Handling Concept Drifts Using Dynamic Selection of Classifiers

Paulo R. Lisboa de Almeida*, Luiz S. Oliveira*, Alceu de Souza Britto Jr.‡ and § and Robert Sabourin¶
*Universidade Federal do Paraná, DInf, Curitiba, PR, Brazil
Email: {prlalmeida,lesoliveira}@inf.ufpr.br
‡Universidade Estadual de Ponta Grossa, Deinfo, Ponta Grossa, PR, Brazil
§Pontifícia Universidade Católica do Paraná, PPGIa, Curitiba, PR, Brazil
Email: alceu@ppgia.pucpr.br
¶École de Technologie Supérieure, Montreal, QC, Canada
Email: robert.sabourin@etsmtl.ca

***Abstract*—This work describes the Dynse framework, which uses dynamic selection of classifiers to deal with concept drift. Basically, classifiers trained on new supervised batches available over time are add to a pool, from which is selected a custom ensemble for each test instance during the classification time. The Dynse framework is highly customizable, and can be adapted to use any method for dynamic selection of classifiers given a test instance. In this work we propose a default configuration for the framework which has provided promising results in a range of problems. The experimental results have shown that the proposed framework achieved the best average rank when considering all datasets, and outperformed the state-of-the-art in three of four tested datasets.**

***Keywords*-Concept Drift; Dynamic Selection of Classifiers; Ensemble of Classifiers.**

I. Introduction

In non-stationary environments a classification problem may violate the common assumption that the data distribution and the learned concept do not change over time, characterizing the phenomenon of concept drift. Possible changes may occur in the distribution of the incoming data (virtual concept drift), or in the conditional distribution of the target concept, while the distribution of the input may stay the same (real concept drift) [1], [2].

A concept drift may be defined as abrupt, gradual probabilistic or gradual continuous, depending on the speed of the changes. Beyond that, it may be considered severe or intersected, depending on the severity of the changes between concepts. A concept is defined as recurrent when it represents an old definition that occurs again, usually motivated by seasonal changes. A detailed description about the concept drift properties can be found in [1]–[3].

A prediction method in this kind of dynamic environment is required to have a mechanism to adapt as the learned concept evolves through time. The challenge is to do it quickly, keeping the method accuracy. Some typical applications that present the concept drift phenomenon are the intrusion, spam and fraud detection, medical decision support and climate data analysis [1], [4].

A common approach employed to deal with concept drift is to keep a window containing the M latest supervised input data samples, which are used to update the classifier. A possible drawback of this strategy is the classifier will "forget" old instances that may be useful in the current concept. The family of FLORA algorithms [5] is a good example of this approach, in which an interesting contribution is the use of windows with adaptive size. Some variants of the windowed scheme are described in [4], [6], and [7], where a fading factor is applied to gradually forget old training instances or old trained models.

In a different direction some methods have been based on a trigger that tries to detect the exact moment when the concept changes. The rational behind that is to react just when the concept drift is detected, adapting the classifiers or discarding old data. Several trigger based methods were proposed in the last years, such as in [8]–[12].

Another interesting approach to deal with concept drift is based on the use of classifier ensembles. Most methods in this approach use a strategy to estimate the competence of the classifiers in the pool regarding the current concept. A common strategy is to verify the accuracy of the classifiers using the latest supervised instances received. These methods may train new classifiers from the most recent supervised data to keep the pool up to date. On the other hand, usually they remove classifiers from the pool by considering some metric, like the classifiers accuracy in the latest supervised data or their age. Some examples of ensemble based methods can be found in [13]–[16].

Despite the used approach, methods devoted to deal with concept drift most often try to keep their knowledge up to date with respect to the current concept. To this end, they must discard or at least reduce the importance of old data or trained models (possibly representing old concepts).

In this paper we propose the Dynamic Selection Based Drift Handler (Dynse) framework, which is designed to address the concept drift problem using dynamic selection of classifiers. Instead of keeping only the latest concepts (classifiers) as done in most works, we keep as much classifiers (trained with different data, collected at different times from distinct concepts) as possible in the pool, from which we can select the most appropriate ensemble for a

given test instance, considering the current concept. The rationale behind that is to profit from the inherent dynamic behavior of Dynamic Selection of Classifier based methods to react swiftly to a variety of concept drift scenarios, such as: a) severe or intersected changes; b) presence of recurrences; and c) presence of stable regions, i.e. regions where the concept does not change.

