Concept drift detection using autoencoders in data streams processing

Maciej Jaworski$^{1[0000-0002-8410-4231]}$,
Leszek Rutkowski$^{1,2[0000-0001-6960-9525]}$, and Plamen Angelov$^{3[0000-0002-5770-934X]}$

[1] Department of Computational Intelligence, Czestochowa University of Technology
Czestochowa, Poland
{maciej.jaworski,leszek.rutkowski}@pcz.pl

[2] Information Technology Institute, University of Social Sciences
Łódź, Poland

[3] Department of Computing and Communications, Lancaster University
Lancaster, United Kingdom
p.angelov@lancaster.ac.uk

Abstract. In this paper, the problem of concept drift detection in data stream mining algorithms is considered. The autoencoder is proposed to be applied as a drift detector. The autoencoders are neural networks that are learned how to reconstruct input data. As a side effect, they are able to learn compact nonlinear codes, which summarize the most important features of input data. We suspect that the properly learned autoencoder on one part of the data stream can then be used to monitor possible changes in the following stream parts. The changes are analyzed by monitoring variations of the autoencoder cost function. Two cost functions are applied in this paper: the cross-entropy and the reconstruction error. Preliminary experimental results show that the proposed autoencoder-based detector is able to handle different types of concept drift, e.g. the sudden or the gradual.

Keywords: Autoencoder · Data stream mining · Concept drift detection.

1 Introduction

In recent years, the topics connected with data stream mining attracted much attention of machine learning researchers [1], [6], [9–17], [24], [27–34], [37], [38]. In data streams it is quite common that the underlying data distribution can change over time. It is known in the literature as the 'concept drift' [40], [41]. Another characteristics of the streaming data is their potentially infinite size and high rates of arriving at the system. Therefore, proper data stream mining algorithms should be resource-aware [8], [15].

The data stream can be defined as a sequence of data elements

$$S = (x_0, x_1, x_2, \dots). \tag{1}$$

In this paper the case of unsupervised learning is considered, which means that there are no classes assigned to data elements. Hence, the data stream element s_n is a D-dimensional vector defined as follows

$$x_n = \left[x_n^1, \ldots, x_n^D\right]. \tag{2}$$

The data stream mining algorithm should be able to react to possible changes in data distribution [16], [20], [41]. The concept drift handling can be realized in two ways, i.e. passively or actively. In the passive approach, a concept drift reaction mechanism is incorporated in the data stream mining algorithm itself. It can bed done by applying the sliding window techniques [7] or ensemble methods [28], [29]. In the active approach, there is an external drift detector functioning in parallel to the main algorithm. Based on the information from the detector the algorithm can make a decision about rebuilding the model. Many active drift detectors focus on monitoring the current accuracy of the model. If it decreases, then it means that the model becomes invalid and needs to be modified. En examples of such methods are the Drift Detection Method (DDM) [18] and the Early Drift Detection Method (EDDM) [3]. Other active drift detection methods rely on statistical tests. They look for possible changes in parameters of data probability density functions. Among others, the Welch's, the Kolmogorov-Smirnov's [27], or the Page-Hinkley's tests are often used [19].

In this paper, we present an active concept drift detection approach based on autoencoders. This approach is similar to the one presented in [23], in which the Restricted Boltzmann Machine (RBM) [39] was applied as a drift detector. This idea was further extended with resource-awareness strategies to deal with fast data streams [26] or to deal with missing values [25]. Autoencoders are important neural network models often used in the field of deep learning [21]. They are used, for example, in greedy-pretraining methods of learning deep neural networks [4], [5]. Similarly to RBMs, the properly learned autoencoder contains in its middle layer the information about the data probability distribution. It can be then used to check whether the data from another part of the data stream belong to the same distribution as the part on which the autoencoder was trained. This can be checked, for example, by measuring the reconstruction error or the cross-entropy of new data.

