$ and includes this tuple into the model afterwards $h_t = learn(h_{t-1}, s_t)$.

3.1 Concept Drift Detection

Before applying a drift detector, a memory strategy must be defined. The sliding window Ψ keeps n recent points from the stream. It pushes incoming data to the top and removing the oldest one from bottom. For further tests we define two subsets: First, a recent window $R = \{\mathbf{x_i} \in \Psi\}_{i=n-r+1}^{n}$ storing the last r samples. The second should at best represent the current concept and without any assumption on distributions is sampled uniform from the remaining part of Ψ forming $W = \{\mathbf{x_i} \in \Psi | i < n - r + 1 \wedge p(\mathbf{x}) = \mathcal{U}(\mathbf{x}|1, i)\}$ with $|W| = |R| = r$.

Concept drift is the change of joint distributions of a set of samples and corresponding labels between two points in time:

$$\exists \mathbf{X} : p(\mathbf{X}, y)_t \neq p(\mathbf{X}, y)_{t-1} \tag{1}$$

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

The term virtual drift refers to a change in distribution $p(\mathbf{X})$ for two points in time but is also present at real concept drift [1]. Therefore, we assume with the following detector to identify every concept drift through $p(\mathbf{X})$. Obviously, the goal of detectors is to identify any change as soon as possible, as in Eq. (1). Therefore a detector should be monitoring data, rather then performance values.

The Kolmogorov-Smirnov test is a non parametric test inspecting one dimensional data without any assumptions of underlying distributions [11]. It compares the absolute distance between two empirical cumulative distributions $dist_{w,r} = \sup_x |F_W(x) - F_R(x)|$. If this lower bound of maximum distance is greater as the test statistic, then the null hypothesis is rejected, with significance level α, which is the left part of Eq. (2). For two subwindows with the same size, the test is reduced to:

$$dist_{w,r} > c(\alpha)\sqrt{\frac{n+r}{nr}} = \sqrt{-\frac{1}{2}ln\alpha}\sqrt{\frac{n+r}{nr}} \overset{\text{(n=r)}}{=} \sqrt{-\frac{ln\alpha}{r}} \tag{2}$$

Due to the restriction to one dimensional distributions, the test in Eq. (1) applies to all dimensions, therefore at any point t, d tests must be done, because $\mathbb{R}^d$. With this extension and based on Eq. (2) the following can be stated:

Lemma 3.1 *Given* $\mathbf{X}_t$ *and* $\mathbf{X}_{t-1}$ *with* $\mathbf{X}_k = \{\mathbf{x}_j\}_{j=1}^n \in \mathbb{R}^d$, $p(\mathbf{X}) = \prod_{i=1}^{d} p(\mathbf{x}^{(i)})$, $\forall \mathbf{x}^{(i)} \sim$ *i.i.d and their cumulative distributions* F_t,F_{t-1}, *KS-Test detects any change between* $p(\mathbf{X}_t)$ *and* $p(\mathbf{X}_{t-1})$, *i. e.* $\exists \mathbf{x}^{(i)} : p_t(\mathbf{x}^{(i)}) \neq p_{t-1}(\mathbf{x}^{(i)})$, *with probability* $1-\alpha$, *if the difference in distribution is least* $\sqrt{\frac{-ln\alpha}{r}}$.

To implement Lemma 3.1, we set $\mathbf{X}_t = R$ and $\mathbf{X}_{t-1} = W$. When it comes to many statistical tests the problem with multiple hypothesis testing arises, with the consequence of false positives due to random chance. According to Bonferroni-Dunn, we reduce this effect by setting $\alpha = 0.001$ and choosing a relative small test group $r = 30$ and $n = 200$, which increases the required distance. However, remaining false signals are not critical due to the insertion strategy in Sec 3.3. Note that with decreasing p-value α, *dist* increases, and with increasing window size r, *dist* decreases. Hence, they behave competitively w.r.t. *dist* value.

3.2 Robust Soft Learning Vector Quantization

The Robust Soft Learning Vector Quantization [2] is a probabilistic prototype based classification algorithm, capable of online learning. Given a labeled dataset $\mathbf{X} = \{(\mathbf{x_i}, y_i) \in \mathbb{R}^d \times \{1, \ldots, C\}\}_{i=1}^n$ as classification task. The RSLVQ assumes $\mathbf{X}$ can be represented as class dependent Gaussian mixture model and approximates this mixture by a set of m prototypes $\Theta = \{(\boldsymbol{\theta}_j, y_j) \in \mathbb{R}^d \times \{1, \ldots, C\}\}_{j=1}^m$, where each prototype represents a multi-variate Gaussian model, i. e. $\mathcal{N}(\boldsymbol{\theta}_j, \sigma)$. The RSLVQ maximizes the objective function:

$$\mathcal{L} = \sum_{i=1}^{n} log\frac{p(\mathbf{x}_i, y_i|\Theta)}{p(\mathbf{x}_i|\Theta)} \tag{3}$$

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

Where $p(\mathbf{x}_i, y_i|\Theta)$ is the probability density function that $\mathbf{x}_i$ is generated by the mixture model of the same class and $p(\mathbf{x}_i|\Theta)$ is the overall probability density function of $\mathbf{x}_i$ given Θ. Eq. 3 will be optimized with online stochastic gradient ascent (SGA). Note that it converges to an unbiased estimate in the streaming context like in a batch context if no concept drift occurs [12].