In fact, the proposed method may adapt to the possible frequent environment changes in two different moments. First, when it considers a growing pool of classifiers that is created under the environment changes, and second, when it selects from this pool the most promising classifiers dynamically, i.e, during the operational phase. Thus, the proposed system presents a dynamic behavior when the pool is generated and also when the classifiers are selected.

The designed framework can be easily adapted to different problems. Even the method used to dynamically build the ensembles, which is named *Classification Engine*, is interchangeable. In this work we assess the proposed framework employing its default configuration using two artificial, and two real world well known datasets for testing concept drift handling methods. The experimental results have shown that the proposed method is very promising when compared with related works in the literature. It was possible to observe a better average accuracy than the state-of-the-art and a faster reaction to concept drifts in most tested scenarios.

II. Dynamic Selection of Classifiers

When dealing with classification problems taking into account dynamic selection of classifiers, we are trying to find a good "custom selected" classifier or ensemble for the unlabeled instance x. For this purpose, the competence of each available classifier is estimated on a local region of the feature space during the classification phase. This local region is usually defined as the K-neighborhood of x in a validation set Q, where the class of each instance in Q is known [17], [18]. For a comprehensive review about dynamic selection of classifiers, please refer to [19].

A list of methods based on dynamic selection of classifiers may include the Overall Local Accuracy (OLA) method [17], the Local Class Accuracy (LCA) method [17], the *A Priori* and *A Posteriori* selection methods [20] and the family of K-Nearest Oracles (KNORA) algorithms [18].

In this work we have used the KNORA-ELIMINATE (KNORA-E) algorithm as the ensemble selection method (the classification engine module described in Section III). In order to make this paper self contained, we briefly describe the KNORA-E method, introducing a modification in the original algorithm to deal with noisy environments.

A. KNORA-E For Noisy Environments

Given a pool of classifiers P and the set N_x containing the k nearest neighbors of the test instance x in the validation set Q, the KNORA-E method works basically by selecting the classifiers in P that correctly classify all neighbors in N_x. The selected classifiers are then combined using the Majority Voting technique to classify x.

In our implementation, we modified the original KNORA-E algorithm in order to introduce a new slack variable l, where $0 \leq l < k$. With this modification, considering that N_x contains k neighbors, all classifiers that correctly recognize at least $k - l$ instances in N_x will be selected to be part of the ensemble. The slack variable was introduced as a simple, yet effective solution to cope with environments containing noise using the KNORA-E. The presence of noise may make it impossible for a classifier correctly classify all k neighbors for $k > 1$ (one or more of the neighbors can be noise).

III. Proposed Method

In this section we present the Dynse framework, which is a new tool for dealing with concept drifts that uses the neighborhood of the test instance defined in a validation set to dynamically select a suitable ensemble for it. The framework is designed to cope with real concept drifts of any speed and severity (in the future we intend to propose configurations of the framework to cope with virtual concept drifts). The only assumption made about the data is that some supervised samples will be available over time to train and select classifiers dynamically, and those samples will be available in a batch form (e.g. the method will receive a batch containing T supervised instances every month to adapt to the current environment).

A general overview of the proposed framework is presented in Figure 1, where each new supervised batch contains 6 samples, and only the latest supervised batch is employed as the *accuracy estimation window* W (nevertheless the size of this window can be configured for each problem to achieve optimal performance in stable or changing regions - See Figure 3).

In the framework, every supervised batch received is used to build a new classifier, which is added to a pool P. Any classification algorithm can be employed and, in case of new supervised batches do not contain enough instances to build a new classifier, V batches can be accumulated before training a new classifier. The classifier training phase is shown as a dashed line in Figure 1.