The rest of the paper is organized as follows. In Section 2 autoencoders are presented in more detail. In section 3 methods for concept drift detection using autoencoders are proposed. In section 4 the results obtained in numerical simulations are demonstrated. Section 5 concludes the paper and indicates possible future research directions.

2 Autoencoders

The autoencoder is a kind of feed-forward neural network, which is learned to reconstruct input data [22]. There are many types of autoencoders [21], like the denoising autoencoder [2], the sparse autoencoder [35] or the contractive autoencoder [36]. Generally, the autoencoder is composed of two parts: the encoder and the decoder. The aim of the encoder is to transform input data in a nonlinear way, extracting the most important features of the data. The decoder is learned to reconstruct the output of the encoder back to the input data as close as possible. In this paper, we consider autoencoders consisting of $L + 1$ layers. Let $y_j^{(l)}(x_n)$ denote the value of the j-th neuron in the l-th layer obtained for input data x_n. Then, the values of neurons in the l-th layer can be computed using the following formula

$$y_j^{(l)}(x_n) = \sigma\left(\sum_{i=1}^{N_{l-1}} w_{ij}^{(l)} y_i^{(l-1)}(x_n) + b_j^{(l)}\right),\ l = 1, \dots, L, \tag{3}$$

where $w_{ij}^{(l)}$ are weights of synapses between the $(l-1)$-th and the l-th layers, $b_j^{(l)}$ are the biases, and N_l is the size of the l-th layer. Function $\sigma(z)$ is an activation function. Many different types of activation functions can be applied in autoencoder. In this paper, we apply the most common one, i.e. the sigmoid function defined below

$$\sigma(z) = \frac{1}{1 + \exp(-z)}. \tag{4}$$

The 0-th layer is the input layer, i.e. $y_j^{(0)}(x_n) = x_n^j$. The size of the output layer is equal to the input layer size, i.e. $N_L = N_0 = D$. If we do not apply any sparsity constraints while learning the autoencoder, the middle layer should be of the smallest size, i.e.

$$D = N_0 > N_1 > \dots > N_{\frac{L}{2}} < \dots < N_{L-1} < N_L = D \tag{5}$$

The autoencoder is trained as a usual feedforward neural network by minimizing a cost function. Let $C(x_n)$ be the cost function obtained for the x_n data element. Then, for example, a weight $w_{ij}^{(l)}$ would be updated in each step using the following formula

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \eta \frac{\partial C(x_n)}{\partial w_{ij}^{(l)}}, \tag{6}$$

where η is the learning rate. Analogous formula applies for biases.

In practice, the learning is often performed on minibatches of data instead of single data elements. Let us assume that the minibatch M_t consists of B subsequent data

$$M_t = (x_{tB}, \dots, x_{(t+1)B-1}). \tag{7}$$

Then, the neural network parameters are obtained using the arithmetic average of gradients obtained for all data from the minibatch

$$w_{ij}^{(l)} := w_{ij}^{(l)} - \eta \frac{1}{B} \sum_{m=tB}^{t(B+1)-1} \frac{\partial C(x_m)}{\partial w_{ij}^{(l)}}. \tag{8}$$

Many different types of cost functions can be used. In this paper, we consider two of them. The cross-entropy:

$$C_{CE}(x_n) = -\sum_{j=1}^{D} \left[x_n^j \log_2 \left(y_j^{(L)}(x_n) \right) + \left(1 - x_n^j\right) \log_2 \left(1 - y_j^{(L)}(x) \right) \right], \quad (9)$$

and the reconstruction error:

$$C_{RE}(x_n) = \sqrt{\sum_{j=1}^{D} \left(x_n^j - y_j^{(L)}(x_n) \right)^2}. \quad (10)$$

The monitoring of cost function values can be used to detect potential drifts in data distribution. The detection procedure is described in the next section.

