Droplet Ensemble Learning on Drifting Data Streams

Pierre-Xavier Loeffel, Albert Bifet, Christophe Marsala, Marcin Detyniecki

Droplet Ensemble Learning on Drifting Data Streams

Pierre-Xavier Loeffel[1,2], Albert Bifet[4], Christophe Marsala[1,2], and Marcin Detyniecki[1,2,3]

[1] Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6, F-75005, Paris France,
[2] CNRS, UMR 7606, LIP6, F-75005, Paris, France,
[3] Polish Academy of Sciences, IBS PAN, Warsaw, Poland
[4] LTCI, Télécom ParisTech, Université Paris-Saclay, 75013, Paris, France
{pierre-xavier.loeffel, christophe.marsala, marcin.detyniecki}@lip6.fr
albert.bifet@telecom-paristech.fr

Abstract. Ensemble learning methods for evolving data streams are extremely powerful learning methods since they combine the predictions of a set of classifiers, to improve the performance of the best single classifier inside the ensemble. In this paper we introduce the Droplet Ensemble Algorithm (DEA), a new method for learning on data streams subject to concept drifts which combines ensemble and instance based learning. Contrarily to state of the art ensemble methods which select the base learners according to their performances on recent observations, DEA dynamically selects the subset of base learners which is the best suited for the region of the feature space where the latest observation was received. Experiments on 25 datasets (most of which being commonly used as benchmark in the literature) reproducing different type of drifts show that this new method achieves excellent results on accuracy and ranking against SAM KNN [1], all of its base learners and a majority vote algorithm using the same base learners.

Keywords: Concept Drift, Ensemble Learning, Online-Learning, Supervised Learning, Data Streams

1 Introduction

The explosion of data generated in real-time from streams has brought to the limelight the learning algorithms able to handle them. Sensors, stock prices on the financial markets or health monitoring are a few example of the numerous cases in real life where data streams are generated. It is therefore important to devise learning algorithms that can handle this type of data.

Unfortunately, these data streams are often non-stationary and their characteristics can change over time. For instance, the trend and volatility of the stock prices can suddenly change as a consequence of an unexpected economic event. This phenomenon, referred as *concept drift* (when the underlying distribution which generates the observations on which the algorithm is trying to learn

changes over time), raises the need to use adaptive algorithms to handle data streams.

In this paper we propose a novel ensemble method which aims at obtaining good performances regardless of the dataset and type of drift encountered. One of the main characteristic of this method is that, it determines the regions of expertise of its base learners (BL) in the feature space and selects the subset of BL which is the best suited to predict on the latest observation. This new method outperforms SAM-KNN [1], a new classifier algorithm for data streams that won the Best Paper award at ICDM 2016.

The main contributions of the paper are the following:

- a new streaming classifier for evolving data streams, which weights its base learners according to their local expertise in the feature space.
- an extensive evaluation over a wide range of datasets and type of drifts.
- a discussion on how the new method, DEA over-performs the best state of the art algorithms.

The paper is organized as follows: Section 2 lays down the framework of our problem and goes through the related works. Section 3 details the proposed algorithm while Section 4 presents the datasets used as well as the experimental protocol. Section 5 presents and discuss the results of the experiments and finally Section 6 concludes.

2 Framework and related work

In this section we present the framework of our problem and we discuss related works on learning algorithms handling concept drift.

2.1 Framework

The problem being addressed here is supervised classification on a stream of data subject to concept drifts. Formally, a stream endlessly emits observations $\{x_1, x_2, ...\}$ (where $x_i = \left\{x_i^1, ..., x_i^k\right\} \in X = \mathbb{R}^k$, k designates the dimension of the feature space and i designates the time step at which the observation was received) which are unlabeled at first but for which a label $y_i \in Y = \{1, ..., c\}$ is being received a constant amount of time $u \in \mathbb{R}^{+*}$ after x_i. We will work in the framework where the label of x_i is always received before reception of x_{i+1}. The goal is to create an on-line classifier $f : X \to Y$ which can predict, as accurately as possible, the class y_i associated to x_i.