Receiving a new supervised batch also causes the update of the current accuracy estimation window W, which should contain the latest M supervised batches ($|W| = M$). This window is used by the framework to estimate the competence of the classifiers in the local region of each instance to be classified. It can be seen as the set Q described in Section II. The size of the accuracy estimation window M is directly related to the stability-plasticity dilemma [21], since a bigger value of M could generate a more accurate system when the concept is stable, at the cost of a slower recover when a concept drift occurs.

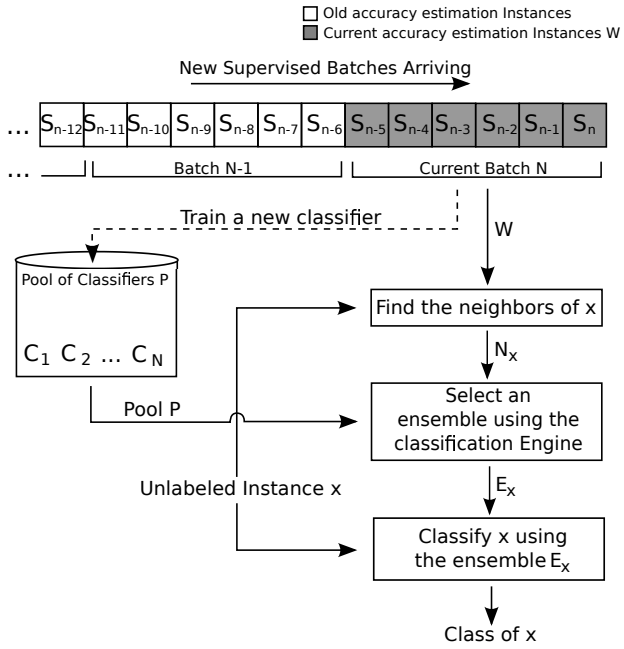


Figure 1. The Dynse Framework basic scheme for $M = 1$ in a scenario where each supervised batch contains 6 supervised samples

When a test instance x needs to be classified, the framework performs the following steps:

1) The k instances in the validation window W that represent the local region (neighborhood) of the test instance x are defined. These instances are represented by $N_x = \{x_1, x_2, ..., x_k\}$.
2) The Classification Engine CE uses the local region N_x to estimate the competence of each classifier in the pool P. This module then uses the estimated classifier competences to dynamically select a suitable ensemble E_x to classify x. Formally, the classification engine can be seen as a function $CE(N_x, P) = E_x$.
3) Finally, x is classified using the selected ensemble E_x, where the fusion rule used to combine the classifiers in E_x can be defined by the Classification Engine.

It is worth mentioning that the pool of classifiers P should be kept as big as possible, since a bigger number of classifiers trained at different moments may generate better custom selected ensembles to classify the unlabeled instances. Nevertheless, due to some constraints like processing time or available memory, it may be necessary to prune some classifiers. Under these circumstances a classical approach, like forgetting the oldest or the worst performing classifier may be implemented.

Since the performance of the dynamic selection methods has shown to be problem dependent, the classification engine of the proposed framework was planned as an independent module. Thus, the CE module can be implemented using any method for dynamic classifier selection based on the use of a local region in the feature space to evaluate the competence of the classifiers. As previously described in the current version of the classification engine we have the Knora-eliminate method proposed in [18].

A. Dynse Default Configuration

The Dynse framework is quite flexible, containing a range of components that can be interchanged or adjusted in order to achieve optimal performance in the problem being modeled. Some of these components are the base learner, the classification engine CE, the number of neighbors k representing the local region for the dynamic selection scheme, and the size of the accuracy estimation window M.

Besides the framework flexibility, we have payed special attention to the initial fine tuning of its components. The idea is to provide a default configuration that represents a good trade-off between performance in stable and concept changing regions. To this end, we have performed a set of experiments using different problems (see Section IV). The proposed default configuration is described as follows:

- A classifier is built for every V new supervised batches received. The value of V must be big enough to train a classifier (in most of our tests, $V = 1$).
- $M = 4 \times V$
- $k = 9$
- The KNORA-E method is used as the classification engine (CE), with the slack variable kept as 2 ($l = 2$).

IV. Experiments

In this section we assess the performance of the Dynse framework considering real world and artificial well known datasets. In order to make easier the comparison with other methods, we implemented the Dynse framework using the Massive Online Analysis (MOA) framework [22], which contains the implementation of the main state-of-the-art methods.