3.3 Prototype Adaptation Strategy

In this work, the insertion strategy adapts actively to abrupt or gradual concept drift. We allow missing classes in R, since streams have unbalanced classes and class wise insertion is usually infeasible. If a drift is detected as in Sec. 3.1, window R represents a new context and Θ represents an approximate summary of the last context. The current set of prototypes gets out of date. Therefore, *all*[1] prototypes from Θ will be *replaced* by m new prototypes.

A good starting point by means of approximating an unknown mixture model by Gaussian mixture is the mean of points, i. e. $\boldsymbol{\theta}_{new} = \frac{1}{r}\sum_{\mathbf{x}\in R}\mathbf{x}$ [13]. This is followed by *online* SGA of Eq. 3, optimizing *all* prototypes on given window R. Once more, not all classes must be present in the window R. Further, it can be shown for two prototypes and two classes that the proposed approach leads to a lower error on R instead of taking no action. Therefore, false signals are not critical, because every replacement always leads to lower error and false signals do not harm.

3.4 Time and Memory Complexity

The RRSLVQ optimized via online SGA has the time complexity of $\mathcal{O}(m)$ at given time t for m prototypes without concept drift. The Concept drift handling has the complexity $\mathcal{O}(r \cdot m)$. Note that KS-Test can be implemented in log-linear time [5]. The demand of memory depends on number of prototypes m and sliding window size n. Therefore, the approach needs at maximum $m \times d$ and $n \times d$ as real numbered integers in memory, which accumulates in complexity of $\mathcal{O}(d \cdot (m+n)) = \mathcal{O}(d \cdot (m+200))$ with $n = 200$. For using RRSLVQ in embedded systems with restrictive memory, we follow the *any memory framework* by [1, p. 6] and we suggest to set the number of prototypes plus the window length to a maximum of $d \cdot (m + 200) < k$ with k as maximum integer memory storage capacity.

4 Experiments

We follow the study design from [14]. We use the Sine, Stagger and Mixed stream generator with abrupt (A) and gradual over 1000 samples (G) drift for each generator, so there are six datasets. Each dataset contains one million samples and drift occurs from sample 2000, every 1000 samples after last drift is finished. For a full dataset description see [7]. As a rule of thumb, $m = 6$

[1] Relevance-based removal is very similar in prediction performance in our experiments.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

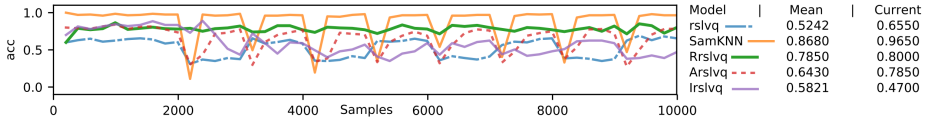

Figure 1: Performance of baseline RSLVQ, RRSLVQ, Incremental-RSLVQ, Adwin-RSLVQ and SamKNN [7] on MixedGenerator stream in SciKit-Multiflow [15]. Plot shows clear drops in performance of non-RRSLVQ methods during abrupt concept drift. From point 2000, drift happens every 1000 points after last drift is finished. Line shows accuracy over last 200 samples. Best viewed in color.

Number Classes/Features	Sine (A) 2/2	Sine (G) 2/2	Stagger (A) 2/3	Stagger (G) 2/3	Mixed (A) 2/4	Mixed (G) 2/4
SAM-KNN [7]	88.8 (22)	88.7 (22)	93.7 (397.23)	93.6 (398)	**85.9** (24)	**85.7** (18)
OzaBaggingAdwin [16]	70.5 (8)	70.9 (8)	81.0 (9)	81.2 (9)	69.2 (10)	69.2 (10)
Hoeffding Adaptive Tree [17]	53.3 (**7**)	54.9 (**7**)	77.2 (**7**)	76.8 (**7**)	53.1 (**8**)	52.7 (**8**)
RSLVQ [13]	49.8 (33)	49.9 (33)	80.6 (33)	80.8 (33)	56.0 (33)	68.0 (33)
RRSLVQ	87.2 (28)	87.1 (28)	83.7 (33)	83.8 (33)	78.6 (33)	78.6 (33)
AdaptiveRandomForest [18]	**88.9** (132)	**89.0** (132)	**96.5** (130)	**96.1** (129)	80.4 (196)	80.1 (194)

Table 1: Interleaved test-then-train accuracy on concept drift streams. Moving average of accuracy on one million samples. Winner marked bold. Required time in minutes shown in brackets.

and $\sigma = 12$ for both LVQ variants. The results[2] are presented in Table 1. Our approach shows a boost in performance to baseline RSLVQ and is comparable to other concept drift classifier. Note that the slightly reduced performance of the RRSLVQ at Stagger can be explained by the categorical features, but RRSLVQ operates in Euclidean space. The RRSLVQ provides stable performance during drift and high adaptation rate shown in Fig. 1, superior w.r.t. other methods.