An on-line classifier is a classifier which can operate when data are received in sequence (as opposed to a batch classifier which needs a full dataset from scratch to operate) and can evolve over time (i.e. its learned model is constantly updated with the latest observations). Formally, the operating process of an on-line classifier is described thereafter:

– When an observation x_t is received at time t, it outputs a prediction $\hat{y}_t$. The true class y_t is then released and, after computation of the prediction error according to the 0-1 loss function: $\mathscr{L}(y, \hat{y}) = \mathbb{I}_{\{y \neq \hat{y}\}}$ (where $\mathbb{I}$ is the indicator function), the classifier is updated with the latest observation: $f_t = Update(f_{t-1}, \{x_t, y_y\})$. Our goal then, is to minimize the average error $\left(\frac{1}{n} \sum_{i=1}^{n} \mathscr{L}(y_i, \hat{y}_i)\right)$ over the n observations received so far.

In the considered framework, the hidden joint distribution $P(X, Y)$ (called *Concept*) which generates the couples (x_i, y_i) at each time step, is also allowed to unexpectedly change over time: a phenomenon referred as *Concept Drift*. Formally [1], concept drift occurs at time t if $P_{t-1}(X, Y) \neq P_t(X, Y)$. According to Bayes rule: $P(X, Y) = P(Y/X) P(X)$. Thus, a drift of concept can result either in a change of the posterior probability of the classes $P(Y/X)$ (called *real drift*) either in a change of the distribution of the features $P(X)$ (called *virtual drift*) either in both.

The types of drifts can be further categorized according to the speed at which they occur. We say that a drift is *abrupt* when the drift last for one observation ($P_{t-1}(X, Y) \neq P_t(X, Y)$ and the concept is stable before $t-1$ and after t) or conversely that it is *incremental* when the drift last more than one observation ($P_{t-k}(X, Y) \neq ... \neq P_{t-1}(X, Y) \neq P_t(X, Y)$ and the concept is stable before $t-k$ and after t). *Reoccurring drift*s, happen when a previously learned concept reappears after some time ($\exists k \in \mathbb{N} / P_{t-k}(X, Y) = P_t(X, Y)$).

2.2 Related work

Several methods have been proposed in order to deal with the issue of drifting concepts on data streams, the majority of which being ensemble methods.

ADACC was introduced in [11]. It maintains a set of BL which are weighted every τ time steps according to their number of wrong predictions. It then randomly selects one BL from the worst half of the ensemble and replaces it by a new one which is protected from deletion for a few time steps. The final prediction is given by the current best performer. The algorithm also includes a mechanism to remember past concepts.

Dynamic Weighted Majority (DWM) is an ensemble method introduced in [6]. Each of its BL has a weight which is reduced in case of a wrong prediction. When a BL's weight drops bellow a given threshold, it is deleted from the ensemble. If all the BL output a wrong prediction on an instance, a new classifier is added to the ensemble.

ADWIN Bagging (Bag Ad) was introduced in [9] and improves the On-line Bagging algorithm proposed by Oza and Rusell [10] by adding the ADWIN algorithm as a change detector. When a change is detected, the worst performing BL is replaced by a new one.

Similarly, **ADWIN Boosting** (Boost Ad) improves the on-line Boosting algorithm of Oza and Russell [10] by adding ADWIN to detect changes.

Leveraging Bagging (Lev Bag) was introduced in [7] and further improves the ADWIN Bagging algorithm by increasing re-sampling (using a value λ larger than 1 to compute the Poisson distribution) and by adding randomization at the output of the ensemble by using output codes.

Hoeffding Adaptive Tree (Hoeff Tree) was introduced in [12] and uses ADWIN to monitor the performance of the branches on the tree. When the accuracy of a branch decreases, it is replaced with a more accurate one.