Since most of the evaluated methods have many parameters to adjust for each problem, we have used for all of them, including our method, their default configuration (available at the MOA framework). The motivation for that is to provide a fair comparison.

It is worth mentioning that, during the tests, we did not use any classifier forgetting method in our proposed framework, i.e. all classifiers are kept in the pool. With this configuration, we intend to check the ability of the Dynse framework to correctly choose the most adequate ensemble of classifiers according to the current concept.

The proposed framework was compared with seven other methods which represent different approaches to deal with concept drifts. The tested methods and the corresponding acronyms used in this work are listed below:

- **Dynse K-E92**: The Dynse Framework implemented using its default configuration.

- **DDM**: The Drift Detection Method (DDM) trigger based method proposed in [8].
- **EDDM**: The Early Drift Detection Method (EDDM) trigger based method proposed in [23].
- **HAT**: The Hoeffding Adaptive Tree proposed in [9] using the ADWIN [11] method as a trigger.
- **AUE**: The Accuracy Updated Ensemble (AUE) method proposed in [14].
- **LevBag**: The Leveraging Bagging method proposed in [10] using the ADWIN trigger to detect changes.
- **OzaASHT**: The method proposed in [24] using Adaptive-Size Hoeffding Trees.
- **OzaADWIN**: The method proposed in [24] using the ADWIN trigger.

All tested methods were configured to use Hoeffding Trees as base learners. This base learner was chosen due to its fast test/training phases, and due to its online training ability, which is required in some tested approaches. All presented results are an average of 30 executions. The following datasets were used in the experiments:

A. STAGGER Concepts

The STAGGER Concepts is a two class artificial dataset introduced in [6]. This dataset contains abrupt real concept drifts and its instances are represented by three discrete features. We defined tree different concepts as in the original work [6], plus a fourth concept, with the same boundaries of the first one, which was generated to simulate a recurring scenario.

In the experiments, we defined that the concept would change for every 10 steps, and for each step, 20 samples were given for training, and 200 were given for testing. The proposed framework were configured to create a new classifier for every new supervised batch received ($V = 1$).

B. SEA Concepts

Developed in [13], the SEA Concepts artificial dataset contains three randomly generated real features f_1, f_2 and f_3 in the range $[0, 10]$ and two possible classes $y \in \{positive, negative\}$, where the boundary that separates the classes is given by $f_1 + f_2 \leq \theta$. Concept drifts are introduced by varying the θ threshold, where the values of θ for each of the four possible concepts are 8, 9, 7 and 9.5, respectively. Class noise is inserted by swapping the classes of 10% of the instances.

The testing procedure employed was the same as in [16], where for each time step, a supervised batch containing 250 samples is given for training, and another batch containing 250 samples from the same concept is generated for testing. The concept is changed for each 50 steps, thus generating a test with 200 steps. We must point out that in [16] only the training instances contain noise, whilst in out tests both training and testing instances have noise. We have considered $V = 1$.

C. Forest Covertype Dataset

Available at the UCI Repository [25], this dataset defines the classification task as identifying the forest cover type for 30×30 meters cells. Each sample is described by 10 numerical and 44 categorical attributes. The class attribute, which belongs to the range $[1, 7]$, identifies the forest cover type. The dataset is composed of 581,012 samples and was used as a benchmark in [4], [10], [24].

To evaluate the methods using this dataset we employed an interleaved batches approach where, following the instances ordering present in the original dataset, at each step a batch containing 2.500 samples is given for training, and the subsequent batch of 2.500 samples is given for testing. In the next step, the previous testing batch is used for training, and the next batch containing 2.500 samples is given for testing. The procedure is repeated until all instances in the dataset are used. Again, in these tests, we considered that each new supervised batch would be used to generate a new classifier, thus $V = 1$ and $M = 4$.