5 Conclusion

The proposed method is competitive with current state of the art algorithms and is to our best knowledge the first concept drift handling (RS)LVQ algorithm in the streaming setting. Especially, in streams with high rates of drift, the RRSLVQ shows remarkable stability over time. The KS-Test seems to detect occurring changes in data and supports the concept drift handling process with good indicators at given time. Compared to ensemble approaches, it provides a very simple and interpretable model. The memory complexity is easy to bound and well-suited for embedding systems.
Future work should tackle the dimension-wise testing at every time step to avoid unneeded tests. Besides, it should be tested on real-world stream data, with extensive evaluation of methods, not shown due to space limitations.

[2]Note that Kappa statistics are omitted due to space issues.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

References

[1] Joao Gama, Indre Zliobaite, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):1–37, 2014.

[2] Sambu Seo and Klaus Obermayer. Soft Learning Vector Quantization. *Neural Computation*, 15:1589–1604, 2002.

[3] Michiel Straat, Fthi Abadi, Christina Göpfert, Barbara Hammer, and Michael Biehl. Statistical Mechanics of On-Line Learning Under Concept Drift. *Entropy*, 20(10), 2018.

[4] Albert Bifet and Ricard Gavaldà. Kalman Filters and Adaptive Windows for Learning in Data Streams. *Discovery Science*, pages 29–40, 2006.

[5] Denis Moreira dos Reis, Peter Flach, Stan Matwin, and Gustavo Batista. Fast Unsupervised Online Drift Detection Using Incremental Kolmogorov-Smirnov Test. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '16*, (1):1545–1554, 2016.

[6] Christophe Salperwyck, Marc Boullé, and Vincent Lemaire. Concept drift detection using supervised bivariate grids. *Proceedings of the International Joint Conference on Neural Networks*, 2015-Septe, 2015.

[7] Viktor Losing, Barbara Hammer, and Heiko Wersing. KNN classifier with self adjusting memory for heterogeneous concept drift. *Proceedings - IEEE International Conference on Data Mining, ICDM*, 1:291–300, 2017.

[8] Jonathon Climer and Michael J Mendenhall. Dynamic Prototype Addition in Generalized Learning Vector Quantization. *Advances in Self-Organizing Maps and Learning Vector Quantization*, 428:355–368, 2016.

[9] Andreas Zell, Günter Mamier, R Hübner, N Schmalzl, Tilman Sommer, and Michael Vogt. SNNS: An Efficient Simulator for Neural Nets. In *MASCOTS '93, Proceedings of the International Workshop on Modeling, Analysis, and Simulation On Computer and Telecommunication Systems, January 17-20, 1993, La Jolla, San Diego, CA, USA*, pages 343–346, 1993.

[10] Viktor Losing, Barbara Hammer, and Heiko Wersing. Interactive online learning for obstacle classification on a mobile robot. *Proceedings of the International Joint Conference on Neural Networks*, 2015-Septe(2), 2015.

[11] Raul H C Lopes. *Kolmogorov-Smirnov Test*, pages 718–720. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[12] Lianne Ippel, Maurits Kaptein, and Jeroen Vermunt. Dealing with data streams: An online, row-by-row, estimation tutorial. *Methodology*, 12(4):124–138, 2016.

[13] Sambu Seo, Mathias Bode, and Klaus Obermayer. Soft nearest prototype classification. *IEEE Transactions on Neural Networks*, 14(2):390–398, 2003.

[14] Albert Bifet, Jesse Read, and Geoff Holmes. Efficient Online Evaluation of Big Data Stream Classifiers Categories and Subject Descriptors. *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68, 2015.

[15] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. Scikit-Multiflow: A Multi-output Streaming Framework. *CoRR*, abs/1807.0, 2018.

[16] Stuart Oza, Nikunj, Russell and N.C. Oza. Online bagging and boosting. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Waikoloa, Hawaii, USA, October 10-12, 2005*, volume 3, pages 2340–2345. IEEE, 2005.

[17] Albert Bifet and Ricard Gavaldà. Adaptive Learning from Evolving Data Streams. In *Advances in Intelligent Data Analysis VIII*, pages 249–260, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[18] Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9):1469–1495, 2017.