AccuracyUpdatedEnsemble (AUE) described in [4] maintains a weighted ensemble of BL and uses a weighted voting rule for its final prediction. It creates a new BL after each chunk of data which replaces the weakest performing one. The weights of each BL are computed according to their individual performances on the latest data chunk.

Finally, **SAM KNN** [1], best paper award at ICDM 2016, is a new improvement method of the KNN algorithm. It maintains the past observations into 2 types of memories (short and long term memory). The task of the short term memory is to remain up to date according to the current concept whereas the long term memory is in charge of remembering the past concepts. When a concept change is detected, the observations from the short term memory are transfered to the long term memory.

3 The Droplets Ensemble Algorithm

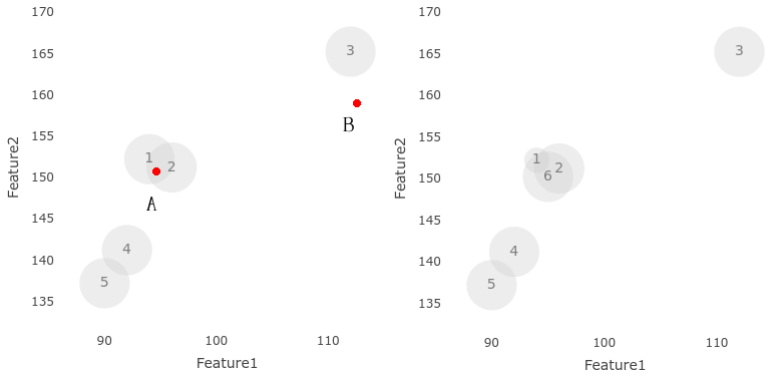

Fig. 1. Example of map learned in 2 dimensions. Left: before update of the model with the 6^{th} observation (received at point A). Right: after update of the model with the 6^{th} observation.

Our main goal in designing our new ensemble algorithm dealing with a data stream subject to concepts drift, is to take into account the local expertise of each of its BL on the region of the feature space where the latest observation

was received. This means that it gives more weight to the predictions of the BL which demonstrated an ability to predict accurately in this region.

We propose DEA (Droplets Ensemble Algorithm), an ensemble learning algorithm which dynamically maintains an ensemble of n BL $\left(F = \left\{f^1, ..., f^n\right\}\right)$ along with an ensemble of p Droplets $\left(Map = \left\{D^1, ..., D^p\right\}\right)$ up to date with respect to the current concept.

The BL can be any learning algorithms, as long as they are able to classify on a data stream subject to concept drifts.

A Droplet is an object which can be represented as a k-dimensional hypersphere (with k the dimension of the feature space). Each Droplet D^t is associated with an observation x_t and holds a pointer to a BL: f^i $(i \in \{1, ..., n\})$. The values taken by x_t correspond to the coordinates of the center of the Droplet in the feature space whereas f^i corresponds to the BL which managed to achieve the lowest prediction error on a region of the feature space defined around x_t.

Figure 1. shows an example of Map learned where the numbers represent the time step at which each Droplet has been received.

We now go through the algorithm in details.

3.1 Model Prediction

Algorithm 1 Model Prediction

Inputs: $F = \left\{f^1, ..., f^n\right\}$: Ensemble of base learners,
$Map = \left\{D^1, ..., D^p\right\}$: Ensemble of existing Droplets,
x_t: Latest unlabeled observation,
x_{const}: Normalization constants
Output: $\hat{y_t}$: Estimated class for x_t

$x_t^{norm} \leftarrow Normalize\left(x_t, x_{const}\right)$
$OD_t \leftarrow Get\ overlapped\ Droplets\left(Map, x_t^{norm}\right)$
If $(OD_t \neq \emptyset)$
  Foreach $D^h \in OD_t$ $(h \in \{a, ..., u\})$
    $\hat{y_t^h} \leftarrow Predict\left(D^h, x_t\right)$
  End Foreach
  $\hat{y_t} \leftarrow Majority\ Vote\left(\hat{y_t^a}, ..., \hat{y_t^u}\right)$
Else
  $D^{nn} \leftarrow Get\ Nearest\ Droplet\left(Map, x_t^{norm}\right)$
  $\hat{y_t} \leftarrow Predict\left(D^{nn}, x_t\right)$
End If