D. Nebraska Weather Dataset

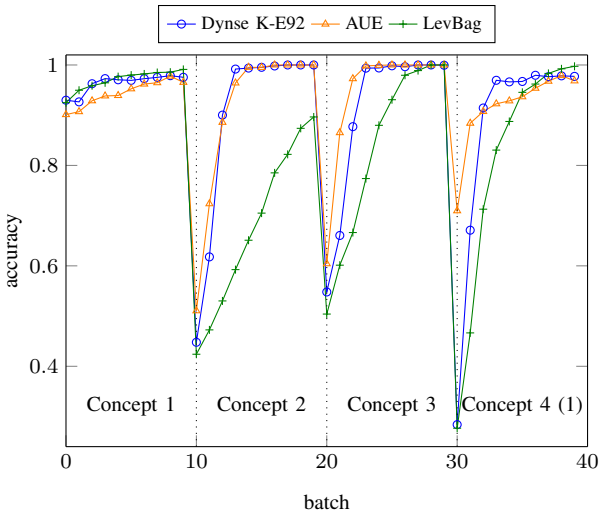
This dataset refers to the weather data collected by the U.S. National Oceanic and Atmospheric Administration, in the Offutt Air Force Base in Bellevue, Nebraska. It has an extensive range of 50 years, containing 18,159 samples, and the presence of diverse weather patterns. In this dataset the class labels are binary, indicating the presence or absence of rain in each sample [16].

We employed the same configuration as in [16], where only the eight features with a missing feature rate less or equal than 15% were used. The remaining missing values are replaced by the mean of the features in the preceding and following samples. Also as in [16], a interleaved batches approach was used in the tests, where each batch contain 30 samples. Due to the small number of supervised samples given at each step, we defined that for the methods that builds new classifiers according to the new supervised batches, a new classifier should be build using 3 accumulated batches ($V = 3$).

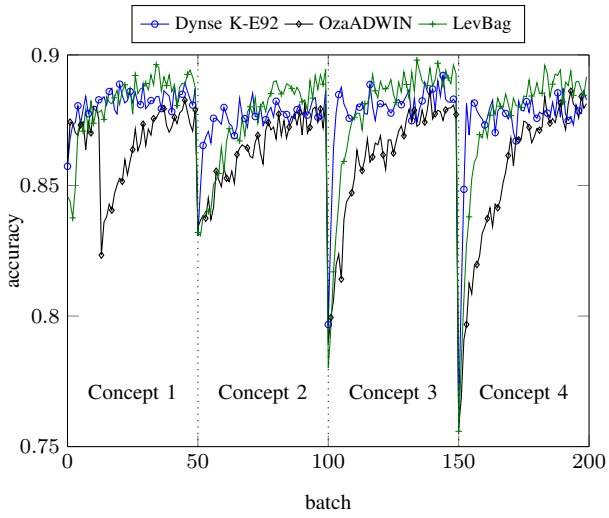
E. Results and Discussion

Figure 2 contains the average accuracy achieved by the three best performing methods for each testing batch of each tested dataset. A summary containing the average accuracies of the methods when considering all batches of each dataset, their ranks in each dataset, and their average rank can be seen in Table I. The best results are shown in bold.

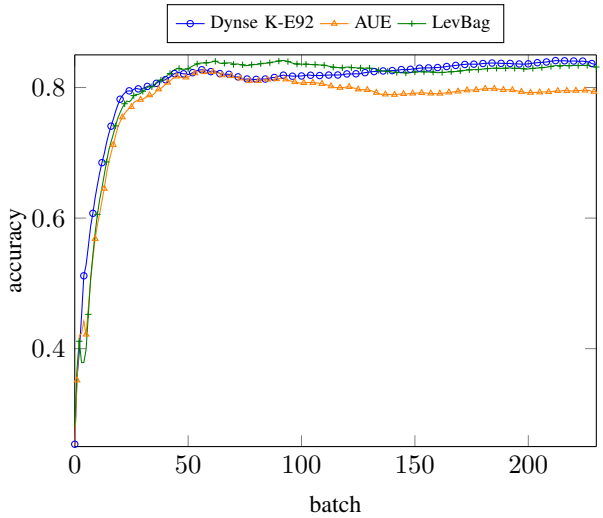
The results show that the Dynse framework using its default configuration was the best performing method in the SEA Concepts, Forest Covertype and Nebraska Weather datasets, and was the second best performing method in the STAGGER Concepts dataset. When considering the artificial datasets (Figures 2a and 2b), it is possible to check that the proposed framework showed a fast recover when the concept



