Network of Experts: Learning from Evolving Data Streams Through Network-Based Ensembles

Heitor Murilo Gomes[1(✉)], Albert Bifet[1], Philippe Fournier-Viger[2], Jones Granatyr[3], and Jesse Read[4]

[1] University of Waikato, Hamilton, New Zealand
{heitor.gomes,albert.bifet}@waikato.ac.nz
[2] Harbin Institute of Technology, Shenzhen, China
philfv8@yahoo.com
[3] University of Lisbon, Lisbon, Portugal
jones.granatyr@gaips-inesc-id.pt
[4] LIX - École Polytechnique, Palaiseau, France
jesse.read@polytechnique.edu

Abstract. Ensemble classifiers are a promising approach for data stream classification. Though, diversity influences the performance of ensemble classifiers, current studies do not take advantage of relations between component classifiers to improve their performance. This paper addresses this issue by proposing a new kind of ensemble learner for data stream classification, which explicitly defines relations between component classifiers. These relations are then used in various ways, e.g., to combine the decisions of component models. The hypothesis is that an ensemble learner can yield accurate predictions in a streaming environment based on a structural analysis of a weighted network of its component models. Implications, limitations and benefits of this assumption, are discussed. A formal description of a network-based ensemble for data streams is presented, and an algorithm that implements it, named Network of Experts (NetEx). Empirical experiments show that NetEx's accuracy and processing time are competitive with state-of-the-art ensembles.

Keywords: Data stream · Classification · Ensemble learning

1 Introduction

High-speed data stream mining has gained in importance in recent years due to the tremendous amount of real-time data generated by networks, mobile phones and sensors. Building predictive models from data streams is of uttermost necessity for many applications such as those related to the Internet of Things [11]. But designing an effective data stream learning algorithm is not easy as it must process a large number of instances at a fast pace. A key challenge is that an

T. Gedeon et al. (Eds.): ICONIP 2019, LNCS 11953, pp. 704–716, 2019.
https://doi.org/10.1007/978-3-030-36708-4_58

algorithm must learn useful models using limited computational resources. A second challenge is that data evolves, i.e., the underlying data distribution may change over time, resulting in the well known problem of concept drifts [15].

An important data stream mining task is classification. In recent years, classifiers have been proposed to cope with different aspects of data stream classification. An emerging approach is to use classifier ensembles [1,4,18,22,28] as they frequently achieve better accuracy than single classifiers. Moreover, ensembles can often deal with concept drifts in a less drastic way than using a single classifier. For example, a single classifier may discard an hypothesis completely when faced with a concept drift, while an ensemble may only replace (or reset) a few of its component classifiers. Some of the first studies on ensemble classifiers for data streams have focused on adapting existing algorithms to handle data streams. This is the case of Online Bagging and Online Boosting [28]. Many ensembles were designed for data stream classification, some dealing with concept drift explicitly [1,4], and others implicitly [18,19,22].

Most ensemble classifiers are based on the intuition that component classifiers must be diverse to allow their combination to achieve higher accuracy than a single classifier [17]. That is true for many successful ensemble methods such as Bagging [6], AdaBoost [13] and Random Forest [7]. The subjective notion of diversity has been well-studied [8,24,25] and yet there is not a "one size fits all" metric for measuring diversity between ensemble members, or any proof that correlates a given diversity measure and its impact on predictive performance. Even though it is difficult to formalize or measure the contribution of "diversity" to an ensemble's overall prediction accuracy, it is intuitively easy to rationalize why combining an homogeneous set of classifiers cannot achieve better (or worse) accuracy than any of its members. A limitation of current ensemble classifiers is that they do not take advantage of the relations between component classifiers to improve their performance.

This paper addresses this issue by proposing a new kind of ensemble learner for data stream classification, called *network-based ensemble*. Such ensemble explicitly defines relations between members. These relations are then used in various ways such as to combine the decisions of similar classifiers. A structure is imposed on components of an ensemble to highlight their diversity, so that it can be better exploited. An algorithm implementing this idea is presented, named Network of Experts (NetEx). Experiments show that NetEx's accuracy and processing time are competitive with state-of-the-art ensembles.