3 Concept drift detection with the autoencoder

We suspect that the autoencoder properly trained on one part of the data stream can be applied to monitor possible changes in data distribution in the following parts. For any cost function C (given by (9) or (10), the average value of this cost function for minibatch M_t of size B is given by

$$C(M_t) = \frac{1}{B} \sum_{x_m \in M_t} C(x_m) \quad (11)$$

Apart from the cost function itself, we want to investigate also the trend $Q(C, t)$ of cost function C for subsequent minibatches. To ensure that the trend is computed only on the basis of the most recent data, we apply additionally the forgetting mechanism, as it was done in [23]. The trend at time t is given by the following formula

$$Q(C, t) = \frac{\overline{n}_t \overline{TC}_t - \overline{T}_t \overline{C}_t}{\overline{n}_t \overline{T^2}_t - \left(\overline{T}_t\right)^2}. \quad (12)$$

In standard regression procedure, the terms in formula (12) are givevn simply by appropriate arithmetic averages. In our case, becasue of the forgetting mechanism applied, the arithmetic averages are replaced by the following recurrent formulas

$$\overline{TC}_t = \lambda \overline{TC}_{t-1} + t * C(M_t),\ \overline{TC}_0 = 0, \quad (13)$$

$$\overline{T}_t = \lambda \overline{T}_{t-1} + t,\ \overline{T}_0 = 0, \quad (14)$$

$$\overline{C}_t = \lambda \overline{C}_{t-1} + C(M_t),\ \overline{C}_0 = 0, \quad (15)$$

$$\overline{T^2}_t = \lambda \overline{T^2}_{t-1} + t^2,\ \overline{T^2}_0 = 0, \quad (16)$$

$$\overline{n}_t = \lambda \overline{n}_{t-1} + 1,\ \overline{n}_0 = 0, \quad (17)$$

where λ is a forgetting factor and it takes values lower than 1. The trend of the cost function should be close to 0 as long as there is no concept drift in the data stream. After the drift occurs, then $Q(C,t)$ is suspected to increase to some positive value. Since we consider two cost functions in this paper, there are two trends analyzed as well, i.e. $Q(C_{CE},t)$ and $Q(C_{RE},t)$.

4 Experimental Results

In this section, we present preliminary simulation results of experiments conducted on simple synthetic datasets with concept drift[4]. The dataset was generated in the same manner as it was done in [23], i.e. based on synthetic Boltzmann machines with randomly chosen parameters. At the beginning, we applied two Boltzmann machines, which then were used to generate to datasets with static probability distributions

$$S_1 = (x_{1,1}, x_{1,2}, x_{1,3} \dots), \tag{18}$$

$$S_2 = (x_{2,1}, x_{2,2}, x_{2,3}, \dots). \tag{19}$$

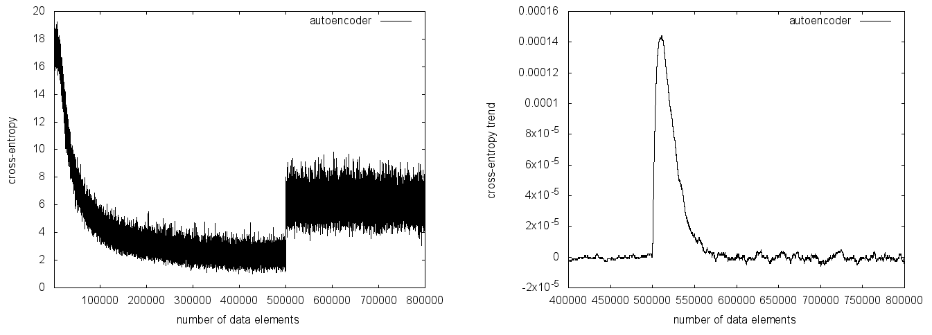
The dimensionality of the data in both sets is $D = 20$. The datasets S_1 and S_2 were used to generate two datasets with concept drifts. The dataset with sudden concept drift is denoted as $S_s = (x_{s,1}, x_{s,2}, \dots)$, where $x_{s,n}$ is taken from S_1 if $n < 500000$ and from S_2 if $n \geq 500000$. The dataset with gradual concept drift is denoted as $S_g = (x_{g,1}, x_{g,2}, \dots)$, where $x_{g,n}$ is from S_1 if $n < 500000$ and from S_2 if $n \geq 600000$. In the interval $[500000; 600000)$ the data element $x_{g,n}$ is drawn from S_1 with probability $\frac{600000-n}{100000}$ and from S_2 with probability $\frac{n-500000}{100000}$. Hence, the two datasets S_s and S_g are constructed in such a way, that the concept drift occurs at after processing 500000 data elements.