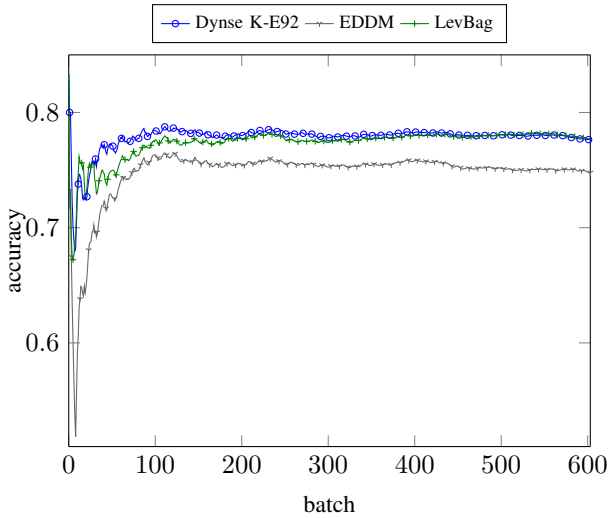
(a) STAGGER Concepts Result



(b) SEA Concepts Result



(c) Forest Covertype Result



(d) Nebraska Result

Figure 2. Average accuracy achieved in each testing batch for the three best performing methods.

changed, often adapting to the new concept faster than the methods in the state-of-the-art. The average rank in Table I shows that, on average, the proposed framework was the best performing method when considering all the tests.

It is worth reminding that all results in Figure 2 and in Table I were achieved using the default configuration of the Dynse framework. We believe that by a fine tuning (using cross-validation on the available supervised data, for instance) of the Dynse parameters better results may be achieved. For instance, the Dynse framework can be tuned according to the environment properties by selecting the most suitable CE and the size (M) of the accuracy estimation window.

To illustrate this, consider the plot in Figure 3, which contains a test using the default configuration of the proposed method, and a configuration that uses a accuracy estimation window size equals to 1 (the remainder of the configuration equals to the default one) in the SEA Concepts dataset. As one can observe, the simple tuning on the accuracy window size may generate better results in concept changing areas ($M = 1$) or in stable areas ($M = 4$).

We also paid special attention in the Leveraging Bagging method, since it achieved good results in most tests. The result in the SEA Concepts and in the Forest Covertype datasets (Figures 2b and 2c) indicates that this method may lead to a good performance, specially in stable regions. Mostly probably this method is getting a benefit from a more diverse pool of classifiers generated by the bagging method implemented by its authors [10].

Finally, we should state that these results are just a prove of concept of our proposed approach, since some methods in the state-of-the-art, like the Leveraging Bagging method,

Table I
SUMMARY OF THE EXPERIMENTAL RESULTS BASED ON 30 REPLICATIONS. METHOD AVERAGE ACCURACY, STANDARD DEVIATION AND RANK POSITION FOR EACH DATASET, PLUS THE GENERAL AVERAGE RANK OF EACH METHOD.

Method	**STAGGER Concepts**		**SEA Concepts**		**Forest Covertype**		**Nebraska Weather**		**Average**
	Average(%)	Rank	Average(%)	Rank	Average(%)	Rank	Average(%)	Rank	**Rank**
Dynse K-E92	90.8 ± 16.9	2	**87.7 ± 1.3**[2]	**1**	**80.5 ± 7.6**	**1**	**77.8 ± 1.2**	**1**	**1.25**
DDM	70.0 ± 8.1	6	86.1 ± 2.5	5	65.7 ± 9.9	7	74.7 ± 1.3	4	5.5
EDDM	72.5 ± 9.7	5	84.3 ± 4.4	7	48.3 ± 6,7	8	74.7 ± 2.7	3	5.75
HAT	67.0 ± 19.8	7	86.1 ± 2.4	4	74.7 ± 7.6	6	72.3 ± 2.2	8	6.25
AUE	**92.8 ± 10.9**	**1**	84.2 ± 2.7	8	77.8 ± 8.1	3	73.6 ± 0.7	5	4.25
LevBag	82.1 ± 20.0	3	87.7 ± 2.2	2	80.4 ± 8.8	2	77.3 ± 1.4	2	2.25
OzaASHT	67.0 ± 19.5	8	85.3 ± 2.5	6	76.9 ± 7.8	4	73.5 ± 1.6	6	6.0
OzaADWIN	77.6 ± 19.9	4	86.1 ± 2.2	3	76.6 ± 8.5	5	73.4 ± 1.5	7	4.75
Learn++.NSE [16] [1]	-	-	96.6 ± 0.2[3]	-	-	-	75.9 ± 0.7[4]	-	-

