Clustering-Driven and Dynamically Diversified Ensemble for Drifting Data Streams

Łukasz Korycki
Department of Computer Science
Virginia Commonwealth University
Richmond VA, USA
Email: koryckil@vcu.edu

Bartosz Krawczyk
Department of Computer Science
Virginia Commonwealth University
Richmond VA, USA
Email: bkrawczyk@vcu.edu

***Abstract*—**
Data stream mining is a rapidly developing branch of contemporary machine learning. Ensemble approaches have proven themselves to be highly effective in this domain, due to their predictive power and capabilities for handling evolving data. One of the key aspects of ensemble learning is diversity among base classifiers – it improves accuracy and allows for anticipating and recovering from concept drifts. It has been shown that while diversity is desirable during changes, it may impede learning when data becomes stationary. In this paper, we present a novel ensemble technique that exploits the idea of dynamic diversification, which increases diversity during changes and reduces it when a stream becomes stable. The algorithm uses online clustering for this task by creating locally specialized base learners trained on spatially related instances. Three control strategies based on the novel *range* heuristic for managing a trade-off between error (a change indicator) and diversity are utilized. Additionally, two intensification strategies are proposed for exploitation of newly arriving instances, allowing for faster adaptation. Experimental study evaluates the general performance and diversity of the proposed algorithm, proving its capabilities to outperform state-of-the-art ensembles dedicated to drifting data stream mining.

***Keywords*-machine learning; data stream mining; classification; concept drift; ensemble learning; diversity.**

I. Introduction

Contemporary real-life problems are characterized by the massive volume of information that is continuously generated, leading to a phenomenon known as data flood. Machine learning systems must be capable of scaling-up to such large and evolving data sets. This has lead to the notion of data streams, which pose significant challenges to learning systems and thus require dedicated solutions [1].

Data stream is a potentially unbounded sequence $\langle \mathbf{X_1}, \mathbf{X_2}, ..., \mathbf{X_n}, ...\rangle$, in which each element $\mathbf{X_i}$ is a set of instances (or a single instance in a case of online setting). In learning from data streams we are not only concerned with the predictive accuracy, but also with the speed of adaptation and usage of computational resources [1]. Additionally, characteristics of incoming instances (such as class boundaries, distributions, or number of classes) may be subject to change over time. This is known as concept drift and must be appropriately handled, or the quality of classifier will deteriorate. There are two main solutions allowing for tackling concept drift: drift detectors (external mechanisms that monitor the properties of stream) and adaptive learners (such as sliding windows or incremental classifiers).

Ensemble learning for data stream mining gains an increased attention [2], taking advantages of the mentioned solutions and using multiple classifiers at the same time [3]. Ensembles have been shown not only to improve predictive accuracy, but also to efficiently handle concept drift by replacing outdated classifiers in the pool [4].

Diversity is known as one of the key factors influencing the ensemble performance. In general, it is desirable to create mutually different, yet complementary, base learners to expand the overall competency of the whole committee. One group of methods used to diversify ensembles are region-based algorithms that train dedicated classifiers for different subspaces of data. For static data sets, Lee and Kim [5] proposed the eSVM ensemble, dividing the data set into heterogeneous parts used by base learners. It is also possible to generate clusters for each class or even split it internally to create an ensemble of one-class models [6]. Even though a plenty of works on diverse ensembles exist, there are still doubts how diversity should actually be defined or whether the relation between it and performance is always the same [7]. Existing works on diversity measures show that they are efficient monitoring tools, but fail as a driving force behind the ensemble creation process [8]. In the data stream context, having diverse learners increases the chance that at least one of them will retain its competence, or will have short recovery time, after concept drift [9]. However, when the stream displays stationary characteristics, high diversity may harm the predictive performance of the ensemble. Therefore, a proper control of diversity factor is a key issue.

In this paper, we propose a new ensemble learning scheme that allows for a dynamical diversification of base classifiers. It uses an online clustering approach to create locally specialized classifiers trained on chunks of spatially related instances. Diversity management is controlled by a drift detector that allows to dynamically change the diversification level according to the current state of the stream. Extensive experimental study on synthetic and real data stream benchmarks shows efficacy of the proposed ensemble and offers an insight into the role of diversity in data streams.

II. Ensemble diversity for data streams

The idea of adapting diversity to streaming data environments is not a recent one, however, the coverage of it is still surprisingly limited. The first and most interesting work considering diversity and evolving data streams was presented by Minku [9]. It posed the important question – how diversity may influence learning from drifting streams. The main conclusion of the work is that high diversity improves recovery from concept drifts, but on the other hand, it usually impedes the learning process during stable periods. Additionally, the work points out that the more severe is concept drift, the higher diversity is needed. To control the level of diversity, authors dynamically manipulated the Poisson distribution used by the online bagging algorithm [10]. Lower values of λ provided higher diversity, while higher values of the parameter were suitable for stable periods. Minku and Yao [11] developed the presented idea and created a more sophisticated algorithm (DDD), focusing on maintaining high-diversity ensemble for concept drifts and low-diversity one for stable substreams.

While ensemble diversification seems as a valuable direction in data stream mining, it must be done in a controlled fashion. It is remarkable that although there are some works presenting ensembles promoting diversification, almost none of them investigate the diversity measures explicitly [12].

Brzezinski [13] presented a work that emphasized the lack of diversity analysis in data streams domain. He investigated six well-known diversity measures, using different ensembles and drifting data streams, showing that they can be easily used in such settings and analyzing the impact of changes on the measures.

III. Proposed algorithm

We present a first attempt to build a clustering-driven ensemble that promotes dynamic diversity for drifting data streams. Based on the published observations concerning diversity and concept drifts, we dynamically adjust it accordingly to the current state of a stream. Algorithm 1 presents the general framework implementing the clustering-based idea, while in the next paragraphs we discuss each of its components in details.

Ensemble. We use instances $\mathbf{x_i}$ to update the core modules of our algorithm – a set of clusters $\mathbf{C_i}$ and an ensemble $\mathbf{L_i}$ (we maintain one of each for a whole stream). To create a diverse ensemble we utilize one classifier per cluster. Whenever a new cluster is formed, so the number of clusters increases $|\mathbf{C_i}| > |\mathbf{C_{i-1}}|$, a new base learner is added to the ensemble. Each classifier learns only from data belonging to its cluster. This space partitioning creates classifiers specialized in recognizing different parts of data, so in general, they should be diverse, yet mutually complementary. We combine decisions of the base learners using accuracy weighted majority voting. One must also remember that the base

Algorithm 1: The online clustering-driven ensemble using dynamic diversification.

```
Data: ensemble size $e$, ClusteringStrategy ($parameters$),
      DriftIndicator ($parameters$), ControlStrategy
      ($parameters$)
Result: ensemble $\mathbf{L_i}$ at every iteration $i$
Initialization: $i \leftarrow 1$, $\mathbf{L_0} \leftarrow []$, $\mathbf{C_0} \leftarrow []$
repeat
    receive incoming instance $\mathbf{x_i}$;
    request the true label $y$ of instance $\mathbf{x_i}$;
    update clusters ClusteringStrategy($\mathbf{C_{i-1}}, \mathbf{x_i}, e$);
    if $|\mathbf{C_i}| > |\mathbf{C_{i-1}}|$ then
        add new classifier $l$ to $\mathbf{L_{i-1}}$
    $d \leftarrow$ DriftIndicator();
    $r \leftarrow$ ControlStrategy($d$);
    $\mathbf{I_c} \leftarrow$ indices of $r|\mathbf{C_i}|$ centroids closest to $\mathbf{x_i}$;
    for $j \leftarrow 1$ to $len(\mathbf{I_c})$ do
        update classifier $\mathbf{L_i}[\mathbf{I_c}[\mathbf{j}]]$ with $(\mathbf{x_i}, y)$;
    $i \leftarrow i + 1$;
until stream ends;
```

learners have to be able to handle concept drifts, since the framework improves adaptivity, but it does no provide it itself.

Clustering strategy. The clusters are actualized in an online manner. The incoming instance $\mathbf{x_i}$ is assigned to a cluster based on `ClusteringStrategy` used. Currently, there are few such algorithms available, even less tries to tackle data streams problems, like evolving concepts or anomaly detection. In our framework we use an online k-means algorithm [14], which has been shown as a competitive to the offline k-means++ algorithm. However, it tends to find slightly more clusters than a given parameter e. The method generates clusters in an online manner, but it does not use any direct mechanism to handle changes in data, so the approximation of the optimal solution may be impeded. It is not a crucial aspect in our case, since we focus mainly on the changes in the conditional probability, rather than on data distribution, therefore, we do not need very precise clusters. Nevertheless, the algorithm will modify its centroids incrementally starting from near-optimal seeds. To boost this process we use a windowing mechanism by applying moving average to centroid's features. Our experiments show that such approach is sufficient in practice.

Drift indicator. After updating the clusters, we collect the measure of concept drift d calculated by `DriftIndicator`. There are several online change detectors already published, many of them are based on statistical tests and error measures. They are usually dedicated for different types of concept drift. In our framework, we use the windowed error ϵ_ω of size ω as a drift indicator, calculated using the moving average method. It is assumed that the error increases when concept changes, since classifiers need time to adapt to the drifting data. One must remember that the indicator is very sensitive to the window size and access to labeled instances.

Control strategy. The drift indicator d is used to control the value of the range variable $r \in \langle 0, 1 \rangle$, by applying `ControlStrategy`. The strategy defines a relation (trade-off) between d and r. The range variable determines how many classifiers use a newly arriving instance. First, Euclidean distances between the object and all centroids are calculated. Then, indices of the $r|\mathbf{C_i}|$ closest centroids are selected and classifiers paired with them are updated. The varying range of classifiers controls the level of diversity among base learners. The idea is as follows – when concept drift appears the error increases and r decreases. As a result, instances are used only by the closest classifiers and they start differentiating from each other. When a concept becomes stable, the error decreases and r increases. This leads to the situation in which base learners more frequently learn from the same instances, therefore, diversity should decrease in favor of variance reduction.

The work [9] indicates that dosing different amounts of diversity is beneficial, however, it does not determine how high or low the diversity should be. The linear relation is probably the most straight-forward one. We define three different control strategies using a sigmoid function $\sigma(x) = x(\beta - 1)/(2\beta x - \beta - 1)$, where $x = 1 - \epsilon$ and $\beta \in \langle 0, 1 \rangle$ controls the shape of the curve representing relation between error (ϵ) and range (diversity).

- `LINEAR` ($\beta = 0$) – it maintains the linear relation between error and diversity, so any change in the former will result in directly proportional change in the latter.
- `STABLE` ($\beta < 0$) – diversity increases slower than error, so this approach promotes low-diversity ensembles.
- `DIVERSE` ($\beta > 0$) – it is opposite to the previous strategy, so it promotes higher-diversity committees.

Intensity. The algorithm presented above is a basic version. Since we want our framework to be able to adapt to varying speed of changes, we may need to somehow intensify the diversification process. One should be aware of the fact that when the drift appears abruptly and we have access to but few instances from the new concept, the diversification method, based on a single instance usage, may be insufficient for changing diversity effectively. We propose a simple improvement which aims to boost the process by using a single instance several times. We consider two approaches.

- `FX` – it uses a fixed number of duplicates regardless a state of a stream. The idea is to equally intensify both increasing and decreasing diversification.
- `ER` – the strategy increases the number of duplications if an error is higher, so it promotes intensification when concept drift occurs.

IV. Experimental study

A. Evaluation methods

In our evaluation section we aim to investigate three major properties of our method.

The first one is **general performance** during drifts (in tables: Drifts) and stable periods (Stable). We report the average values of prequentially calculated accuracy and Cohen's kappa. For stable substreams we also present standard deviations (StdS) that indicate whether performance during them is indeed stable.

Secondly, we investigate the ability of our method to generate **diversity** during concept drifts and to reduce it when stable periods appear. We explicitly measure 2 different diversity measures: disagreement (D) and double fault (DF) to show that our method of diversification effectively manages it as intended. We present time series of the metrics along with classification performance, which is, in fact, our drift indicator.

Finally, we evaluate our framework **on real data streams** to find if it is able to compete with other well-known ensembles. Since currently we cannot define concept drifts for most of the real streams, we investigate the algorithms in general, using prequential accuracy and kappa. Even if the streams are not as predictable as the synthetic ones, they are surely more reliable, since we still do not know how authentic data generators are.

B. Data stream benchmarks

Using MOA [15], we generated 14 artificial streams based on 5 different synthetic concepts and consisting of different types of concept drift. They are summarized in Tab. I.

Table I: The summary of the used synthetic data streams.

Name	Inst	Attr	Cls	First	Dist	Width	Drifts	Noise
SEA1	600k	3	2	150k	300k	100	3	0.05
SEA2	600k	3	2	150k	300k	10k	3	0.05
STAG1	600k	3	2	150k	300k	100	3	-
STAG2	600k	3	2	150k	300k	10k	3	-
RBF1	1m	15	5	250k	250k	100	3	0.05
RBF2	1m	15	5	250k	250k	10k	3	0.05
RBF3	1.2m	15	5	400k	400k	50k	2	0.15
RBF4	1.2m	15	5	400k	400k	100k	2	0.15
TREE1	1m	15	5	250k	250k	100	3	0.05
TREE2	1m	15	5	250k	250k	10k	3	0.05
TREE3	1.2m	15	5	400k	400k	50k	2	0.15
TREE4	1.2m	15	5	400k	400k	100k	2	0.15
HYPER1	500k	15	5	-	-	-	-	0.01
HYPER2	500k	15	5	-	-	-	-	0.01

The HYPER1 stream was generated using a magnitude of change $t = 0.001$ and HYPER2 using $t = 0.01$.

For the final evaluation we used 15 real streams in total. Most of them (12) are pure real streams, while 3 (Hepatitis, Lymph and Wine) were generated on the basis of real data sets coming from UCI. The nature of the data streams indicates that they very likely consists of concept drifts. They are presented in Tab. II.

C. Set-up

Configurations. We evaluate our framework using different strategies for controlling diversity – linear (CL), diversity-oriented (CLD) or stability-oriented (CLS), as well as,

Table II: The summary of the used real data streams.

Name	Inst	Attr	Cls
Activity	10 853	43	8
ActivityRaw	1 048 570	3	6
Airlines	539 383	7	2
Connect4	67 557	42	3
Covertype	581 012	54	7
DJ30	138 166	8	30
Electricity	45 312	8	2
Gas	13 910	128	6
Hepatitis	1 000 000	20	2
Lymph	1 000 000	19	4
Poker	829 201	10	10
Sensor	2 219 804	5	54
Spam	9 324	499	2
Weather	18 159	8	2
Wine	1 000 000	658	2

without dynamic diversification (static SCL). We check how intensification methods using a fixed (CL+FX) and error-driven (CL+ER) number of duplications influence performance. Finally, we combine the strategies and evaluate them on synthetic streams. Due to limited space, on real streams we evaluate only two configurations (CLD+ER and CLS+ER).

Parameters. For the online k-means strategy we selected $e = 10$ as a number of target clusters. It determines the desired size of an ensemble, however, it may usually vary between 10 and 15 base learners [14]. Since we want to adapt to various speed of changes, including sudden ones, for windowed clusters and error indicators we use size $\omega = 100$ to make it reactive. For the `FX` intensification strategy we set $n = 20$ as a number of duplications per instance. As base learners we selected Adaptive Hoeffding Trees (AHT).

Other ensembles. During the detailed analysis of our framework on synthetic streams we compare it with another algorithm that promotes dynamic diversity – online bagging based on [9]. We control λ using the same control strategies as in our solution (BAG, BAG-D and BAG-S). On real streams we compare our two most promising configurations with the bagging algorithms and 6 well-known and widely cited online ensembles: Learn++.NSE (LNSE++), Dynamic Weighted Majority (DWM), Accuracy Updated Ensemble (AUE), Adaptable Diversity-based Online Boosting (ADOB, recently added to MOA), Online Smooth Boosting (OSB) and OzaBag-ASHT (OB-AHT), which is a basic online bagging algorithm using AHT and maintaining static diversity among them. For all ensembles we selected $e = 10$ for their size. The rest of parameters are set to default values and base learners are AHT.

Measurements. All presented series results (accuracy and kappa) were collected using the prequential evaluation and window which size was $\omega = 1000$. We distinguish results for concept drifts and stable periods by calculating averages separately. For each change j that has its peak at p_j and width w_j we treat results within $i \in \langle p_j - w_j/2, p_j + w'_j \rangle$ as measurements for drifts. We set $w'_j = w_j/2$ for $w \geq 20000$ and $w' = 10000$ for $w < 20000$, since we treat concept drift as something relative to the model performance and drops in effectiveness can be longer than an actual period of concept change, especially a short one. To make the evaluation setting more realistic, we simulated a limited access to labeled data. We provided a warm start on each data stream using 10% of it, as fully labeled initialization batch, and then only 10% of randomly selected instances from the remaining part.

D. Results

Obtained results clearly show that our diversification heuristic is effective and that the whole framework, if only properly configured, can effectively adapt to changes. Different strategies and combinations evince different behaviors, but only some of them can be established as the proper ones. In general, differences in results for kappa are greater than for accuracy. Below we discuss all the results, showing why certain approaches are better than the others.

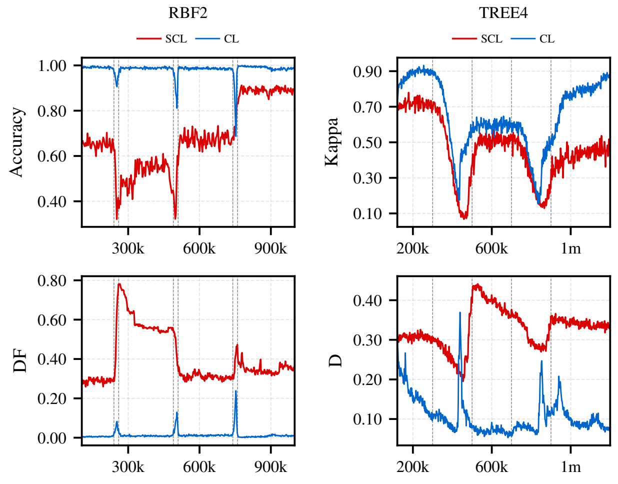

Figure 1: Performance and diversity series for RBF2 (left) and TREE4 (right), using different strategies.

Static vs. dynamic diversity. First of all, we can see in Fig. 1 that when static diversification is used (SCL), the diversity level changes not adequately to the error (concept drift, between dotted lines) and the adaptation of the framework may be very unstable. At the same time, once we add the linear control mechanism (CL), the reaction to changes is significantly better adjusted. The dynamic range effectively increases diversity just after a drift occurs and, as a result, leads to faster reduction of an error. One may look a bit closer to see that the error stops increasing once diversity is high enough and that recovery begins subsequently. At the same time, when the error is getting low, the diversity level is being reduced, leading to stabilization. The same observation is valid for all results we describe in the next paragraphs.

In the general summary (Tab. III), one can see that CL outperforms the static SCL during drifts and stable periods. These observations allows us to conclude that our method is an effective mechanism for generating dynamic diversification. We made the comparison, since for some ensembles it is possible that they increase their diversity itself, when a drift occurs [13]. It does not happen in our case.

Table III: The summary of synthetic streams results for accuracy (left) and kappa (right).

Algorithm	All	Drifts	Stable	StdS	All	Drifts	Stable	StdS
SCL	0.6753	0.5331	0.6519	0.0808	0.4833	0.2459	0.3477	0.1251
CL	0.8838	0.7221	0.9197	0.0420	0.8143	0.5615	0.8594	0.0734
CLD	0.8453	0.6569	0.8726	0.1010	0.7599	0.4625	0.7941	0.1580
CLS	0.8866	0.7564	0.9228	0.0297	0.8224	0.6052	0.8694	0.0573
CL+FX	0.8227	**0.7856**	0.8539	**0.0246**	0.7168	0.6268	0.7360	**0.0492**
CL+ER	**0.9028**	0.8258	**0.9411**	0.0272	0.8434	**0.7042**	0.8957	0.0526
CLD+FX	0.8241	0.7628	0.8550	0.0267	0.7164	0.5907	0.7353	0.0557
CLD+ER	0.8925	0.7855	0.9297	0.0352	0.8288	0.6365	0.8764	0.0674
CLS+FX	0.8042	0.7622	0.8369	0.0314	0.6848	0.5891	0.7069	0.0573
CLS+ER	0.8981	0.8247	0.9379	0.0283	**0.8444**	0.7026	**0.8983**	**0.0492**
BAG	0.8410	0.6989	0.9097	0.0481	0.7508	0.5291	0.8473	0.0796
BAG-D	0.7645	0.6468	0.8055	0.1062	0.6223	0.4354	0.6876	0.1606
BAG-S	0.8507	0.7499	0.9199	0.0298	0.7680	0.6164	0.8649	0.0548

Change-diversity trade-off. The results for different relations between the drift indicator and diversity show that, in general, stability-oriented or at least linear approaches improve adaptation, while strategies focusing on increased diversity tend to impede it (Tab. III). For both accuracy/kappa when our framework and the bagging algorithm were promoting higher diversity (CLD and BAG-D) during the whole learning process, they worked worse than the configurations using the linear (CL and BAG) or stability-oriented control (CLS, BAG-S). The latter achieved better results not only for stable periods (about 0.92/0.87 for CLS), than the former (0.87/0.79 for CLD), but also during drifts (about 0.75/0.61 for CLS, 0.66/0.46 for CLD). Differences for the bagging approaches were even bigger than for our framework. We can also notice that the high-diversity strategies were much more unstable during pure concepts (about 0.10 for accuracy standard deviation and 0.16 for kappa) than stability-based ones (about 0.3 and 0.6 respectively). The latter were also slightly more stable than the linear approaches.

We can see in Fig. 2 and 3 that all high-diversity ensembles were, indeed, maintaining higher values of disagreement. It could be beneficial during drifts, however, CLD and BAG-D algorithms struggled with reducing the high level of diversification after drifts. It usually resulted in prolonged adaptation, even if drops in performance were sometimes on the same level (SEA1, STAGGER2). The linear and stability-based methods maintained lower diversity even during drifts, but at the same time, they were stabilizing learning much more quickly. CLS was able to trigger very precise diversification just around drift peaks on STAGGER2 and TREE3, as well as, to significantly reduce maximal loss on TREE3 and RBF4.

Applying intensification. While considering intensification improvements, there is a clear indication that the strategy using the error-driven heuristic (CL+ER) is more reasonable in practice than the approach which tries to intensify both diversification and stabilization (CL+FX). We can see in

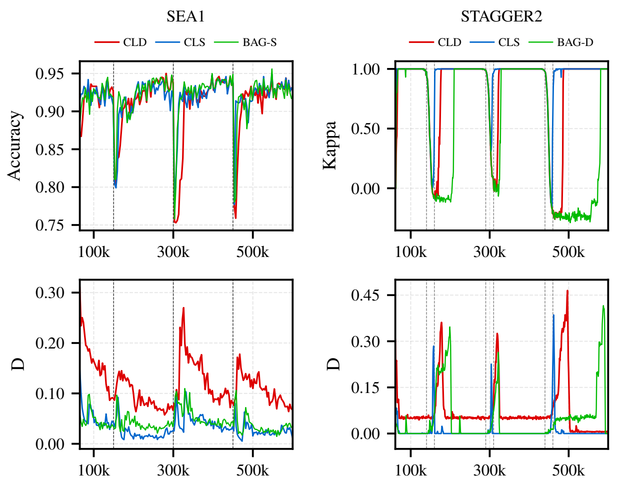

Figure 2: Performance and diversity series for SEA1 (left) and STAGGER2 (right), using different strategies.

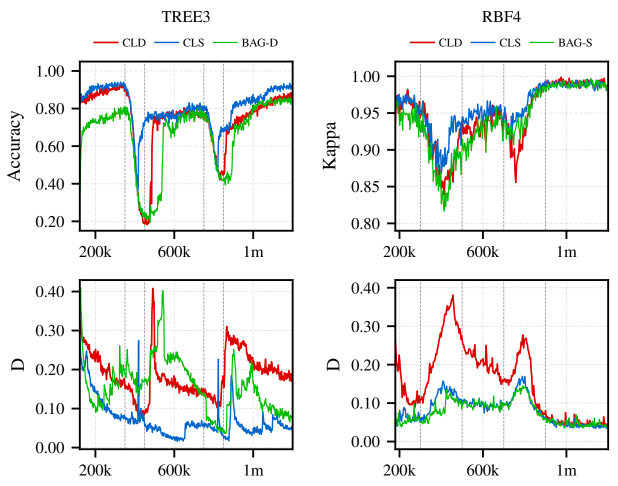

Figure 3: Performance and diversity series for TREE3 (left) and RBF4 (right), using different strategies.

Tab. III that although CL+FX slightly improves recovery from drifts (0.79/0.62) over basic CL (0.72/0.56) it impedes the performance of our framework during stable periods (0.85/0.74 and 0.92/0.86, respectively). Since the latter are longer than the former, the overall performance is also worse (0.82/0.72 for CL+FX and 0.88/0.81 for CL). In Fig. 4 and 5 we can see that CL+FX disturbed the diversity control, making it almost unresponsive to drifts. On the other hand, CL+ER was maintaining relatively low diversity during stable periods, similarly to basic CL, and at the same time, it was able to intensify recovery from drifts by increasing diversity even more precisely. One can notice that diversification peaks were concurring with the lowest performance drops on TREE1, RBF2 and TREE4.

CL+ER provides significant improvements over basic CL mainly during drifts (0.83/0.70). Improvements are less significant for stable periods. The FX strategy obtained the lowest standard deviation, since better approaches were

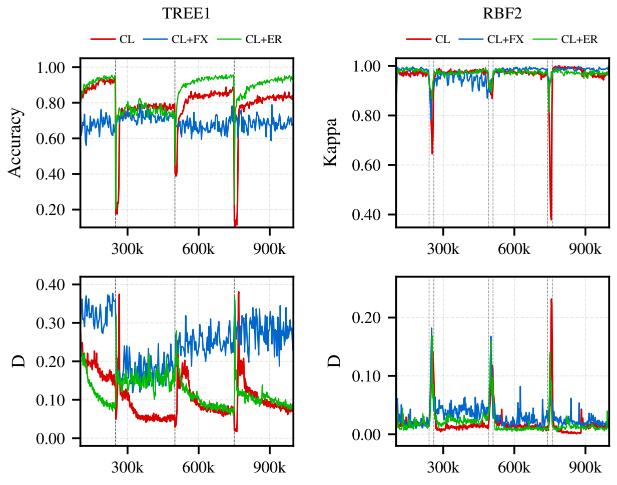

Figure 4: Performance and diversity series for TREE1 (left) and RBF2 (right), using different strategies.

constantly improving their performance, leading to bigger differences between the lowest and highest values, even during stable periods.

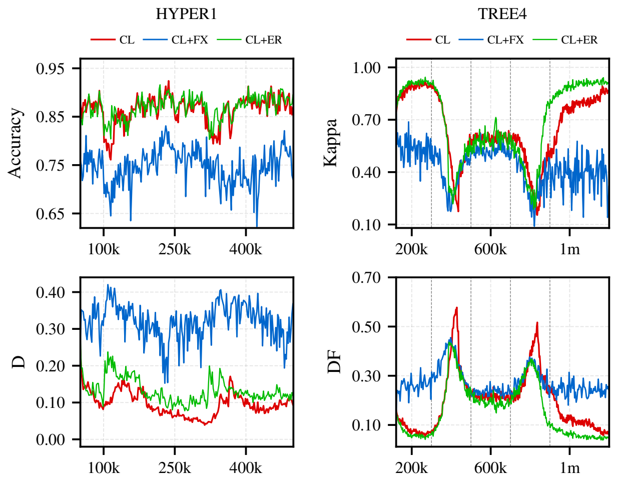

Figure 5: Performance and diversity series: for HYPER1 (left) and TREE4 (right), using different strategies.

Combinations. We can see that results for combinations of the two mentioned mechanisms reflect the relations between these methods and that the intensification strategies are the main factor behind obtained improvements. Firstly, for both CLD and CLS heuristics differences between FX and ER were significant (about 0.07-0.09/0.11-0.16) as for CL+FX and CL+ER. Secondly, differences between FX and ER strategies combined with low or high diversity approaches were minor, similarly to CLS and CLD. Thirdly, one can notice that adding ER slightly reduced the gap between CLD and CLS, as well as, that FX even reversed the relation between the control strategies. Finally, out of two best combinations, CLS+ER was better than CLD+ER regarding performance during drifts (0.82/0.70 and 0.79/0.64, respectively in Tab. III). Nonetheless, they were still on a similar level as simpler CL+ER. In Fig. 6 we can see that the FX heuristic was disturbing the adaptation process regardless of a control strategy used (SEA2) and that, depending on the strategy, ER worked better or worse than BAG-S (TREE2).

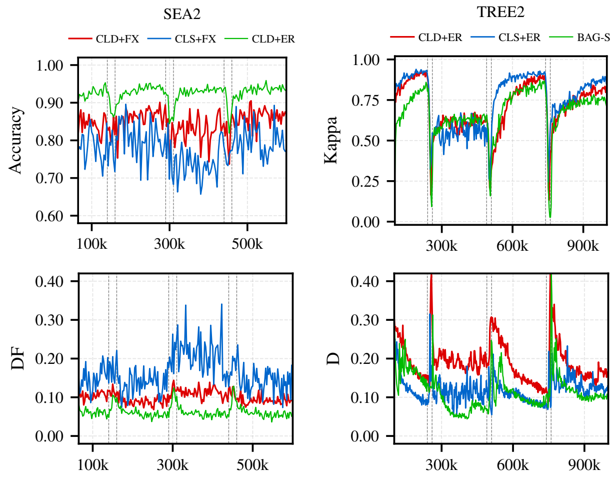

Figure 6: Performance and diversity series for SEA2 (left) and TREE2 (right), using different strategies.

Real data streams. In Tab. IV we can see results for the final evaluation conducted on real data streams. It presents average accuracies and kappa values for two different configurations of our framework, all bagging heuristics and six popular ensembles. We can see that while for bagging the dynamic λ approach was insufficient to make it competitive with most of the ensembles, our best clustering-driven configuration (CLS+ER, 0.79/0.62) was able to outperform 8 out of 9 other algorithms on average and to be at least competitive to the second best ensemble – DWM (0.78/0.59).

In fact, CLD+ER and CLS+ER were on a very similar level of performance on almost all data streams, however, the latter one turned out to be much more efficient on two very long and constantly unstable steams – Activity Raw (0.58/0.47 for CLD+ER and 0.86/0.81 for CLS+ER) and Sensor (0.45/0.44 and 0.62/0.61, respectively). It is possible that these streams are characterized by several short drifts or that the ensembles cannot saturate the concepts sufficiently, so they suffer from constant fluctuations and as a result, CLD+ER had multiple problems while reducing diversity.

The CLS+ER ensemble was the most often the best or second best solution (0.53 of cases for accuracy and 0.60 for kappa, Fig. 7), outperforming all other ensembles for both metrics on Covertype, Electricity and Weather (bold green), and being better than not our ensembles on Activity and Spam (green). Only for Poker the CLD+ER ensemble was able to be more effective than CLS+ER and, at the same time, than the other algorithms. Regarding accuracy, DWM was slightly more often the best algorithm (0.33), however, it never was the second best (usually being worse

Table IV: Results on real data stream benchmarks with the respect to accuracy and kappa metrics.

(a) Accuracy

Stream	CLD+ER	CLS+ER	BAG	BAG-D	BAG-S	OB-AH	LNSE++	DWM	AUE	ADOB	OSB
Activity	**0.7150**	0.6961	0.6224	0.5821	0.6180	0.6819	0.4890	0.6877	0.5200	0.6494	0.6697
ActivityRaw	0.5837	0.8612	0.4558	0.4057	0.5683	0.6928	0.5139	**0.8756**	0.4993	0.4053	0.6270
Airlines	0.6033	0.6003	0.6220	0.6150	0.6182	**0.6536**	0.6181	0.6560	0.6323	0.4726	0.6365
Connect4	0.7038	0.7087	0.6818	0.6581	0.6876	0.7023	0.6300	**0.7150**	0.6603	0.6080	0.7127
Covertype	0.8046	**0.8119**	0.6713	0.6508	0.6860	0.7396	0.7564	0.7960	0.7440	0.3704	0.7359
DJ30	0.9475	0.9518	0.7956	0.6521	0.8451	0.9814	0.8230	0.9813	**0.9912**	0.9819	0.9858
Electricity	0.7997	**0.8101**	0.7231	0.7109	0.7368	0.7786	0.7185	0.7858	0.7308	0.7058	0.7724
Gas	0.6495	0.6652	0.5551	0.5459	0.5503	0.5526	0.4398	**0.7162**	0.4861	0.3848	0.5489
Hepatitis	0.9083	0.9093	0.8397	0.8347	0.8410	0.9026	0.8695	0.8886	0.9156	0.9111	**0.9178**
Lymph	0.8893	0.8726	0.7944	0.7868	0.7998	0.8947	0.8448	0.8623	0.8898	0.8806	**0.8978**
Poker	**0.7315**	0.7039	0.5855	0.5585	0.5982	0.7203	0.5465	0.6640	0.6047	0.5543	0.7155
Sensor	0.4518	0.6192	0.0616	0.0538	0.2611	0.4741	0.3344	**0.6796**	0.4421	0.0303	0.3678
Spam	**0.9171**	0.9152	0.8857	0.8544	0.8867	0.7508	0.6712	0.8044	0.4853	0.8607	0.7589
Weather	0.7329	**0.7351**	0.6879	0.6760	0.6819	0.7018	0.6791	0.6912	0.7234	0.7236	0.7063
Wine	0.9270	0.9239	0.8707	0.8686	0.8709	0.9323	0.9131	0.9208	0.9320	0.9283	**0.9336**
Average	0.7577	**0.7856**	0.6568	0.6302	0.6833	0.7440	0.6565	0.7816	0.6838	0.6311	0.7324

(b) Kappa

Stream	CLD+ER	CLS+ER	BAG	BAG-D	BAG-S	OB-AH	LNSE++	DWM	AUE	ADOB	OSB
Activity	**0.6264**	0.6048	0.4978	0.3991	0.5051	0.5933	0.3069	0.5971	0.3086	0.5592	0.5819
ActivityRaw	0.4723	0.8107	0.1699	0.0896	0.3561	0.5504	0.2839	**0.8280**	0.2731	0.0329	0.4494
Airlines	0.1914	0.1838	0.2114	0.1708	0.2079	0.2684	0.2132	**0.2836**	0.2371	-0.0823	0.2327
Connect4	0.3110	**0.3601**	0.1869	0.0004	0.2367	0.2661	0.1511	0.3293	0.1969	0.2202	0.3148
Covertype	0.6829	**0.6981**	0.4708	0.4362	0.4967	0.5784	0.6082	0.6739	0.5846	0.0126	0.5676
DJ30	0.9456	0.9502	0.7884	0.6399	0.8397	0.9807	0.8168	0.9807	**0.9909**	0.9813	0.9853
Electricity	0.5848	**0.6075**	0.4333	0.4024	0.4597	0.5315	0.4103	0.5513	0.4433	0.3547	0.5174
Gas	0.5716	0.5973	0.4661	0.4509	0.4614	0.4690	0.3257	**0.6595**	0.3855	0.2504	0.4656
Hepatitis	0.7061	0.7111	0.5567	0.5411	0.5602	0.6968	0.6159	0.6594	**0.7310**	0.7057	0.7398
Lymph	0.7934	0.7632	0.6462	0.6336	0.6551	0.8032	0.7132	0.7454	0.7939	0.7762	**0.8092**
Poker	**0.5130**	0.4640	0.2359	0.1771	0.2574	0.4807	0.2035	0.3908	0.2808	0.1188	0.4785
Sensor	0.4402	0.6110	0.0430	0.0349	0.2461	0.4631	0.3203	**0.6728**	0.4304	0.0111	0.3546
Spam	**0.7860**	0.7701	0.6932	0.6397	0.7011	0.4887	0.3516	0.5117	0.1688	0.6378	0.4936
Weather	0.3319	**0.3512**	0.3023	0.2123	0.0752	0.1428	0.2118	0.1154	0.2455	0.2605	0.1743
Wine	0.8893	0.8844	0.8043	0.8011	0.8045	0.8972	0.8681	0.8799	0.8968	0.8911	**0.8992**
Average	0.5897	**0.6245**	0.4337	0.3753	0.4575	0.5474	0.4267	0.5919	0.4645	0.3820	0.5376

than both our algorithms in such cases) and more often 6th or 7th (0.33) than CLS+ER (0.2). For kappa both ensembles were equally frequently the best classifiers (0.27) and the rest relations remained very similar. Between the best and the worst ensembles, OB-AH and OSB were competitive to CLD+ER, however, still significantly worse than CLS+ER, especially for kappa values (tha gap was about 0.08). It is also worth noting that even if CLS+ER was sometimes on further ranks (3-7), it was usually on a similar level as the best ones (for example Connect4 or Poker) or on a lower, but still not dramatically worse one (for example Gas or Sensor).

We can see that dynamic diversification for the online bagging performed much worse than the clustering-driven solutions. In general, BAG-S tuned out to be the best option (0.68/0.46) and BAG-D (0.63/0.38) to be the worst one. Therefore, considering both our framework and the bagging algorithms, the relations between different configurations were very similar to those observed on synthetic streams. However, the dynamically diversified bagging algorithms were worse than most of ensembles. They were not able to improve upon static OB-AH (0.74/0.55). BAG-S was better on average only than very poorly performing LNSE++ (0.66/0.43) and ADOB (0.63/0.38), which was very often the worst algorithm (0.47 of cases). BAG-D almost always occupied the last ranks (0.80 of cases for both metrics).

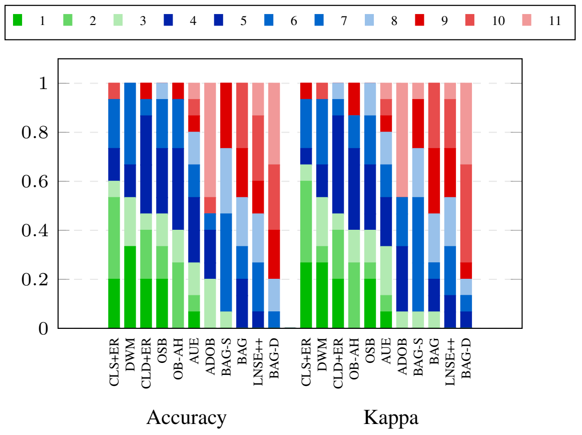

Figure 7: Ranks of ensembles for all real data streams as a fraction of all cases (15).

E. Conclusions

As a summary based on the presented results, we can conclude the following.

- **The clustering-driven diversification based on the dynamic range parameter is able to provide reactive and efficient diversification**. The presented results also confirms observations made in [9], that high diversity during changes improves adaptation and that smooth reduction of diversification after drifts helps with restoring stability.
- **The stability-oriented or at least linear control is preferable in our framework over the high-diversity one.** The former can be still sensitive to changes and recover sufficiently fast, while the latter does not provide any significant improvement during drifts and, at the same time, it impedes adaptation by overdosing diversity when stable periods emerge.
- **The intensification methods, based on learning from duplications of single instances, may significantly improve recovery from drifts.** The error-driven method, which intensifies diversification and leaves stabilization on a regular level, provides such enhancements over the basic algorithm without intensification, while still being competitive during stable periods. Applying this mechanism to the whole stream results in a disturbed diversification control and impaired adaptation.
- **The best proposed configuration of our framework is able to outperform most of the considered well-known and cited ensembles or, at least, to be competitive.** It proves that the abilities of our framework displayed and analyzed on synthetic streams, are also present in realistic scenarios.

V. Summary and future works

In this work, we proposed the online clutering-driven ensemble framework exploiting dynamic diversification. Using fully-controlled synthetic streams, we investigated different approaches to the diversity control and intensification, as well as, we combined both methods and evaluated all configurations. The best ones turned out to be very precise and effective in changing diversity and improving adaptation. Furthermore, we evaluated our framework on real data streams to show how competitive our solution is while comparing with several well-known online ensembles. In our future works, we plan to apply different drift-aware clustering methods and more sophisticated intensification approaches.

References

[1] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Comp. Int. Mag.*, vol. 10, no. 4, pp. 12–25, 2015.

[2] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[3] D. Marron, E. Ayguadé, J. R. Herrero, J. Read, and A. Bifet, "Low-latency multi-threaded ensemble learning for dynamic big data streams," in *2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017*, 2017, pp. 223–232.

[4] M. Woźniak, "Application of combined classifiers to data stream classification," in *Computer Information Systems and Industrial Management - 12th IFIP TC8 International Conference, CISIM 2013, Krakow, Poland, September 25-27, 2013. Proceedings*, 2013, pp. 13–23.

[5] Y.-R. Lee and H.-N. Kim, "A data partitioning method for increasing ensemble diversity of an eSVM-based P300 speller," *Biomedical Signal Processing and Control*, vol. 39, pp. 53 – 63, 2018.

[6] B. Krawczyk, M. Woźniak, and B. Cyganek, "Clustering-based ensembles for one-class classification," *Inf. Sci.*, vol. 264, pp. 182–195, Apr. 2014.

[7] E. K. Tang, P. N. Suganthan, and X. Yao, "An analysis of diversity measures," *Machine Learning*, vol. 65, no. 1, pp. 247–271, Oct 2006.

[8] R. Lysiak, M. Kurzynski, and T. Woloszynski, "Optimal selection of ensemble classifiers using measures of competence and diversity of base classifiers," *Neurocomputing*, vol. 126, pp. 29–35, 2014.

[9] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, May 2010.

[10] N. C. Oza, "Online bagging and boosting," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, Oct 2005, pp. 2340–2345 Vol. 3.

[11] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 619–633, April 2012.

[12] S. G. T. de Carvalho Santos, P. M. Gonçalves, Júnior, G. D. dos Santos Silva, and R. S. M. de Barros, "Speeding up recovery from concept drifts," in *Proceedings of the 2014th European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part III*, ser. ECMLPKDD'14, 2014, pp. 179–194.

[13] D. Brzezinski and J. Stefanowski, "Ensemble diversity in evolving data streams," in *Discovery Science*, 2016, pp. 229–244.

[14] E. Liberty, R. Sriharsha, and M. Sviridenko, *An Algorithm for Online K-Means Clustering*, 2016, pp. 81–89.

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