In the experiments, we used the simple 3-layered autoencoder ($L = 3$). The sizes of the 0-th and the 2-nd layer are equal to $N_0 = N_2 = D = 20$, to match the dimensionality of the data. The middle layer is chosen to be of size $N_1 = 15$. The learning rate and minibatches sizes are set to $\eta = 0.05$ and $B = 20$, respectively. In each experiment, the autoencoder is trained only for the first 400000 data elements. Then, it is turned to the monitoring mode (no learning is performed, only the values of the cost functions are analyzed). The forgetting factor λ for trend measures was set to 0.998.

4.1 Sudden concept drift detection

The first experiment was conducted on the S_s dataset, with the sudden drift. The results obtained for the cross-entropy loss function are presented in Fig. 1. As can be seen, the value of the cross-entropy decreases while the autoencoder learns until the 400000-th data element. Then it monitors the incoming data. Since the distribution of data does not change until the 500000-th element, the

[4] All the experiments were conducted using our own software implemented in C/C++ and it can be found at www.iisi.pcz.pl/˜mjaworski

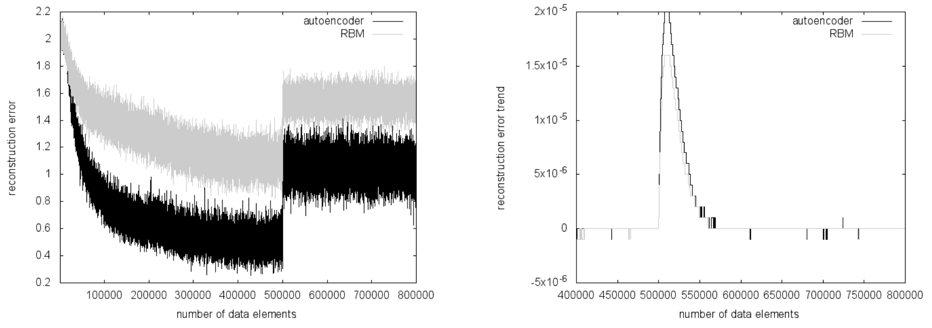


a) values b) trend

Fig. 1. Cross-entropy values and trend as a function of the number of processed data elements, for dataset S_s with sudden concept drift.

cross-entropy values fluctuate around some fixed level. Then the sudden drift occurs and the cross-entropy suddenly achieves higher values. It is more clearly visible in Fig. 1 b), where the trend of cross-entropy values is presented. The peak at $n = 500000$ matches the occurrence of the sudden drift.

In the case of the reconstruction error, the results can be additionally compared with the drift detector based on the RBM, presented in [23]. We applied the same parameters as for the autoencoder. The RBM was learned using the Contrastive Divergence (CD-k) method with K=1. The results are shown in Fig. 2. .



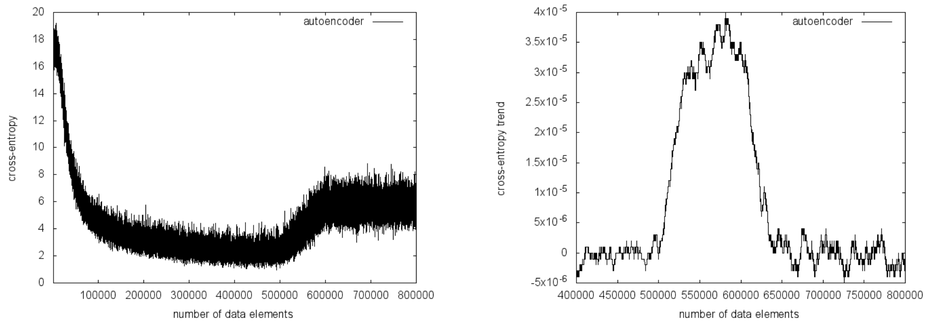
a) values b) trend

Fig. 2. Reconstruction error values and time as a function of the number of processed data elements, for data set S_s with sudden concept drift.

It seems that the autoencoder wins with the RBM approach in all three considered aspects. In Fig. 2 a) we can observe that the values of the reconstruction error are lower for the autoencoder. It might be important if we wanted to use the network simultaneously for other purposes, besides the concept drift detection. In Fig. 2 b) it is visible that the signal peak of suddent change detection is higher for the autoencoder.

4.2 Gradual concept drift detection

In the next experiment, analogous simulations were carried out for the dataset S_g with gradual concept drift. The results concerning the cross-entropy are presented in Fig. 3, whereas in Fig. 4 the comparisons of reconstruction error values are demonstrated.



a) values

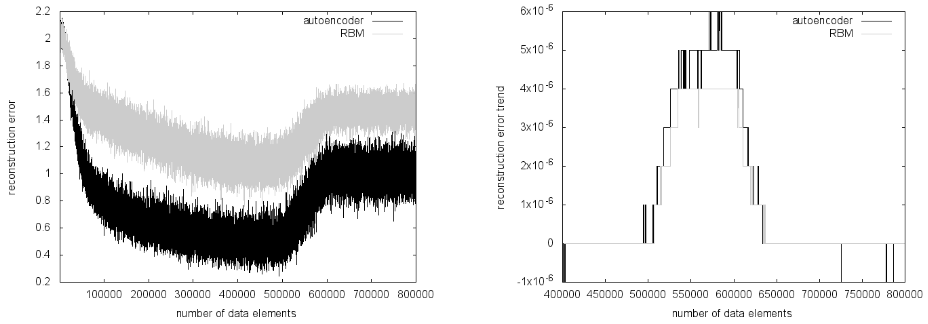
b) trend

Fig. 3. Cross-entropy values and trend as a function of the number of processed data elements, for dataset S_g with gradual concept drift.

In this experiment, the values of cost functions raise smoothly whilst the gradual drift takes place between the 500000-th and the 600000-th data elements. Figure 4 confirms the superiority of the autoencoder over the RBM-based approach also in the case of gradual drift.

5 Conclusions and future work

In this paper, the autoencoder was proposed as a drift detector in time-changing data streams. First, the neural network learns the model on the beginning part of the stream. Then it is used to monitor possible changes in the following parts. The variations of the cost function values are analyzed. If he changes turn out to be large, it means that the concept drift occurred, and it can be a signal for the data stream mining algorithm to rebuild the current model. Two cost functions were applied in this paper, i.e. the cross-entropy and the reconstruction error. The preliminary results obtained in experiments carried out on simple synthetic datasets confirmed that the autoencoder can be successfully used as a concept



a) values

b) trend

Fig. 4. Reconstruction error values and trend as a function of the number of processed data elements, for data set S_g with gradual concept drift.

drift detector. It was demonstrated that it is capable of handling both sudden and gradual concept drifts. In future research, we plan to extend the proposed method to make it more resource-aware for fast-changing data streams and to make it able to deal with missing values.

Acknowledgments

This work was supported by the Polish National Science Centre under grant no. 2017/27/B/ST6/02852

References

1. Aggarwal, C.: Data Streams: Models and Algorithms. Springer, LLC, New York (2007)
2. Alain, G., Bengio, Y.: What regularized auto-encoders learn from the data-generating distribution. J. Mach. Learn. Res. **15**(1), 35633593 (Jan 2014)
3. Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., Morales-Bueno, R.: Early drift detection method. In: Fourth international workshop on knowledge discovery from data streams. vol. 6, pp. 77–86 (2006)
4. Bengio, Y.: Learning deep architectures for ai. Found. Trends Mach. Learn. **2**(1), 1–127 (Jan 2009)
5. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Proceedings of the 19th International Conference on Neural Information Processing Systems. pp. 153–160. NIPS'06, MIT Press, Cambridge, MA, USA (2006)
6. Bifet, A.: Adaptive stream mining : pattern learning and mining from evolving data streams. Frontiers in artificial intelligence and applications, IOS Press, Amsterdam, Berlin (2010)
7. Bifet, A., Gavald, R.: Learning from Time-Changing Data with Adaptive Windowing, pp. 443–448 (2007)

8. Bilski, J., Kowalczyk, B., Grzanek, K.: The parallel modification to the Levenberg-Marquardt algorithm. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing. pp. 15–24. Springer International Publishing, Cham (2018)
9. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proc. 6th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining. pp. 71–80 (2000)
10. Duda, P., Rutkowski, L., Jaworski, M., Rutkowska, D.: On the parzen kernel-based probability density function learning procedures over time-varying streaming data with applications to pattern classification. IEEE Transactions on Cybernetics **50**(4), 1683–1696 (2020)
11. Duda, P., Jaworski, M., Cader, A., Wang, L.: On training deep neural networks using a streaming approach. Journal of Artificial Intelligence and Soft Computing Research **10**(1), 15–26 (2020)
12. Duda, P., Jaworski, M., Rutkowski, L.: Convergent time-varying regression models for data streams: Tracking concept drift by the recursive parzen-based generalized regression neural networks. International Journal of Neural Systems **28**(02), 1750048 (2018)
13. Duda, P., Jaworski, M., Rutkowski, L.: Knowledge discovery in data streams with the orthogonal series-based generalized regression neural networks. Information Sciences **460-461**, 497 – 518 (2018)
14. Dyer, K.B., Capo, R., Polikar, R.: Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. IEEE Transactions on Neural Networks and Learning Systems **25**(1), 12–26 (2014)
15. Gaber, M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: A review. Sigmod Record **34**(2), 18–26 (June 2005)
16. Gałkowski, T., Krzyżak, A., Filutowicz, Z.: A new approach to detection of changes in multidimensional patterns
17. Gama, J.: A survey on learning from data strems: current and future trends. Progress in Artificial Intelligence **1**(1), 45–55 (April 2012)
18. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with Drift Detection, pp. 286–295. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
19. Gama, J., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 329–338. KDD '09, ACM, New York, NY, USA (2009)
20. Gomes, J., Gaber, M., Sousa, P., Menasalvas, E.: Mining recurring concepts in a dynamic feature space. IEEE Transactions on Neural Networks and Learning Systems **25**(1), 95–110 (Jan 2014)
21. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press (2016), http://www.deeplearningbook.org
22. Hinton, G.E., Zemel, R.S.: Autoencoders, minimum description length and helmholtz free energy. In: Proceedings of the 6th International Conference on Neural Information Processing Systems. p. 310. NIPS93, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1993)
23. Jaworski, M., Duda, P., Rutkowski, L.: On applying the Restricted Boltzmann Machine to active concept drift detection. In: Proceedings of the 2017 IEEE Symposium Series on Computational Intelligence. pp. 3512–3519. Honolulu, USA (2017)
24. Jaworski, M., Duda, P., Rutkowski, L.: New splitting criteria for decision trees in stationary data streams. IEEE Transactions on Neural Networks and Learning Systems **29**(6), 2516–2529 (2018)

25. Jaworski, M., Duda, P., Rutkowska, D., Rutkowski, L.: On handling missing values in data stream mining algorithms based on the restricted Boltzmann machine. In: Gedeon, T., Wong, K.W., Lee, M. (eds.) Neural Information Processing. pp. 347–354. Springer International Publishing, Cham (2019)
26. Jaworski, M., Rutkowski, L., Duda, P., Cader, A.: Resource-aware data stream mining using the Restricted Boltzmann Machine. In: Rutkowski, L., Scherer, R., Korytkowski, M., Pedrycz, W., Tadeusiewicz, R., Zurada, J.M. (eds.) Artificial Intelligence and Soft Computing. pp. 15–24. Springer International Publishing, Cham (2019)
27. Lemaire, V., Salperwyck, C., Bondu, A.: A Survey on Supervised Classification on Data Streams, pp. 88–125. Springer International Publishing, Cham (2015)
28. Ludwig, S.A.: Applying a neural network ensemble to intrusion detection. Journal of Artificial Intelligence and Soft Computing Research **9**(3), 177188 (2019)
29. Pietruczuk, L., Rutkowski, L., Jaworski, M., Duda, P.: How to adjust an ensemble size in stream data mining? Information Sciences **381**(C), 46–54 (Mar 2017)
30. Rafajłowicz, E., Rafajłowicz, W.: Testing (non-) linearity of distributed-parameter systems from a video sequence. Asian Journal of Control **12**(2), 146–158 (2010)
31. Rafajłowicz, E., Rafajłowicz, W.: Iterative learning in repetitive optimal control of linear dynamic processes. In: Artificial Intelligence and Soft Computing. pp. 705–717. Springer International Publishing, Cham (2016)
32. Rafajłowicz, E., Rafajłowicz, W.: Iterative learning in optimal control of linear dynamic processes. International Journal of Control **91**(7), 1522–1540 (2018)
33. Rafajłowicz, E., Wnuk, M., Rafajłowicz, W.: Local detection of defects from image sequences. International Journal of Applied Mathematics and Computer Science **18**(4), 581–592 (2008)
34. Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., Herrera, F.: A survey on data preprocessing for data stream mining: Current status and future directions. Neurocomputing **239**, 39 – 57 (2017)
35. Ranzato, M., Poultney, C., Chopra, S., LeCun, Y.: Efficient learning of sparse representations with an energy-based model. In: Proceedings of the 19th International Conference on Neural Information Processing Systems. p. 11371144. NIPS06, MIT Press, Cambridge, MA, USA (2006)
36. Rifai, S., Vincent, P., Muller, X., Glorot, X., Bengio, Y.: Contractive auto-encoders: Explicit invariance during feature extraction. In: Proceedings of the 28th International Conference on International Conference on Machine Learning. p. 833840. ICML11, Omnipress, Madison, WI, USA (2011)
37. Rutkowski, L., Jaworski, M., Pietruczuk, L., Duda, P.: A new method for data stream mining based on the misclassification error. IEEE Transactions on Neural Networks and Learning Systems **26**(5), 1048–1059 (2015)
38. Rutkowski, L., Pietruczuk, L., Duda, P., Jaworski, M.: Decision trees for mining data streams based on the McDiarmid's bound. IEEE Transactions on Knowledge and Data Engineering **25**(6), 1272–1279 (2013)
39. Smolensky, P.: Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chap. Information Processing in Dynamical Systems: Foundations of Harmony Theory, pp. 194–281. MIT Press, Cambridge, MA, USA (1986)
40. Tsymbal, A.: The problem of concept drift: definitions and related work. Tech. Rep. TCD-CS-2004-15, Computer Science Department, Trinity College Dublin, Ireland (April 2004)
41. Zliobaite, I., Bifet, A., Pfahringer, B., Holmes, G.: Active learning with drifting streaming data. IEEE Transactions on Neural Networks and Learning Systems **25**(1), 27–39 (Jan 2014)