[1] The results refers to the experiments in [16], where a Naive Bayes were used as the base classifier.
[2] There is a tie with the LevBag method when considering the accuracy. Nevertheless the LevBag has a higher standard deviation.
[3] The authors generated the testing batches without noise.
[4] When using a Naive Bayes as the base classifier, as in [16], we achieved a accuracy of 77.9%.

achieved similar results to our framework, and we do not consider the executed tests are enough to claim that the default configuration of the Dynse should be considered a better alternative in every concept drift scenario. Instead, we consider these positive results as a good indication that our framework did benefit from the dynamic selection of classifiers approach to deal with concept drifts, thus giving some important guidelines for our future work described in Section V.

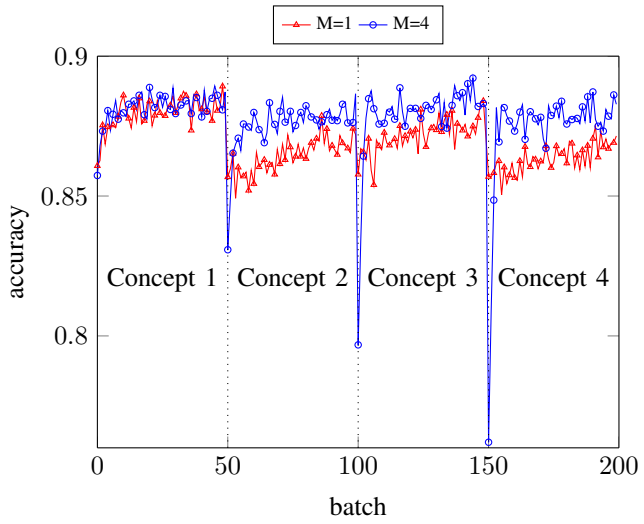


Figure 3. The Dynse framework using its default configuration ($M = 4 \times V = 4$) and using $M = V = 1$ in the SEA Concepts dataset.

V. CONCLUSION

In this paper we presented the Dynse framework, which is a new local accuracy based dynamic classifier selection method designed to deal with the concept drift phenomenon. One of the main features of our framework is its flexibility to handle a variety of problems, since the only assumption made by the method is that some supervised batches will be available over time for training new classifiers and for estimating the classifiers competence.

Our tests showed that the Dynse framework is capable of keeping a good performance in a variety of datasets, outperforming the average accuracy of the state-of-the-art methods in three of the four tested datasets, and showing the best average rank when considering all tested datasets. A particular test in the SEA Concepts dataset showed that by adjusting just the accuracy estimation window size parameter of our framework it is possible to achieve better results in stable or concept changing areas.

In this work we also proposed a modification in the original KNORA-E algorithm to introduce a slack variable l to handle noisy environments. This modification provided good results in our tests when applying the KNORA-E as a classification engine for the Dynse framework, under the presence of both noisy data and concept drifts (SEA Concepts tests).

As future work we intend to test the proposed framework using different classification engines in order to check the impact of different dynamic selection of classifiers methods to deal with concept drift scenarios. Tests with different parameter configurations and strategies for generating classifiers for the pool will also contribute with a better understanding of the dynamic selection of classifiers under non-static environments.

The results achieved by the Leveraging Bagging method indicates that we could better benefit from the pool of classifiers by increasing its diversity through the use of some method like the Bagging or Boosting. Or future works will include this approach, since it could be specially beneficial in the beginning of the test or when the concept change, when the number of classifiers in the pool from the current concept is relatively small.