At time t, upon reception of a new unlabeled observation x_t the first step is to normalize the values of x_t according to a vector of normalization constants x_{const}

found on the initialization step[5]. Then OD_t, the set of Droplets which contains the normalized coordinates of the latest observation is computed. If $OD_t \neq \emptyset$, the predicted value for this observation is given by a simple majority vote of the BL associated with the overlapped Droplets in OD_t. On the other hand, if $OD_t = \emptyset$, the learner associated with the nearest Droplet D^{nn} is used for prediction. For instance, in the left plot of Fig. 1., if an observation is received at the position of point A, the BL associated with D^1 and D^2 will be used for prediction whereas if an observation is received at the position of point B, only the BL associated with D^3 will be used for prediction.

The prediction process is summarized in Algorithm 1.

3.2 Model Update

Once the true label y_t associated with the latest observation x_t is released, each BL f^i (with $i \in \{1, ..., n\}$) predicts on the latest observation and the vector of the prediction errors $e_{t+1} = \left\{e^1_{t+1}, ..., e^n_{t+1}\right\}$ (with $e^i_{t+1} \in \{0, 1\}$) is set aside. The BL are then updated with $\{x_t, y_t\}$.

The next step is to search for the BL which will be associated to the new Droplet D^t. This is done by summing the prediction errors achieved by each BL on the N nearest Droplets , where N is a parameter defined by the user. If an unique BL minimizes this sum, it is associated to the new Droplet, otherwise (if at least 2 BL minimizes the sum of prediction error) the search space is expanded in turns to the $N + 1, N + 2, N + 3, ...$ nearest Droplets until a single best performer is found.

The new Droplet D^t is then added to the feature space at the coordinates of x^{norm}_t. This Droplet is given a default radius $R_{default}$ (where $R_{default}$ is a parameter defined by the user), stores the vector of prediction errors e_{t+1} and creates a pointer to the best BL f^k found on the previous step.

The algorithm then goes through the set of overlapped Droplets OD_t and if it is not empty, it decreases the influence of the Droplets in OD_t which have outputted a wrong prediction on x_t. This is done by shrinking their radius which will make them less likely to predict on a future observation received in this region of the feature space. Formally, for each Droplet u in OD_t :

1. Compute the overlap between D^u and the latest Droplet:
   $Overlapp_u = R_{default} + R_u - \|x^{norm}_u - x^{norm}_t\|$ (where $\|.\|$ denotes the Euclidean distance)
2. Update the radius of D^u: $R_{u,t+1} = R_{u,t} - \frac{Overlapp_u}{2}$.
3. Delete D^u if $R_{u,t+1} \leq 0$.

For instance the right plot of Fig. 1. shows the updated model after reception of an observation at the position of point A and where the BL associated with D^1

[5] This is simply done by computing the average μ^i as well as the standard deviation σ^i of each feature on the initialization set and by transforming the i^{th} feature of x_t into $\frac{x^i_t - \mu^i}{\sigma^i}$

outputted a wrong prediction on the 6^{th} observation whereas the BL associated with D^2 predicted correctly.

Finally, a memory management module is ran at each time step to ensure that p, the user defined parameter for the maximum number of Droplets allowed in memory is not exceeded. If the memory is full, the algorithm uses 3 different criteria to select the Droplet which will be removed:

1. Remove the Droplet with the smallest radius.
2. If all the Droplets have the same radius, remove the Droplet which has outputted the highest number of wrong prediction.
3. If criteria 1. and 2. failed, remove the oldest Droplet.

Algorithm 2 summarizes the model update process.

Algorithm 2 Model Update

Inputs: $R_{default}$: Default radius of a Droplet,
$F = \left\{f^1, ..., f^n\right\}$: Ensemble of base learners,
$Map = \left\{D^1, ..., D^p\right\}$: Ensemble of existing Droplets,
x_t: Latest unlabeled observation,
y_t: True label latest observation,
p: Maximum number of Droplets allowed in memory,
OD_t: Set of overlapped Droplets at time t
Output: Updated DEA

Foreach f^i in F
    $e^i_{t+1} \leftarrow Get\ Prediction\ Error\left(f^i_t, \{x_t, y_t\}\right)$
    $f^i_{t+1} \leftarrow Update\ Base\ Learner\left(f^i_t, \{x_t, y_t\}\right)$
End foreach
$f^k \leftarrow Search\ best\ base\ learner\left(Map, \{x_t, y_t\}\right)\ ,\ k \in \{1, ..., n\}$
$D^t \leftarrow Create\ Droplet\left(R_{default}, x_t^{norm}, f^k, e_{t+1}, sum\ errors = 0\right)$
$Map \leftarrow Add\ Droplet\left(Map, D^t\right)$
Foreach $D^u \in OD_t$
    $R_{u,t+1} \leftarrow Update\ Radius\left(R_{u,t}\right)$
    If $R_{u,t+1} \leq 0$
        $Map \leftarrow Remove\ Droplet\left(Map, D^u\right)$
    End if
End foreach
If $\left(Card\left(Map\right) \geq p\right)$
    $Map \leftarrow Memory\ Management\left(Map\right)$
End if

3.3 Running time and space requirements

Running time: Provided that each of the base learner runs in constant time at each time step, the temporal complexity of both the prediction and update steps of DEA is $\mathcal{O}\left(i.p\right)$ whith i is the number of observations generated by the stream so far and p the maximum number of Droplets allowed in memory.

Space requirements: As previously explained, the maximum number p of Droplets saved into computer memory is constrained and so is the number n of base learners. This means that, as long as each of the n base learner constrains its memory consumption at each time step, the space complexity of DEA will be $\mathcal{O}(n+p)$ which is independant of the number of observations generated by the stream so far.

4 Experimental Framework

In this section, we describe the datasets on which the experiments have been conducted, their characteristics as well as the experimental protocol used.

4.1 Datasets

A total of 25 artificial and real world datasets have been used. These datasets have been chosen for the diversity of their characteristics, which are summarized thereafter:

Dataset	Features	Classes	Observations	# Drifts	Type of Drift
Agrawal	9	2	100 000	0	N/A
Random Tree	10	6	100 000	15	Abrupt, Reoc.
Waveform	21	3	100 000	Continuous	Real, Local
LED	7	10	100 000	Continuous	Real, Local
KDD Cup	41	2	494 000	N/A	N/A
Rotating Check	2	2	419 600	Continuous	Real, Gradual
SPAM	500	2	9324	N/A	N/A
Usenet	658	2	5 931	N/A	N/A
Airlines	7	2	153 200	N/A	N/A
Multi Dataset	3	2	200 000	3	Abrupt, Real
Multi Dataset NO	3	2	200 000	Continuous	Abrupt, Real, Virt.
Weather	8	2	18 100	N/A	N/A
SEA	3	2	50 000	3	Real, Abrupt
Chess	2	8	200 000	0	N/A
Transient Chess	2	8	200 000	Continuous	Virt., Reoc.
Mixed Drift	2	15	600 000	Continuous	Incr., Abrupt, Virt.
Moving Square	2	4	200 000	Continuous	Real, Gradual
Interchanging RBF	2	15	200 000	9	Real, Abrupt
Moving RBF	10	5	200 000	Continuous	Real, Gradual
Cover Type	54	7	581 000	N/A	N/A
Electricity	8	2	45 300	N/A	N/A
Outdoor Stream	21	40	4 000	N/A	N/A
Poker Hand	10	10	829 200	N/A	N/A
Rialto	27	10	82 200	N/A	N/A
Rotating Hyp.	10	2	200 000	Continuous	Real, Gradual

Table 1. Datasets used and their characteristics

Most of these datasets have frequently been used in the literature dedicated to streams subjects to concept drifts. Also, please note that in this table, an "N/A" value doesn't mean that there is no concept drift. It means that, because the dataset comes from the real world, it is impossible to know for sure the number of drifts it includes as well as their type.

The first 4 datasets from Agrawal to LED have been generated using the built-in generators of MOA[6]. A precise description of theses datasets can be found in the following papers [13,14,15]. The KDD Cup 10 percent dataset was introduced and described in [2]. Rotating Check board was created in [5] and the version CB (Constant) dataset was used (constant drift rate). SPAM was introduced in [3] and Usenet was inspired by [8]. Airlines was introduced in the 2009 Data Expo competition. The dataset has been shrinked to the first 153 000 observations.

Multidataset is a new synthetic dataset created for this paper. Every 50 000 observations, the concept drifts to a completely new dataset, starting with Rotating checkboard, then Random RBF, then Rotating Hyperplane and finally SEA. In the basic version, the successive concepts overlap each other whereas in the No Overlap (NO) version the datasets are shifted and the data are randomly generated on each dataset.

Finally, all the datasets listed after Weather have been retrieved from the repository[7] given in the paper of Losing et al. [1].

All the datasets used as well as the code of the DEA algorithm and the results of the experiments are available at the following link[8].

4.2 Experimental Setting

MOA have been used to conduct the experiments and provide the implementation of the classifiers. DEA was also implemented in MOA. The code for SAM KNN was directly retrieved from the link provided in their paper[1][9].

All the parameters of all the classifiers were set to default values (except for the training period which was set to 100 observations for all the learners and for the number of observations allowed in memory which was set to 400 for DEA and SAM KNN) and for all the datasets. In the case of the Droplets algorithm, the default radius was set to 0.1 and the minimum number of neighbors considered was set to 20 for all the experiments. We used all the algorithms described in this paper as BL for DEA (they were chosen because of their availability on MOA) with the exception of SAM KNN and Majority Vote. The simple majority

[6] http://moa.cms.waikato.ac.nz/
[7] https://github.com/vlosing/driftDatasets
[8] https://mab.to/o5iNvZdhH
[9] https://github.com/vlosing/driftDatasets

vote algorithm (which uses the same BL as DEA) was used as a base-line for performance comparison.

Leaving all the parameters to default values for all the datasets is required because there is no assumptions regarding the structure of the data or the type of drifts the classifiers will have to deal with. Therefore, it wouldn't be relevant to optimize parameters that would be suitable for a particular concept, at a particular time and for a particular dataset.

The goals of the experiments were to compare the performance of DEA against one of the currently best adaptive algorithm (SAM KNN), assess how DEA was faring against another ensemble algorithm which is given the same BL (Majority Vote) and assess whether DEA was able to outperform each of its BL.

For each dataset, the performance of the algorithms was computed using the prequential method (interleaved test-then-train): when an unlabelled observation is received, the algorithm is first asked to predict its label and the prediction error is recorded (test). Once the true labelled is realeased, the classifier is trained with this labelled observation (train). This method has the advantage of making use of the whole dataset.

5 Results and discussion

The accuracy (percentage of correct classifications) obtained by each algorithm on each dataset are reported in Table 2. Bold numbers indicate the best performing algorithm. The bottom 2 lines show the average accuracy as well as the average rank obtained by each algorithm on all the datasets.

The results indicate that DEA managed to obtain the best average accuracy as well as the best average rank on the 25 datasets considered. In particular, the average rank obtained demonstrates the ability of DEA to perform consistently well regardless of the characteristics of the dataset and of the type of drifts encountered. This is an interesting property because it is often impossible to predict how the stream will evolve over time and thus, an algorithm which can deal with a very diversified set of environments could be useful as it wouldn't be possible to pick right from the beginning the algorithm which is the best suited for the whole dataset.

This good performance also confirms that using the local expertise of the BL as a selection criteria to decide which subset will be used for prediction should be considered as a way to improve the performances of an ensemble learning algorithm. Indeed, DEA over-performed the ensemble learning algorithms which rely on the latest performances to weight their BL (ADACC, DWM, AUE, ...) as well as a Majority Vote algorithm which simply ask all the algorithms to collaborate for prediction, independently of the observation received.

Dataset	DEA	SAMKNN	ADACC	DWM	Bag Ad	Lev Bag	Hoeff Tree	Boost Ad	AUE	Maj Vote
Agrawal	94.20	92.53	84.83	81.46	**94.87**	93.97	94.24	90.13	90.80	93.62
Rand Tree	51.75	32.27	29.08	30.25	46.93	**52.36**	35.76	19.80	48.35	43.89
Waveform	83.26	82.15	76.35	77.37	82.95	**84.84**	81.55	81.35	80.49	82.87
LED	73.51	71.12	64.67	63.46	73.94	73.88	**73.95**	73.69	73.93	73.86
KDD Cup	**99.91**	99.90	99.84	99.62	99.87	99.91	99.78	98.91	82.44	82.71
Rotating Check	92.94	91.39	79.65	76.70	84.68	93.03	84.39	**93.95**	82.68	87.41
SPAM	**96.60**	94.95	94.91	92.30	90.95	96.04	90.60	95.54	65.36	69.45
Usenet	60.30	57.24	61.00	60.85	56.43	61.95	56.91	59.17	61.07	**62.28**
Airlines	66.40	65.16	62.44	60.25	68.14	65.41	66.26	62.72	67.36	**68.54**
Multi Dataset	95.11	92.56	90.99	89.53	93.80	**95.31**	92.52	93.71	92.90	93.96
Multi Dataset NO	90.25	85.09	55.00	54.39	88.28	**90.86**	88.73	87.93	89.19	88.02
Weather	76.52	76.02	73.56	72.22	75.00	**78.14**	73.53	74.40	74.55	76.74
SEA	87.77	85.45	85.12	84.72	86.77	**88.33**	86.76	82.33	86.85	87.64
Chess	**94.92**	78.22	14.45	13.84	55.35	94.84	58.49	12.64	88.02	76.67
Transient Chess	**94.98**	85.37	58.03	57.07	56.11	89.52	37.27	56.12	26.03	39.68
Mixed Drift	76.64	**91.57**	37.56	35.63	59.54	75.17	55.70	42.76	68.23	64.91
Moving Square	99.08	97.36	**99.14**	83.31	88.02	87.85	74.89	79.66	67.18	88.67
Inter RBF	98.45	98.00	**98.59**	97.14	88.92	94.10	56.81	94.14	91.61	96.17
Moving RBF	59.18	**86.98**	44.65	45.80	52.31	55.02	38.72	45.43	54.70	53.09
Cover Type	92.69	**93.58**	90.38	84.29	84.02	90.40	80.54	92.55	82.69	86.65
Electricity	**90.66**	82.54	89.55	82.34	83.63	89.49	82.83	87.34	78.98	87.32
Outdoor Stream	69.43	**88.25**	66.88	58.32	58.91	60.21	57.20	57.70	40.08	38.63
Poker Hand	88.09	79.77	79.36	75.64	73.46	85.68	65.54	**91.74**	68.65	80.47
Rialto	70.92	**81.90**	71.21	45.27	49.46	60.43	30.65	18.34	47.35	40.76
Rotating Hyp	86.89	81.42	82.78	83.46	88.02	86.87	86.45	76.58	87.20	**88.44**
Average Accuracy	**83.62**	82.83	71.60	68.21	75.21	81.75	70.00	70.74	71.87	74.10
Average Rank	**2.48**	4.72	6.44	7.96	5.28	2.96	7.12	6.72	6.36	4.96

Table 2. Percentage of correct classification achieved on each dataset.

6 Conclusion

Learning on a data stream subject to concept drifts is a challenging task. The hidden underlying distribution on which the algorithm is trying to learn can change in many unexpected ways, requiring an algorithm which is capable of good performances regardless of the environment encountered.

In order to tackle this issue, we have proposed the Droplets Ensemble Algorithm (DEA), a novel algorithm which combines the properties of an instance base learning algorithm with the ones of an ensemble learning algorithm. It maintains into memory a set of hyper-spheres, each of which includes a pointer to the BL which is the most likely to obtain the best performance in the region of the feature space around that observation. When a new observation is received, it selects the BL which are likely to obtain the best performance in this region and use them for prediction.

The experiments carried on a set of 25 diversified datasets, reproducing a wide variety of drifts show that our algorithm is able to over-perform each of its base learners, a majority vote algorithm using the same base learners as well as SAM KNN (one of the currently best adaptive algorithm) by obtaining the best average accuracy and rank. These results indicate that our algorithm is well suited to be used as a general purposed algorithm for predicting on data streams with concept drifts and that taking into account the local expertise of each BL should be considered in order to improve the performances of an ensemble learning algorithm.

The algorithm can still be further improved and future work will focus on improving the efficiency of the search algorithm.

References

[1] Losing, V., Hammer, B., & Wersing, H. (2016). KNN Classifier with Self Adjusting Memory for Heterogeneous Concept Drift, 1. ICDM

[2] Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In Proceedings of the second ieee international conference on computational intelligence in security and defense applications (pp. 53–58).

[3] Katakis, I., Tsoumakas, G., Vlahavas, I.: Tracking recurring contexts using ensemble classifiers: an application to email filtering. In: Knowledge and Information Systems, 22(3), pp. 371–391 (2010)

[4] Brzezinski, D., & Stefanowski, J. (2014). Reacting to different types of concept drift: the Accuracy Updated Ensemble algorithm. IEEE Transactions on Neural Networks and Learning Systems, 25(1), 81–94.

[5] Elwell, R., & Polikar, R. (2011). Incremental Learning of Concept Drift in Nonstationary Environments. IEEE Transactions on Neural Networks, 22(10), 1517–1531.

[6] Kolter, J. Z., Maloof, M.A.: Dynamic weighted majority: A new ensemble method for tracking concept drift. In: Data Mining, 2003. ICDM 2003. Third IEEE International Conference, pp. 123-130 (2013)

[7] Bifet, A., Holmes, G., Pfahringer, B.: Leveraging bagging for evolving data streams. In: Machine Learning and Knowledge Discovery in Databases, pp. 135–150. Springer Berlin Heidelberg (2010)

[8] I. Katakis, G. Tsoumakas, I. Vlahavas, "An Ensemble of Classifiers for coping with Recurring Contexts in Data Streams", 18th European Conference on Artificial Intelligence, IOS Press, Patras, Greece, 2008.

[9] Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD '09.

[10] N. Oza and S. Russell. Online bagging and boosting. In Artificial Intelligence and Statistics 2001, pages 105–112. Morgan Kaufmann, 2001.

[11] Jaber, G., Cornuejols, A., & Tarroux, P. (2013). A new on-line learning method for coping with recurring concepts: The ADACC system. In Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) (Vol. 8227 LNCS, pp. 595–604).

[12] Bifet, A., & Gavaldà, R. (2009). Adaptive Learning from Evolving Data Streams. Proceedings of the 8th International Symposium on Intelligent Data Analysis, 249–260.

[13] Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. Classification and Regression Trees. Wadsworth, 1984.

[14] R. Agrawal, T. Imielinski, and A. Swami. Database mining: A perfor- mance perspective. IEEE Trans. on Knowl. and Data Eng., 5(6):914–925, 1993.

[15] P. Domingos and G. Hulten. Mining high-speed data streams. In Knowledge Discovery and Data Mining, pages 71–80, 2000.