The rest of this paper is organized as follows. Section 2 defines network-based ensembles. Section 3 describes the proposed NetEx algorithm. Finally, Sect. 4 presents experimental results and Sect. 5 draws the conclusion.

2 Network-Based Ensembles

The proposed concept of network-based ensemble is defined as follows. Let $C = \{c_1, c_2, \ldots, c_M\}$ be a diverse set of classifiers, R **a relation** that defines connections $\Phi = \{\phi_1, \phi_2, \ldots, \phi_P\}$ between members of C, β **a combination**

method that takes into account the structure formed by Φ, and f_ψ **an adaptation function** that updates C and Φ according to the current state of a data stream S. Moreover, it is expected that members of C are different from one another (diverse) to be consistent with the intuitive principle that a homogeneous subset of classifiers cannot contribute to the overall decision any better than any of them alone [23]. We note that for the sake of generality the definition is not bound to any specific method to induce diversity into the ensemble.

The connections defined by the relation R are not restricted to be between pairs of classifiers, although using pairs is an intuitive way of grouping elements in a network [30] and of measuring diversity between classifiers [24]. Also, the relation R is not restricted to be a diversity or similarity measure. The combination method β should use the set of connections Φ to group ensemble members in a way that they can be explored to produce accurate predictions. For example, β can be defined such that any pair (c_k, c_l) from C which connection ϕ_{c_k,c_l} is smaller than a given threshold T must be grouped together for voting. The last component of the proposed definition is the adaptation function f_ψ, which updates the ensemble structure, either periodically or incrementally, to allow it to adapt to drifts. These updates may include adding, removing, or replacing classifiers, and refreshing statistics extracted from classifiers, such as similar predictions counters.

3 The Network of Experts Algorithm

This section presents a novel network-based ensemble, named Network of Experts (NetEx). It relies on an active drift detection strategy instead of relying on a fixed period length parameter as previous approaches [18,19]. The main benefit is that NetEx does not require to fine tune the period length, yet it increases the algorithm's complexity as an adaptive window must be considered for each component model. In SAE2 [19] the relation R was defined as the similarity coefficient (Sc) between a pair of classifiers, connections were activated if they surpassed a Sc_{min} threshold, and a network was induced based on the maximal cliques. SAE2 performed predictions by combining the weighted votes (based on current period accuracy) first at the subnetwork level and then at the network level, which contributes to an indirect drift adaptation technique as recently added classifiers, probably better adapted to the current concept, tend to receive higher weights.

Differently, NetEx defines the relation R as either the Kappa statistic between the output of base model pairs or the Jaccard similarity of the features used to induce the model, and the network is build using the k nearest strategy presented in [29]. NetEx uses an adaptation strategy based on one drift/warning detector per base model, training background learners whenever warnings are detected, and weighting votes based on accuracy calculated on adaptive windows. Also, NetEx uses two diversity inducing techniques: vertical (similar to Leveraging Bagging) and horizontal (random subspaces). The rest of this section describes each aspect of NetEx in details.

Two different similarity weighting functions are presented to define the relation R, namely Kappa statistic κ and Jaccard Index. The reason for using Kappa is because it accounts for agreements that might happen by chance, while also precisely measuring divergence votes on multiclass problems.

Jaccard Index is used to estimate the similarity between finite sample sets [26], and is defined as the size of the intersection divided by the size of the union of the sample sets as shown in Eq. 1, where A and B represents subsets of features used to induce models a and b. The intuition behind using Jaccard to measure the similarity among base models is that models induced using approximately the same features might as well generate very similar models even if online bagging is used. There are other set distance metrics that could be used instead of Jaccard, such as Sorensen-Dice index [12]. However, our problem matches the ideal scenario for applying Jaccard, i.e., it is defined in terms of a binary set membership and element identity (features either belong to the subset or not), and two features are either completely equal or not at all.

$$J(A, B) = \frac{A \cap B}{A \cup B} \tag{1}$$

There are three important factors to take into account when comparing Kappa and Jaccard for measuring similarity in NetEx:

1. Input data to estimate similarity: Kappa is calculated on the output predictions, while Jaccard is calculated on the feature subset;
2. Domain: Kappa ranges from -1 (Inverse dependency) to 1 (Dependency), where 0 represents independency. Jaccard ranges from 0 (No features are shared between models) to 1 (Exactly the same subset or one subset is a superset of the other[1]).
3. Update frequency: To maintain an updated estimation, Kappa must be recalculated after training using each new instance, while Jaccard is updated only when subspaces are defined for each model or when subspaces are reset.

In overall, Kappa provides a more accurate similarity estimation as it is based on the actual outputs. For example, it may happen that a completely different subset of features is used to induce two models, yet the features that compose these subsets may be correlated, thus both models will output very similar predictions. The main concern about using Kappa is that NetEx does not use a fixed update period length to control network updates, thus it is necessary to recalculate Kappa after training using each new instance, which requires a lot of computational resources. Optionally, we could have defined a grace period after which Kappa would be recalculated and the network rebuilt. But we would then be tied to a parameter similar to the period length l of SAE2.

Beyond defining relations, it is also necessary to specify how they will be explored by the ensemble. In this case, how the structure induced by them will be used to boost predictions. In our formal framework this is equivalent

[1] In NetEx, the number of subspaces is fixed, the number of features is the same for all classifiers.

to defining the combination method β. In SAE2, classifiers were combined based on dichotomous connections created based on the Sc_{min} parameter. The goal was to first decide within a set of highly similar classifiers a class label, and then use this decision at a secondary level in which all subsets of classifiers decisions were combined to form the overall ensemble decision.

NetEx uses a similar voting strategy, i.e., it first combines votes within subnetworks and then combines subnetwork votes to obtain an overall prediction. Precisely, when an unknown instance x is to be classified each component model yields one vote weighted by its current estimated accuracy. These individual votes are then combined into an overall subnetwork vote, which is weighted by the average accuracy estimation of its members. This final vote per subnetwork is used to define the overall decision.

The network structure is created based on a variation of the k nearest neighbors network construction technique as proposed in [29]. This method must not be confused with the classical k nearest neighbor learner. In [29] authors present a deterministic approach to construct a network given an arbitrary distance function. Basically, once set a reference vertex, the remaining non-reference vertices are ordered according to the their distance to it. Then, the reference vertex creates a connection with the top k vertices, i.e., closest, from the ordered list.

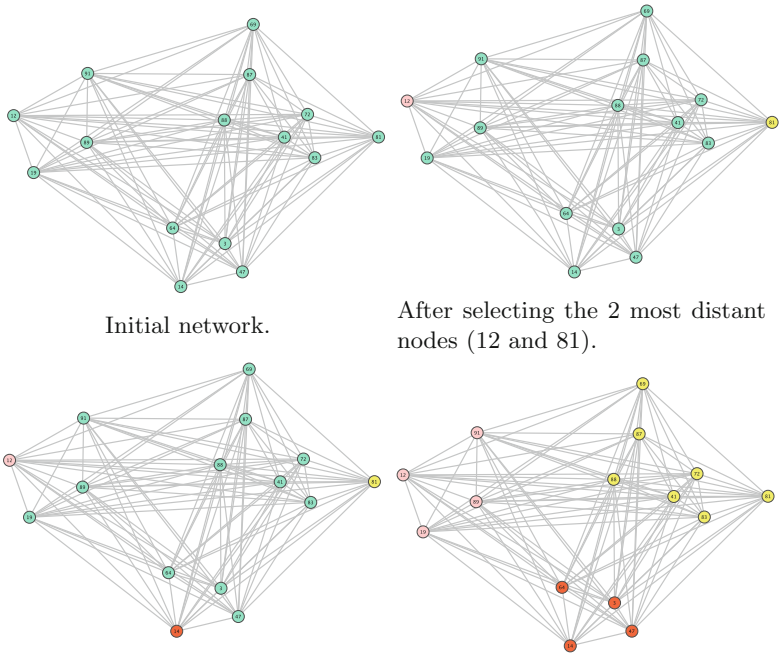
The base algorithm [29] does not specify how the reference vertices are selected. Thus, we have changed it to accommodate a more intuitive network construction approach given our problem. First, we define the k reference vertices, which we name as seed models/nodes, to maximize the overall distance among them. Intuitively, our goal is to create subnetworks as diverse as possible from one another. To do that, we maximize the dissimilarity among seed nodes in an iterative process: first we select the 2 most distant nodes, then the node that is most distant from the previously selected 2, and continue until k is reached. For example, assuming $k = 3$ and that nodes are arranged in Fig. 1 with distance corresponding to their Kappa (or Jaccard) measure, the nodes selected as seeds would be first 14 and 81, and then 12.

There are a multitude of algorithms for finding subgroups on networks [5]. For example, SAE [18] uses weakly connected components to build subnetworks; and [27] which uses a so-called *degeneracy* framework.

This network formation strategy still depends on an hyperparameter (k). However, it is an improvement over SAE2 Sc_{min}'s parameter as it is independent of the connections weight scale. For example, assuming each connection in the example from Fig. 1 were 25% "closer", the resulting subnetworks would be the same.

3.1 The Adaptation Function f_ψ

Following our definition of a network-based ensemble, we have to define the adaptation function f_ψ, responsible for matters such as: how training takes place and when/how the ensemble structure is updated. Our general definition of a network-based ensemble does not explicitly defines a training method, although



Initial network.

After selecting the 2 most distant nodes (12 and 81).

After selecting the most distant node (14) from the previously selected one.

Assigning remaining nodes to the nearest seed node.

Fig. 1. NetEx network formation example.

the definition specifies that component classifiers must be diverse. Thus, this must be taken into account for implementation.

We decided to simulate Online Bagging [28] using a poisson ($\lambda = 6$) distribution as in Leveraging Bagging [1]. Both methods train each model using a randomly selected subset of instances. These methods simulated bootstrap aggregation in an online setting by using a Poisson distribution. The Online Bagging algorithm uses $Poisson(\lambda = 1)$, which means that around 37% of the values output by the Poisson distribution are 0, another 37% are 1, and 26% are greater than 1. This implies that by using Poison (1), 37% of the instances are not used for training (value 0), 37% are used once (value 1), and 26% are trained with repetition (values greater than 1). Leveraging Bagging uses $Poisson(\lambda = 6)$, which implies that 0.25% of values are 0, 45% are lower than 6, 16% are 6, and 39% are greater than 6. Effectively, base models are trained using more instances in Leveraging Bagging, still more resources are used in comparison to Online Bagging.

Besides training classifiers on different subsets of instances, NetEx also trains them on different *subsets of features*. This strategy is known as the Random Subspace Method (RSM) [20]. There are $2^M - 1$ different non-empty subsets of features, which makes it not practical to try every possible combination given

a larger M. However, it is possible to achieve good classification performance in the aggregated ensemble even if only a subset of all possible combinations is explored. The main reason for using random subspaces to train NetEx component classifiers is to enhance diversity among them.

As previously commented, instead of using a fixed window approach for updates, NetEx uses *drift detectors*. Each component classifier is associated with one drift/warning detector. When detector d_j signals a drift warning, then a background learner $c_{bkg}(j)$, is initialized. When d_j outputs the drift signal, then $c_{bkg}(j)$ replaces the current classifier c_j. This approach is similar to that used in Leveraging Bagging [1], although instead of just resetting the classifier, we also start training a replacement beforehand, i.e., when a drift warning is detected similarly to Adaptive Random Forest [16]. NetEx is not bound to a specific drift detection method, however ADWIN [2] is used in the implementation, which has a single parameter that specify the drift confidence level δ. Thus, we have two separate parameters, one for warning detection δ_w and another for drift detection δ_d.

Given our reset strategy based on drift detectors there is no fixed window to estimate accuracy or any other metric. Therefore, to *weight classifiers* we use the estimated accuracy for the given classifier window. This provides a good estimation of the classifier as long as it has seen enough instances, i.e., it will underestimate or overestimate its classification performance if the classifier has seen just a few instances. This is somewhat aided by using background learners as by the time a classifier is added to the ensemble it will have already been trained on a few instances (hundreds or thousands) and will have a good estimation of its accuracy.

It is very difficult to achieve a hyperparameter-free ensemble classifier. For example, we were not able to eliminate the parameter to limit the number of classifiers n and introduced a few other hyperparameters. The following list presents NetEx's hyperparameters accompanied by their short descriptions.

- n defines the total amount of active based models that the ensemble will have at any time. In a stream learning context it is very important to limit the number of classifiers, since memory and processing time are limited assets. n is different from max_c from SAE2, as NetEx starts execution with n component classifiers and maintain it during the whole execution, while in SAE2 max_c defines the maximum number of component classifiers, thus at some point of the execution there might be less than max_c active classifiers.
- k defines the number of seed nodes to build the network. This parameter serves a similar purpose as that from Sc_{min} in SAE2, i.e., it guides connection and subnetwork creation. However, k is easier to set as it is independent from the similarity metric used scale.
- m is the subspace size, which defines the percentage of features randomly assigned to be used for training each base model. A small m increases diversity into the ensemble as it lower the chances that the same subspaces are assigned to the same component classifiers. However, it can decrease performance as low subspace sizes for a high dimensional problem may incur that some important features are never selected.

– δ_d and δ_w represent the ADWIN drift and warning confidence levels. These parameters effectively replace SAE2 l period length, since they define individually the periodicity of when each base model is updated. Effectively, each base model has its own adaptive window.

4 Experiments

We present empirical results comparing ensemble classifiers in both real and synthetic datasets, with and without concept drifts. All experiments were configured and executed using the MOA (Massive On-line Analysis) framework [3]. To evaluate accuracy in all experiments, we applied the Prequential [14] evaluation procedure. The processing time is measured in seconds per MOA CPU time estimation (i.e., it measures the CPU time of the current thread).

The datasets include five real datasets and ten variations of synthetic data streams. The synthetic data contains 1 million instances each, and they simulate either evolving (abrupt, gradual or incremental drifts) or stationary (no drift) streams. Abrupt and gradual drifts are simulate thrice, i.e., one every 250 thousand instances. The data stream configuration is identified by a subscript, e.g., LED_a, such that: {a}brupt, {g}radual, incremental {m}oderate and {f}ast), and {n}o drift (stationary). Table 1 presents a summary of the real datasets used.

Table 1. Real datasets.

ID	# Instances	# Attributes	# Classes
Airlines	539,383	8	2
Electricity	45,312	8	2
Covertype	581,012	54	7
GMSC	150,000	11	2
KDD99	4,898,431	41	23

All experiments in this section use 100 base models[2]; the base learner for all ensembles is the Hoeffding Naive Bayes Tree (HNBT) [21] (grace period = 50 and split confidence = 0.01); specific parameter values for methods other than NetEx were set according to their original publications. Other algorithms used for comparison include the Online Accuracy Updated Ensemble (OAUE) [9], Online Smooth Boosting (OSBoost) [10], Online Bagging (OzaBag) and Online Boosting (OzaBoost) [28], Leveraging Bagging (LevBag) [1] and the Social Adaptive Ensemble algorithm (version 2) (SAE2) [19].

[2] DWM [22] is an exception as it does not include a maximum or target number of base models.

4.1 Jaccard and Kappa Networks

We start the experiments comparing the two connection weighting functions described in Sect. 3, i.e. Jaccard and Kappa. Specifically, we present the results for $k = 5, 10, 20, 30$ using Jaccard or Kappa. As previously mentioned, k stands for the number of seed base models used to create subnetworks according to either Kappa or Jaccard measures. Table 2 presents the results for these experiments.

Table 2. Accuracy - NetEx Jaccard and Kappa varying k. KDD99 did not finish for some variations of Kappa versions, thus KDD99 is not used for the average and ranking calculations.

Dataset	$NetEx_{kap5}$	$NetEx_{kap10}$	$NetEx_{kap20}$	$NetEx_{kap30}$	$NetEx_{jac5}$	$NetEx_{jac10}$	$NetEx_{jac20}$	$NetEx_{jac30}$
LED_a	73.77	73.8	**73.8**	73.78	73.73	73.74	73.76	73.76
LED_g	73.1	73.11	73.1	**73.12**	73.06	73.07	73.09	73.11
SEA_a	89.49	89.49	89.49	**89.49**	89.49	89.49	89.49	89.48
SEA_g	**89.08**	89.07	89.07	89.07	89.07	89.07	89.07	89.07
AGR_a	90.16	90.38	90.66	**90.85**	90.71	90.21	90.27	90.33
AGR_g	86.25	86.63	86.99	87.09	**87.24**	86.48	86.55	86.65
RTS	97.33	97.31	97.06	96.94	**97.39**	97.37	97.36	97.32
RBF_m	86.46	86.47	86.47	86.49	86.36	86.5	86.51	**86.52**
RBF_f	77.11	77.16	77.29	**77.34**	76.9	77.14	77.18	77.22
HYPER	85.08	85.05	85.03	85.08	85.18	85.24	85.26	**85.29**
AIRL	64.94	65.04	**65.14**	65.14	65	64.87	64.92	64.96
ELEC	89.66	89.61	89.65	**89.75**	89.58	89.73	89.67	89.67
COVT	95.11	95.12	95.14	95.12	95.1	95.12	95.14	**95.15**
GMSC	93.55	93.55	93.55	93.55	**93.57**	93.55	93.54	93.55
KDD99	99.96	-	-	-	99.96	99.96	**99.96**	99.96
Avg Rank	5.5	4.54	4.19	**3.08**	4.85	4.85	4.88	4.12
Avg Rank Real	5.63	4	3.63	**3**	5.25	5.5	5.13	3.88
Avg Rank Synt.	5.44	4.78	4.44	**3.11**	4.67	4.56	4.78	4.22

The results in Table 2 suggests that $NetEx_{kap30}$ is the most effective, yet the non-parametric Friedman test indicates that there are no differences among the methods.

4.2 NetEx Compared to Other Ensembles

In this section we compare NetEx against state-of-the-art ensemble learners using prequential accuracy [14].

We highlight NetEx stability in comparison to other ensembles. For example, OAUE obtain good results in general, but fails to obtain a reasonable model for KDD99 (2.45% accuracy while others obtain a minimum of 99.93% accuracy). The same happens for OzaBoost, which obtains the best result for the ELEC dataset (90.67% accuracy), yet obtains the worse results for LED_a, LED_g, RBF_m and RBF_f. Using the same parametrization (besides varying the subspace reset strategy) NetEx may obtain the best results for all datasets considered in

this experiment. This is interesting, since there is a multitude of different problems represented in this benchmark.

We followed these experiments with a non-parametric Friedman test, which indicate that there are significant differences among the evaluated classifiers for these datasets, both when we evaluate all datasets at once and when we conduct the test separately for synthetic and real datasets. Figures 2 presents the results of applying the nemenyi posthoc test to identify the statistically relevant differences (Table 3).

Table 3. Accuracy - NetEx vs. others.

Dataset	*OAUE*	*OSBoost*	*OzaBag*	*OzaBoost*	*LevBag*	*DWM*	*SAE2*	***NetEx***$_{jac30}$
LED$_a$	73.39	72.47	69.04	68.9	**73.95**	71.69	72.53	73.85
LED$_g$	72.58	72.12	68.71	68.54	**73.22**	70.72	72.07	73.16
SEA$_a$	88.8	89.16	87.21	88.25	88.44	86.81	88.94	**89.48**
SEA$_g$	88.19	88.94	87.11	87.92	**89.09**	86.38	88.72	89.07
AGR$_a$	90.16	90.37	83.83	88.12	88.72	76.91	88.17	**91.31**
AGR$_g$	85.24	87.83	79.25	84.66	83.71	76.3	82.4	**87.93**
RTS	96.81	94.78	95.12	96.93	**97.85**	94.76	95.68	97.4
RBF$_m$	84.26	74.51	73.06	65.23	84.34	73.51	65.43	**86.47**
RBF$_f$	57.15	48.7	43.54	26.16	76.77	53.88	39.83	**77.15**
HYPER	**87.8**	86.96	79.37	85.3	85.74	81.04	85.11	85.51
AIRL	65.23	64.56	64.89	60.63	62.82	61.67	59.03	**66.3**
ELEC	87.41	89.51	85.08	**90.67**	89.51	82.19	83.66	89.82
COVT	92.86	92.69	91.49	94.82	95.1	90.03	91.98	**95.18**
GMSC	**93.57**	93.05	93.52	92.32	93.54	92.92	93.46	93.55
KDD99	2.45	99.94	99.93	99.49	**99.97**	99.93	99.88	99.96
Avg Rank	3.47	3.83	6.17	5.87	2.63	6.77	5.6	**1.67**
Avg Rank Synt.	3.2	3.6	6.8	6.2	2.5	6.8	5.2	**1.7**
Avg Rank Real	4	4.3	4.9	5.2	2.9	6.7	6.4	**1.6**

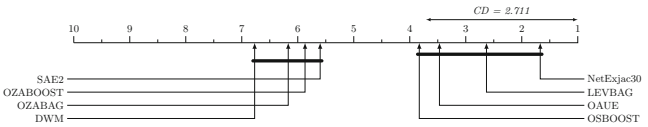


Fig. 2. Nemenyi posthoc with 95% confidence.

Finally, we compare NetEx and the other ensembles in terms of the average CPU time for in Fig. 3. The overall good classification performance of NetEx comes at the expense of a high resources demand. The inefficiency in using computational resources by NetEx is attributable mainly to three aspects of its implementation: (1) it is rare that all base models in the ensemble maintain a background learner at the same time, however in the worst case it is necessary to maintain 2 versions of each base model concomitantly; (2) when a drift is detected it triggers a change in the ensemble, effectively replacing the learner where it was detected by its background learner and causing a recalculation of the

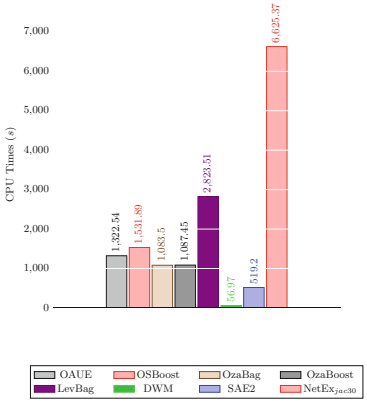


Fig. 3. Average CPU Time for NetEx and others.

network; (3) the subspace reset based on accuracy demands further calculation to estimate each feature probability. For NetEx$_{jac30}$ (2) and (3) are not applicable, since there are no subspace resets in this version and thus it is not necessary to recalculate the network structure, neither perform any estimation on the features.

5 Conclusion

We presented a definition for network-based ensembles for data stream classification and an ensemble learning algorithm based on that definition called the Network of Experts (NetEx). We experimented with NetEx by varying its combination method and including different resetting strategies for the features' subspaces. We compared NetEx against other ensembles for data stream classification and found out that it obtains reasonable results for all datasets. One of the main limitations of the proposed method is the high computational cost, specifically to keep the network structure updated. For future work, we plan to exploit network-based ensembles for semi-supervised learning and other tasks beyond classification problems.

References

1. Bifet, A., Holmes, G., Pfahringer, B.: Leveraging bagging for evolving data streams. In: PKDD, pp. 135–150 (2010)
2. Bifet, A., Gavaldà, R.: Learning from time-changing data with adaptive windowing. In: SIAM (2007)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA data stream mining - a practical approach. Centre for Open Software Innovation (2011). http://heanet.dl.sourceforge.net/project/moa-datastream/documentation/StreamMining.pdf

4. Bifet, A., Holmes, G., Pfahringer, B., Gavaldà, R.: Improving adaptive bagging methods for evolving data streams. In: Zhou, Z.-H., Washio, T. (eds.) ACML 2009. LNCS (LNAI), vol. 5828, pp. 23–37. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-05224-8_4
5. Boccaletti, S., Latora, V., Moreno, Y., Chavez, M., Hwang, D.U.: Complex networks: structure and dynamics. Phys. Rep. **424**(4), 175–308 (2006)
6. Breiman, L.: Bagging predictors. Mach. Learn. **24**(2), 123–140 (1996)
7. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
8. Brown, G., Wyatt, J., Harris, R., Yao, X.: Diversity creation methods: a survey and categorisation. J. Inf. Fusion **6**, 5–20 (2005)
9. Brzezinski, D., Stefanowski, J.: Combining block-based and online methods in learning ensembles from concept drifting data streams. Inf. Sci. **265**, 50–67 (2014)
10. Chen, S.T., Lin, H.T., Lu, C.J.: An online boosting algorithm with theoretical justifications. In: ICML, June 2012
11. Da Xu, L., He, W., Li, S.: Internet of Things in industries: a survey. IEEE Trans. Industr. Inf. **10**(4), 2233–2243 (2014)
12. Dalirsefat, S.B., da Silva Meyer, A., Mirhoseini, S.Z.: Comparison of similarity coefficients used for cluster analysis with amplified fragment length polymorphism markers in the silkworm, bombyx mori. J. Insect Sci. **9**(71), 1–8 (2009)
13. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. ICML **96**, 148–156 (1996)
14. Gama, J., Rodrigues, P.: Issues in evaluation of stream learning algorithms. In: 15th ACM SIGKDD, pp. 329–338. ACM SIGKDD, June 2009
15. Gama, J., Zliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM CSUR **46**(4), 44:1–44:37 (2014)
16. Gomes, H.M., et al.: Adaptive random forests for evolving data stream classification. Mach. Learn. **106**, 1–27 (2017)
17. Gomes, H.M., Barddal, J.P., Enembreck, F., Bifet, A.: A survey on ensemble learning for data stream classification. ACM CSUR **50**(2), 23:1–23:36 (2017)
18. Gomes, H.M., Enembreck, F.: SAE: Social adaptive ensemble classifier for data streams. In: CIDM, pp. 199–206 (2013)
19. Gomes, H.M., Enembreck, F.: SAE2: advances on the social adaptive ensemble classifier for data streams. In: SAC. ACM, March 2014
20. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. **20**(8), 832–844 (1998)
21. Holmes, G., Kirkby, R., Pfahringer, B.: Stress-testing Hoeffding trees. In: PKDD, pp. 495–502 (2005)
22. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: an ensemble method for drifting concepts. J. Mach. Learn. Res. **8**, 2755–2790 (2007)
23. Kuncheva, L.I.: Combining Pattern Classifiers: Methods and Algorithms. Wiley, Hoboken (2004)
24. Kuncheva, L.I., Whitaker, C.J.: Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach. Learn. **51**(2), 181–207 (2003)
25. Kuncheva, L.I., Whitaker, C.J., Shipp, C.A., Duin, R.P.: Limits on the majority vote accuracy in classifier fusion. Pattern Anal. Appl. **6**(1), 22–31 (2003)
26. Levandowsky, M., Winter, D.: Distance between sets. Nature **234**(5323), 34–35 (1971)
27. Nikolentzos, G., Meladianos, P., Limnios, S., Vazirgiannis, M.: A degeneracy framework for graph similarity. In: IJCAI, pp. 2595–2601 (2018)
28. Oza, N.: Online bagging and boosting. In: IEEE SMC, vol. 3, pp. 2340–2345 (2005)

29. Silva, T.C., Zhao, L.: Machine Learning in Complex Networks, vol. 2016. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-17290-3
30. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications, vol. 8. Cambridge University Press, Cambridge (1994)