We also plan to evaluate different classifier pruning strategies in order to keep the pool from increasing its number of classifiers indefinitely. Finally, it is necessary to test the Dynse framework with datasets containing different concept drift scenarios, including gradual and virtual concept drifts.

REFERENCES

[1] T. Hoens, R. Polikar, and N. Chawla, "Learning from streaming data with concept drift and imbalance: an overview," *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 89–101, 2012.

[2] J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 1 – 37, Mar. 2014.

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

[4] B. Krawczyk and M. Woniak, "One-class classifiers with incremental learning and forgetting for data streams with concept drift," *Soft Computing*, pp. 1–14, 2014.

[5] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learn.*, vol. 23, no. 1, pp. 69–101, 1996.

[6] J. Schlimmer and R. Granger, Jr., "Incremental learning from noisy data," *Machine Learning*, vol. 1, no. 3, pp. 317–354, 1986.

[7] D. Martínez-Rego, B. Prez-Snchez, O. Fontenla-Romero, and A. Alonso-Betanzos, "A robust incremental learning method for non-stationary environments," *Neurocomputing*, vol. 74, no. 11, pp. 1800 – 1808, 2011.

[8] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Advances in Artificial Intelligence SBIA 2004*, ser. Lecture Notes in Computer Science, A. Bazzan and S. Labidi, Eds. Springer Berlin Heidelberg, 2004, vol. 3171, pp. 286–295.

[9] A. Bifet and R. Gavaldà, "Adaptive learning from evolving data streams," in *Advances in Intelligent Data Analysis VIII*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2009, vol. 5772, pp. 249–260.

[10] A. Bifet, G. Holmes, and B. Pfahringer, "Leveraging bagging for evolving data streams," in *Machine Learning and Knowledge Discovery in Databases*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2010, vol. 6321, pp. 135–150.

[11] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *In SIAM International Conference on Data Mining*, 2007.

[12] C. Salperwyck, M. Boulle, and V. Lemaire, "Concept drift detection using supervised bivariate grids," in *Neural Networks (IJCNN), 2015 International Joint Conference on*, July 2015, pp. 1–9.

[13] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM Press, 2001, pp. 377–382.

[14] D. Brzeziński and J. Stefanowski, "Accuracy updated ensemble for data streams with concept drift," in *Hybrid Artificial Intelligent Systems*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6679, pp. 155–163.

[15] H. Wang, W. Fan, P. S. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proceedings of the Ninth ACM SIGKDD International Conf. on nowledge Discovery and Data Mining*, ser. KDD '03. New York, NY, USA: ACM, 2003, pp. 226–235.

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

[17] K. Woods, J. Kegelmeyer, W.P., and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *Pattern Analysis and Machine Intel., IEEE Trans. on*, vol. 19, no. 4, pp. 405–410, Apr. 1997.

[18] A. Ko H. R., R. Sabourin, and A. S. Britto, Jr., "From dynamic classifier selection to dynamic ensemble selection," *Pattern Recognition*, vol. 41, no. 5, pp. 1718 – 1731, 2008.

[19] A. S. Britto, Jr., R. Sabourin, and L. E. Oliveira, "Dynamic selection of classifiers - A comprehensive review," *Pattern Recognition*, vol. 47, no. 11, pp. 3665 – 3680, 2014.

[20] L. Didaci, G. Giacinto, F. Roli, and G. L. Marcialis, "A study on the performances of dynamic classifier selection based on local accuracy estimation," *Pattern Recognition*, vol. 38, no. 11, pp. 2188 – 2191, 2005.

[21] S. Grossberg, "Nonlinear neural networks: Principles, mechanisms, and architectures," *Neural Networks*, vol. 1, no. 1, pp. 17 – 61, 1988.

[22] A. Bifet, G. Holmes, B. Pfahringer, P. Kranen, H. Kremer, T. Jansen, and T. Seidl, "Moa: Massive online analysis, a framework for stream classification and clustering," in *Journal of Machine Learning Research (JMLR) Workshop and Conference Proceedings, Volume 11*. Journal of Machine Learning Research, 2010, pp. 44–50.

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

[24] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà, "New ensemble methods for evolving data streams," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 139–148.

[25] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml