Proceedings of the Federated Conference on
Computer Science and Information Systems pp. 19–29

DOI: 10.15439/2015F378
ACSIS, Vol. 5

Recurrent drifts: applying fuzzy logic to concept similarity function

Abad, Miguel Ángel
Facultad de Informática,
Universidad Politécnica de Madrid,
Campus de Montegancedo,
28660 - Boadilla del Monte,
Madrid - Spain
Email: miguel.abad.arranz@alumnos.upm.es

Menasalvas, Ernestina
Facultad de Informática,
Universidad Politécnica de Madrid,
Campus de Montegancedo,
28660 - Boadilla del Monte,
Madrid - Spain
Email: emenasalvas@fi.upm.es

***Abstract*—Recurrent drift, as a specific type of concept drift, is characterised by the appearance of previously seen concepts. Therefore, in those cases the learning process could be saved or at least minimized by applying an already trained classification model. In this paper we propose Fuzzy-Rec, a framework that is able to deal with recurrent concept drifts by means of a repository of classification models and a similarity function.**

Fuzzy logic is used in the framework to implement the similarity function needed to compare different classification models. This is a crucial aspect when dealing with drift recurrence, as long as some measure must be implemented to determine which model better fits a previously seen context. As it can be seen in the experimentation results of this paper, this fuzzy similarity function provides excellent results both in synthetic and real datasets. As a conclusion, we can state that the introduction of fuzzy logic comparisons between models could lead to a better efficient reuse of previously seen concepts, saving computational resources by applying not just equal models, but also similar ones.

I. Introduction

TRADITIONAL data stream classification [1] aims to learn a classification model from a stream of training records in order to use it later to predict the class of unlabeled records with high accuracy. Most of these kinds of classification models lack an efficient adaptation to the environment where they are implemented which, in most cases, is constantly changing. For this reason, coping with the improvement and adaptation of classification algorithms on data streams is still a great challenge, as long as data stream mining imposes some requirements that have to be accomplished, namely: maintaining an efficient behaviour in the system, i.e. stable computational and memory load; while providing suitable quality in the classification process, i.e. high accuracy of predictions.

Concept drift is known as the intrinsic changes that occur on the data being processed during data-mining tasks. These changes might be caused by data distribution alterations or by the appearance of a new context that alters the relations among the data attributes. Keeping this scenario in mind, different concept drift techniques have been extensively applied to cope with changes in the underlying distribution of records over time, allowing classification models to be able to adapt their behaviour when needed [2, 3, 4].

Moreover, it is common in real-world data streams for previously seen concepts to reappear [5]. This represents a particular case of concept drift [6], known as recurring concepts [7]; [8]; [5]; [9]; [10]. An adequate management of recurrent concept drifts would lead to a better overall data stream learning and classification processes efficiency and efficacy.

Some real cases where concept recurrence is likely to appear are:

1) Product recommendation systems. Drift in these kind of systems is usually related to fashion trends, economy fluctuations or other hidden context. Anyway, in the first two causes recurrence it is likely to reappear. This is due to the fact that fashion and economy trends reappear during time. A system able to deal with concept recurrence could save some precious training time by means of reusing previously seen recommendation models.
2) Weather prediction. The changes that occur in weather predictions are usually recurrent according to the seasons. Therefore, prediction models that deal well with a specific season could be reused latter on time.
3) Intrusion detection systems. An intrusion detection system (IDS) is a typical monitoring problem which aims to detect cyber incidents. In this case, a trained classification model could send alerts to the operator when a malfunction in the system occurs. A concept drift in an IDS means that the system is behaving in a different way from that expected. But that different behaviour may be caused by a new kind of intrusion that is probably taking place, or because the system monitored is changing in a controlled environment (no intrusion is taking place). If we were able to store all the patterns that represent the different situations of the system monitored (its concepts), we could reuse previously seen models easily.
4) Fraud detection. A similar situation like the one explained in the case of IDS, would be the case of a set of systems dealing with fraud detection. Fraud detection systems are able to detect misbehaviours on

978-83-60810-66-8/$25.00

a big amount of data. In this kind of scenarios drift is usually related to economy variations or seasons. As in the previous cases, the detection of drift recurrence could aid in the performance and precision values of the prediction mechanism.

Therefore, there are situations in which a new training is not needed, as the new concept is equal or similar to a previous one. In those cases we could reuse a previously-trained model, saving computation costs and thus providing an efficient method to undertake this new context.

Extending this idea, in this work we propose Fuzzy-Rec as a novel data-stream learning system to help in the process of recurrent concept drift management. In situations where concepts reappear we propose to use a fuzzy similarity function to help in the process of getting the most similar model in a specific context. Some approaches already exist for that goal, but they refer to crisp logic based on true/false values. We assume that a similarity function based on fuzzy logic [11] would improve the similarity process, also allowing us to obtain a better knowledge of what is happening in that process. Furthermore a fuzzy logic similarity function could be adapted for each situation, depending on the feature space of the data stream or on the computational capabilities of the system.

This is a crucial aspect to improve the storing process in the repository effectively. When a model has to be stored in the repository, it is required to know if the concept that the model is representing is recurrent or not. In case of a recurrent concept, it should not be stored, as there are already previous models representing it.

In order to calculate the fuzzy similarity level between models, two main features have been used in this work: i) the level of accuracy of the different models involved regarding a specific set of instances; ii) the number of instances used to train the different models, as a measure of maturity and stability of the model. This process helps the system as a whole to save memory consumption, as long as just the models needed are stored in the repository. In this way, the proposed mechanism makes it feasible to work with complex data-stream environments where an overloaded repository would make it difficult to achieve a suitable quality of the system.

To the best of our knowledge, this is the first work to deal with concept drift by means of the use of fuzzy logic to predict similar previously seen concepts.

Experiments performed with different real and synthetic datasets show that Fuzzy-Rec provides similar precision results when comparing it with other approaches.

The rest of the paper is organized as follows. In Section II, we summarize related work on concept drift and fuzzy logic, which is followed in Section III by the preliminaries of the approach where the motivation, challenges and problem definition are stated. Furthermore, in Section IV, we propose Fuzzy-Rec as a solution to work in recurring concept drift environments, with a detailed description of its components and the algorithm used. Section V presents the results obtained by the experimentation phase. Finally Section VI presents the main conclusions and discussion of future lines of research.

II. Related Work

The approach that we propose in this paper relies on the storage of previously learnt concepts. A fuzzy similarity function is used to retrieve a previously built model which is similar to the current one.

Consequently, in this section we review and compare our proposal with methods that address the problems of: recurring concepts, change detection and conceptual equivalence.

A recent review of the literature related to the problem of concept drift can be found in [4]. Moreover, a review on the challenges for adaptative learning systems have been published in [12].

A. Recurring Concepts

There have been several techniques developed to achieve the challenge that arises when dealing with concept drift, be they algorithms adaptations or wrapper mechanisms. New algorithms have recently appeared [1, 2, 3, 4, 5, 13, 14], but some other related challenges have received far less attention. Such is the case of situations where the same concept or a similar one reappears, and a previous model could be reused to enhance the learning process in terms of accuracy and processing time [7, 8, 10, 15, 16].

In this way, most existing proposals do not exploit this and have to learn new concepts from scratch even if they are recurrent. However, there are some solutions that deal with concept recurrence, as is the case of the work presented by Ramamurthy and Bhatnagar [15]. In this research, the authors present an ensemble approach that exploits concept recurrence, using a global set of classifiers learned from sequential data chunks. If no classifier in the ensemble performs better than the error threshold, a new classifier is learned and stored to represent the current concept. The classifiers with better performance on the most recent data form part of the ensemble for labeling new records. In [17] and [18] an ensemble mechanism is used to deal with concept drift. Similarly, in [8] an ensemble is also used, but incremental clustering is performed to maintain information on historical concepts. In this way, the proposed framework captures batches of examples from the stream into conceptual vectors. Conceptual vectors are clustered incrementally according to their distance and for each cluster a new classifier is learnt. Classifiers in the ensemble are then learnt using the clusters. Recently [19] proposed Learn++.NSE, an extension of [20] for nonstationary environments, Learn++.NSE is also an ensemble approach that learns from consecutive batches of data without making any assumptions on the nature or rate of drift. The classifiers are combined using dynamic weight majority and the major novelty is on the weighting function that uses the classifiers time-adjusted accuracy on current and past environments. To deal with resource constraints [21] proposes a novel algorithm to manage a pool of classifiers when learning recurring concepts. The main drawback of these methods, apart from the computational process time needed, is the need of constantly train the models used being them recurrent or not.

More sophisticated approaches that use drift detection [3] have also been proposed to address concept recurrence, such as [7, 10]. These approaches store learned models and reuse them when a similar concept reappears in the stream, thus avoiding the effort necessary to relearn a previously observed concept. The method proposed by Yang et al. [10] consists of using a proactive approach to recurring concepts, which means reusing a concept from the concept history. This concept history is represented as a Markov chain which allows the most probable concept to be selected according to a given transition matrix. The approach proposed by Gama and Kosina [7] uses the drift detection method presented in [3] to identify stable concepts and it also memorizes learned classifiers that represent these concepts. After a change is detected in situations of recurrence, referees are used to choose the most appropriate classifier to be reused (i.e., the referee prediction on the applicability of the classifier is greater than a pre-defined threshold). [22] is a recent work on drift detection which uses a control chart to monitor the misclassification rate of the data stream classifier. RCD [23] is a recent recurring concept drift framework that uses a non-parametric multivariate statistical tests to check for recurrence. [24] proposes a semi-supervised recurring concept learning algorithm that takes advantage of unlabelled data.

In the approach proposed in [16] context-concept relationships are learnt from the concept history. A model from a previously learnt concept associated with a particular context is reused in situations of recurrence. Moreover, the proposed method does not require the partition of the dataset into small batches. The concept representations are learnt by a base learner algorithm from an arbitrary number of records. These concept boundaries are determined when a drift detection method signals a change/drift. To improve [16], which relies on a single classifier to deal with recurring concepts, the use of ensembles has been proposed in [25].

B. Change Detection

Although online learning systems are able to adapt to evolving data without any additional change detection mechanism, the advantage of explicit change detection is providing information about the intrinsic dynamics of the process generating data. In this way, the change detection module characterizes the techniques and mechanisms for drift detection. One advantage of detection models, is that they can provide a meaningful description (indicating the change-points or small time-windows where the change occurs) and the quantification of changes. They may be divided into two different approaches:

- Monitoring the evolution of performance indicators [5]. Some indicators (e.g., performance measures, properties of the data, etc.) are monitored over time. In the work presented in [26] the monitoring of three performance indicators (accuracy, recall, and precision) has been proposed. Furthermore, a highly referenced work that uses this approach is the FLORA family of algorithms developed by [5].
- Monitoring distributions on two different time-windows. A reference window, which usually summarizes past information, and a window over the most recent records. The work proposed by [27] uses statistical tests based on Chernoff bound to determine if the samples drawn from two probability distributions are different and then decide if a concept change occurred. Also [28, 29, 30] approaches are based on monitoring two different time-windows.

[3] and [9] approaches monitor the error-rate of the learning algorithm to find drift events. In [3], when the learning process error-rate increases above certain pre-defined levels, the method signals that the underlying concept has changed. Alternatively [31] uses the distribution of the distances between classification errors to signal drift. If the distance, which results from more consecutive errors is above pre-defined threshold, the underlying concept must be changing and an event is triggered. The basic adaptation strategy after drift is detected is to discard the old model and learn a new one to represent the new underlying concept [3, 31].

C. Conceptual equivalence

To determine whether a certain model represents a new concept or a reappearing one, a similarity measure is required. The current work is an improvement of the *Conceptual equivalence* measure proposed by Yang et al. [9] where a fuzzy logic function [32] is used to better represent the relationship between different concepts.

III. Problem Definition and Preliminaries

This section provides the necessary background to understand Fuzzy-Rec system. We start by motivating and defining the problem, including some basic definitions to understand the basics of the solution proposed as well as the main challenges we are dealing with in this paper.

From now on we assume that the data streams that are used as input in the Fuzzy-Rec model are already preprocessed and adapted to work well with incremental data stream classification processes. In this way, we can then assume that we do not need to preprocess the data streams, this work being out of the scope of this paper.

A. Motivation

Data stream mining algorithms must come up with the problem of having to keep in memory just a limited number of records to train their models. That is why these algorithms have to cope with the task of processing each training record only once, while maintaining a suitable quality on the resulting model. This is the main difference from traditional data mining algorithms, where multiple passes over data are common.

In particular, that leads to classification techniques on data streams where models have to be learned incrementally with each incoming record. With the availability of these kinds of models, it is feasible to predict the class of unlabeled records anytime from an early stage. Obviously, the more training records the better precision values obtained.

However, in scenarios where the data distribution changes the accuracy of the classification is expected to decrease. In

these cases, to continuously maintain the quality of the models, it is also important for them to be able also to detect and adapt anytime to changes in the underlying concept that they represent [2].

One of the causes of changes in the underlying concept may be recurrence, as a particular case of concept drift [5, 7, 8, 10]. In those cases, a previously learned concept is expected to reappear.

We envisage that recognizing and predicting already learned concepts might help the system to better adapt to future changes where these concepts reappear. With that recognition task in place, it would be possible for the algorithms to avoid relearning something from scratch that has been already learned [5, 7, 8, 10, 16]. This same idea has been already explored in [16], where concepts are able to be saved in a repository.

However in our approach we propose a fuzzy based mechanism to decide about similarity of models, improving the storage of the models. As a result, by means of a good similarity selection, the number of instances needed for the training process is expected to decrease.

B. Preliminaries

1) Learning with Concept Drift: Let X be the space of attributes with its possible values; Y the set of possible discrete class values. Let D be the data stream of training records arriving sequentially $X_i = (\vec{x_i}, y_i)$ with $x_i \in X$ (feature space) and $y_i \in Y$, where $\vec{x_i}$ is a vector of the attribute values and y_i is the (discrete) class label for the i^{th} record in the stream. In order to train a base learner based on a classification model m incrementally, these records are processed by m with the goal of predicting the class label of a new record $\vec{x} \in X$, so that $m(\vec{x}) = y \in Y$.

As stated in [9], the concept term is more subjective than objective. That is why in the scope of this paper a concept is represented by the learning results of the classification algorithm used as a base learner, such a Hoeffding Tree [33].

In this field, we consider that a stable concept has been learned when the records used during a given period k are independently and identically distributed according to a probability distribution $P_k(x, y)$. In these situations where concept change, $P_k(x, y) \neq P_{k+1}(x, y)$.

2) Recurring Concepts : A recurring concept change can be detected when the input records during a period k are generated based on the same distribution as a previously observed period, in a way that $P_k(x, y) = P_{k-j}(x, y)$. To deal with these kinds of situation, the model m_k learned from a certain period k could be saved to be reused later if it is needed. This would avoid the need to learn a new model representing the same concept as m_k. With this solution the continuous learning process improves its behaviour, not requiring a previously learned concept to be learnt from scratch. In addition this approach needs fewer training records to be processed than other approaches that do not deal with recurrent concepts. However, to better calculate whether a concept is recurrent or not, a similarity function is usually used. This is the case of the similarity function proposed in [9], which is the starting point in developing the new fuzzy similarity method proposed in this paper.

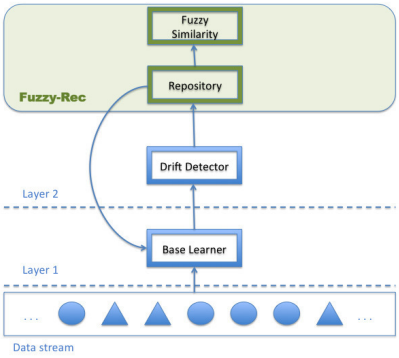

Fig. 1. Fuzzy-Rec Components

C. Challenges

The main challenges we deal with in this paper are:

- Arranging a fuzzy similarity function in order to better calculate the level of similarity between different concepts.
- Improving the precision values of similar recurrent methods.
- Reducing the number of instances needed to train the classification model.

Fuzzy-Rec faces the aforementioned challenges by means of a fuzzy function to deal with concept similarities evaluation.

D. Fuzzy-Rec

The main elements of Fuzzy-Rec, as depicted in figure 1, are:

- A repository of previously seen models.
- A concept similarity function based on fuzzy logic. This function is used to determine the level of similarity between concepts. This fuzzy similarity function is crucial to solve the problem of deciding not just which is the most suitable model, but also if the storage of a specific model in the repository is required.

This is therefore a substantial improvement in the work presented in [16], where the problem of concept drift in recurring scenarios was solved in a similar way also by using a repository of concepts and a crisp similarity function.

The proposed Fuzzy-Rec system allows us to better deal with recurrent situations in a classification problem in data streams, helping the evolving base learner to adapt to drifts. Hence the Fuzzy-Rec system is a feasible tool to be used in a wide range of real application scenarios.

We propose a concept similarity function that uses fuzzy logic and it is based on that presented in [9]. This function is defined by the following parameters:

- A conceptual degree of equivalence based on the matching of two different models classifying many instances,

even when their classifications are both wrong. It is not therefore an accuracy equivalence, but a measure of the level of both models to classify in the same way.
- A measure that represents the difference in the number of records used to train each model. This parameter is intended to provide a measure on the maturity and stability of the model.

From the aforementioned two parameters, a fuzzy [34] similarity value is estimated from a previously defined set of rules, making it possible to obtain the poor, average or high values.

There are two situations where a similarity function is required:

1) A model must be stored in the repository of previously trained concepts: in this case the fuzzy similarity function is used to assess the need to store a new model. If there is a similar model in the repository, storing a similar one would not improve the quality of the classification process while unnecessarily increasing the memory consumption.
2) A drift is taking place and it is time to decide whether the new concept is recurrent or not: in this case, the system has been training two different models in parallel to adapt to the drift in a recurrent way (the current model and a new one).

IV. Implementation of Fuzzy-Rec

As in the case of the MRec system proposed in [16], Fuzzy-Rec can be seen as a two-layer framework:

1) A basic layer where an incremental learning algorithm is able to represent the underlying concept by means of a classification model.
2) An extended layer in which detection and adaptation to concept changes takes place. The detection of recurrent concepts is implemented in this layer. It is also at this level where Fuzzy-Rec implements its fuzzy similarity mechanism.

To provide an in depth knowledge of the implementation of the Fuzzy-Rec system proposed in this paper, first the learning process is described in section IV-A, whereas the description of the fuzzy similarity concepts is presented in section IV-C.

A. The Learning Process

The on-line learning process for the proposed learning system, as well as the method to detect and adapt to recurrent concepts are detailed in Algorithm 1. The process proceeds as follows:

- It continuously processes the records $X_i = \{\vec{x}, y\}$ with $\vec{x} \in X$ as they appear in the Data Stream.
- In line 3, *currentClassifier* represents the base learner classifier that is currently being used to classify unlabeled records. Its prediction y, being right or wrong, on X_i is passed to the drift detection method used to identify the suitable drift level (*stable, warning or drift*), as explained in IV-B.
- If the process is at the normal level (line 7), the base learner represented by the *currentClassifier* is updated with the new training record. This is the same behaviour as in any other traditional data mining model ready to work with data streams.
- In the case of a warning level (line 8), if the repository does not have the *currentClassifier*, or a similar way as referred to in IV-C, the *currentClassifier* is stored. Still at this level (lines 12 and 13), a *newLearner* is updated with the training record; the training record is also added to a $warningWindow$. The $warningWindow$ contains the latest records (which should belong to the most recent concept), and will also be used to calculate the conceptual equivalence and estimate the accuracy of models stored with the current concept.
- When drift is signalled (line 14), until there are enough records (i.e., stability period) in the $warningWindow$ the $newLearner$ is updated. When the stability period is over (line 18) it is compared with repository models in terms of conceptual equivalence as stated in IV-C. If the current underlying concept is recurrent a stored model from the repository is used to represent the recurring underlying concept, otherwise the $newLearner$ is used. It is important to remark that the benefit of implementing a previously seen model is that it does not need to be trained again, as it is supposed to be a stable model. When using the newLearner, it needs to be constantly trained during the learning process as it is still an immature model. Therefore, if newLearner is used there is not a decrease in the number of training instances needed. However, the risk of reusing a not suitable recurrent model is still latent. In those cases, the accuracy of the classification base learner would drop.
- A false alarm (line 24) case is used when a warning is signaled but then the learner returns back to normal without achieving drift. In those cases, both the $warningWindow$ and the $newLearner$ are cleared.

In short, the Fuzzy-Rec system can be seen as a continuous learning process with the following steps:

1) The base learner processes the incoming records from data streams by means of an incremental learning algorithm to generate a decision model m representing the underlying concept. The model m will be used to classify unlabelled records.
2) A drift detection method is continuously monitoring the error-rate of the learning algorithm [3]. When the error-rate goes beyond some predefined levels, the drift detection method signals a *warning* (possible drift) or a *drift*.
3) Throughout the life cycle of the system, two different cases may be used to adapt to changes in the underlying concept: i) the concept similarity method detects that the underlying concept is new, and the base learner has to learn it by processing the current incoming labelled records in an incremental way. ii) the (fuzzy) concept

Algorithm 1 Data Stream Learning Process

Require: Data stream DS, ModelRepository MR

repeat
  Get next record X_i from DS;
  prediction = *currentClassifier.classify(X_i)*;
  DriftDetection.update(prediction);
  switch DriftDetection.level
  case Normal
  currentClassifier.train(X_i);
  case Warning
  if $\neg MR.containsSimilar(currentClassifier)$ **then**
    MR.store(currentClassifier);
  end if
  WarningWindow.add(X_i);
  newLearner.train(X_i);
  case Drift
  repeat
    WarningWindow.add(X_i);
    newLearner.train(X_i);
  until $WarningWindow.size > \tau$ //Stability Period
  if $\neg MR.containsSimilar(newLearner)$ **then**
    currentClassifier = newLearner;
  else
    currentClassifier = MR.getEquivalent(newLearner);
  end if
  case FalseAlarm
    WarningWindow.clear();
    newLearner.delete();
  end switch
until END OF STREAM

similarity method detailed in IV-C detects that the underlying concept is recurrent, and a previous model is applied.

B. Drift Detection Mechanism

The Fuzzy-Rec system needs to know when a concept drift is taking place from the behaviour of a base learner. For this purpose Fuzzy-Rec uses the method proposed by Gama el al. [3]. From this method, it is important to remark the following characteristics:

- The system assumes the observation of periods of stable concepts followed by changes that lead to new stable periods with different underlying concepts.
- The error-rate of the base learning algorithm is considered as a random variable from a sequence of Bernoulli trials.
- The general form of the probability of detecting an error is given by means of a binomial distribution.
- Three different drift levels are defined to manage concept changes: stable or at a control level, warning level and drift or out of control level. These levels represent the confidence of the mechanism of having detected a concept drift.

It also important to note that other similar methods can be used to detect change detections in concepts. Since the Fuzzy-Rec system has been developed as a wrapper mechanism, the specific method used for it is transparent, so it is not necessary to change the learning process.

Having detected a change in the underlying concept, there are some situations in which a concept recurrence appears. In these cases it is worth anticipating to the reappearing concept, in order to improve the learning process efficiency [16]. In order to do so, a concept similarity method must be used.

C. Concept Similarity

To determine whether a certain model represents a new concept or a reappearing one, a similarity measure is required. In this paper, the *Conceptual equivalence* measure is developed by means of a fuzzy logic system [35] where two variables are used to calculate the similarity between two models.

The term "fuzzy logic" was introduced in [34], and is a way of representing many-valued logic, allowing approximate reasoning to be applied through the definition of variables with several truth ranges (from 0 to 1) and rule sets. A rule set determines which fuzzy operator must be used in each case.

By means of using fuzzy logic, it is easy to deal with the concept of partial truth, where a truth value may range from completely true to completely false. In fuzzy logic applications it is common to use linguistic variables to facilitate the implementation of rules and truth values. In this way, a linguistic variable may have several truth values in the same system. These truth values can be seen as subranges of a continuous variable.

In the proposed Fuzzy-Rec system, three linguistic variables are defined:

- The variable $equal\_classified$, used to represent the similarity in the classification precision behaviour of two different models, may take the values: poor, good and excellent.
- The variable $diff\_training$, used to represent the difference that exist in the number of training records used between two different models, may take the values: small and big.
- The variable $similarity$, a variable use to calculate the output of the fuzzy system based on the aforementioned variables, may take the values: poor, average and high.

Therefore, it is assumed to get a value of "high" when calculating the $similarity$ variable, although in some cases the range could be lowered to "average" values, depending on the characteristics of the dataset used.

The variable $equal\_classified$ is based on the method proposed by Yang et al. [9] to calculate its conceptual equivalence. In our case, as it has been outlined, the equivalence between two models when dealing with classification similarity is just one parameter of the global fuzzy function. This parameter is calculated as follows:

1) Given two classification models m_1,m_2 and a sample dataset D_n of n records, it is possible to calculate for each instance X_i=($\vec{x_i}$, y_i) a score, score(D_n) = +1 if (prediction($m_1(\vec{x_i})$) = prediction($m_2(\vec{x_i})$))

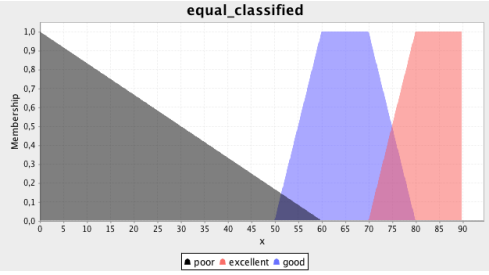

Fig. 2. Membership function of "equal_classified"

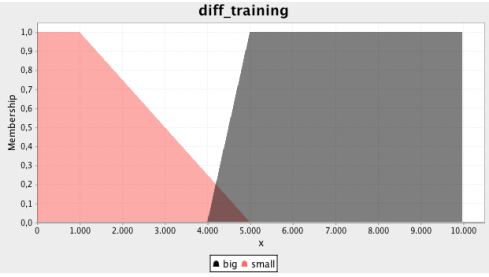

Fig. 3. Membership function of "diff_training"

2) score(D_n) is used to represent the degree of equivalence in the classification process between m_1 and m_2.
3) The final classification equivalence ce value, that is a continuous value score with range [0,1], is calculated by

$$ce = \frac{score(D_n)}{N}$$

Depending on the value of ce, $equal\_classified$ will take one or another membership value, as represented in the figure 2, where we can see the values this variable may take. The larger the output value of ce, the higher the degree of classification equivalence. For the records in D_n it compares how m_1 and m_2 classify the records. As in [9], the similarity in the classification processes is not necessarily related to the accuracy attribute. This means that two models that present low accuracy for a set of records will have a high ce value, and therefore a high $equal\_classified$ value.

As regards the variable $diff\_training$, its value represents the difference in the number of instances used to train each model we are trying to compare. In figure 3 we can see the values this variable may take.

The rule set implemented to develop the fuzzy logic inference is defined as follows:

1) IF $equal\_classified$ IS $poor$ OR $diff\_training$ IS big THEN $similarity$ IS $poor$;
2) IF $equal\_classified$ IS $good$ AND $diff\_training$ IS big THEN $similarity$ IS $poor$;
3) IF $equal\_classified$ IS $good$ AND $diff\_training$ IS $small$ THEN $similarity$ IS $average$;
4) IF $equal\_classified$ IS $excellent$ AND $diff\_training$ is big THEN $similarity$ is $average$;
5) IF $equal\_classified$ IS $excellent$ AND $diff\_training$ is $small$ THEN $similarity$ is $high$;

Finally, a defuzzification method [35] is needed to get a crisp value of the variable measured. In the case of Fuzzy-Rec, "Center Of Gravity" is the method used to calculate the final value of the $similarity$ variable representing the conceptual equivalence, it being a very popular method in which the "center of mass" of the result provides the crisp value.

From the crisp value returned by the defuzzification method, we evaluate if it is above a predefined threshold. In that case, we assume that the models are similar and thus represent the same underlying concept.

It is important to highlight that without the existence of such a fuzzy method the similarity process should be more naive, being able to integrate just one of the aforementioned variables in the process. As a consequence, a biased classification model with a short life cycle but a high performance could be selected as similar to a more mature and stable model. The fuzzy similarity function implemented in Fuzzy-Rec avoids these kind of behaviours, strengthening the similarity process.

V. Experiments

In order to validate the Fuzzy-Rec method, and taking into account that Fuzzy-Rec is an extension of the MRec method cited in [16], two different experiments have been developed:

1) Experiment 1: The goal of this experiment is to prove that the precision of Fuzzy-Rec is similar to the MRec method, and no worse than other methods able to deal with concept drift. In order to do so, accuracy and kappa statistic measures are evaluated.
2) Experiment 2: The goal of this experiment is to prove that the training instances needed by Fuzzy-Rec when drifts appear are fewer than the ones needed when using MRec.

In addition a statistical analysis has been developed to validate the results provided by the execution of experiments 1 and 2.

To sum up, the main goal of this experimentation phase is to test the feasibility of using a fuzzy similarity procedure to determine similar previously seen concept drifts.

A. Parameters setting

To develop the aforementioned experiments, both synthetic and real datasets have been used. A description of the different datasets applied is presented below. Regarding the similarity threshold values needed both for the MRec and Fuzzy-Rec methods, the experiments were developed setting high similarity values; in the specifica case of MRec, a threshold of 0.9 value was used; in the case of Fuzzy-Rec, a similarity value of 0.9 after defuzzification was used. This similarity threshold must be established to afford the comparison process between models. This is important because we must assure that the reused models really fit the context of the data during the

learning process. Hence, lower values of the similarity threshold would lead to reuse models that may be not appropriate to the new concept in course. In contrast, higher values would make MRec and Fuzzy-Rec to look for previously seen models that really fit the concept represented by data. In situations where noise could be present in the data, it is important to set higher similarity threshold values to avoid misconceptions.

Regarding the number of classifiers stored in the repository, 10 was the value set for both experiments.

Below a description of the different datasets used during the experimentation phase is made.

B. Datasets

1) SEA dataset: This synthetic dataset is made up of 1.8M instances, representing two drifts repeated for three times. Specifically this dataset was created by means of the following methodology:

1) Create a dataset with 2 concept drifts, changing from SEA function 1 to function 4.
2) Create a dataset with 2 concept drifts, changing from SEA function 2 to function 3.
3) Merge the previous files for three times in a global dataset to ensure that concepts are mixed and may appear in any time during the execution of the dataset.

2) Hyperplane dataset: A different synthetic dataset with gradual drifting concepts was created based on a moving hyperplane. A hyperplane in d-dimensional space is denoted by equation: $\sum_{i=1}^{d} a_i x_i = a_0$. Instances are labeled as positive if $\sum_{i=1}^{d} a_i x_i \geq a_0$, and as negative if $\sum_{i=1}^{d} a_i x_i < a_0$. Hyperplanes have been used to simulate time-changing concepts because the orientation and the position of the hyperplane can be changed in a smooth manner by changing the magnitude of the weights [13]. This dataset contains 170,000 instances and it represent different recurrent drifts. Taking into account that some noise has been introduced to the dataset, a threshold value of 0.9 is set to ensure that the reused models really fit the concept represented by the data when drifts happen.

3) Electricity dataset: The Electricity Market Dataset (Elec2) [36] is a real dataset that uses data collected from the Australian New South Wales Electricity Market, where the electricity prices are not stationary and are affected by the market supply and demand. The market demand is influenced by context such as season, weather, time of the day and central business district population density. In addition, the supply is influenced primarily by the number of on-line generators, whereas an influencing factor for the price evolution of the electricity market is time. During the time period described in the dataset, the electricity market was expanded with the inclusion of adjacent areas (Victoria state), which led to more elaborate management of the supply as oversupply in one area could be sold interstate.

The Elec2 dataset contains 45,312 records obtained from 7th May 1996 to 5th December 1998, with one record for each half hour (i.e., there are 48 instances for each time period of one day). The class label identifies the change in the price related to a moving average of the last 24 hours. As shown in [36], the dataset exhibits substantial seasonality and is influenced by changes in context. Taking into account that this dataset is expected to have gradual or soft drifts, a similarity threshold of 0.9 is used for this dataset in order to force both MRec and Fuzzy-Rec to reuse just the models associated to concepts really similar to the new appearing one in case of a drift.

4) Sensor dataset: Sensor stream [37] is a real dataset that contains information (temperature, humidity, light, and sensor voltage) collected from 54 sensors deployed in Intel Berkeley Research Lab. The whole stream contains consecutive information recorded over a 2 months period (1 reading per 1-3 minutes) which makes a total of 2,219,803 instances. The learning task of the stream is to correctly identify which of the 54 sensors is associated to the sensor information read. The goal of this experiment is to effectively detect and adapt to the multiple concept drifts that this dataset contains.

Taking into account that recurrent drifts are expected to appear in this dataset, a similarity threshold of 0.9 is set in order to force both MRec and Fuzzy-Rec to use previously seen models just in case there were a high level of certainty of equivalence between concepts.

C. Environment

The implementation of the Fuzzy-Rec learning system was developed in Java, using the MOA [38] environment as a testbed. The specific fuzzy similarity function implemented in Fuzzy-Rec was developed using jFuzzyLogic [39].

During the execution of the different experiments, the following MOA evaluation features were established:

1) The *Prequential-error* method [38] as the main evaluation technique. When using this evaluator, each individual example can be used to test the model before it is used for training, and from this the accuracy can be incrementally updated. Therefore, the results presented in tables I and II have been gathered in this way.
2) The *Naive Bayes* [40] class as base learner.
3) The *SingleClassifierDrift* class as the method in charge of detecting drifts. This class implements the drift detection method of [3] and adapts to drift by learning a new classifier (i.e., discards previous concept representations).

It is important to note that no distributed environment has been available for the execution of the experimentation phase.

In order to develop the statistical analysis R [41] software was used with the "coin" and "multcomp" packages. Taking into account that when comparing several methods over multiple datasets a post-hoc analysis is desired, in this case the post-hoc tests have been developed using the Wilcoxon-Nemenyi-McDonald-Thompson test [42], using the code of [43].

D. Results

A description of the results obtained during the execution of the different experiments presented in the beginning of section V is made below. All the experiments have been executed on the datasets presented in section V-B, and comparisons are made with the following methods:

1) MRec with Naive Bayes class as base learner.
2) The RCD method presented in [23] also using Naive Bayes.

1) Experiment 1: The goal of this experiment was to prove that the precision values (accuracy and kappa statistic) provided by Fuzzy-Rec were better to the ones provided by the MRec and RCD methods.

As it can be seen in table I, when testing the electricity dataset MRec and Fuzzy-Rec improve the RCD precision values, both behaving in a similar way. This is caused by an application of the same recurrent concepts, so in this case the fuzzy similarity function does not improve the precision values. However, in this case it is demonstrates that in the worst case Fuzzy-Rec provides similar precision values to MRec.

In the case of the sensor dataset, Fuzzy-Rec improves the precision values of MRec by applying a better selection of previously seen models. As long as the difference of training instances between models is one of the variables used when comparing models in Fuzzy-Rec, this allows a more precise choice. Due to the fact that MRec does not make use of such variable, it is using in some cases possible similar models that do not count with enough training records, and therefore the similarity process is biased. When comparing Fuzzy-Rec to RCD in this dataset, similar results are obtained.

When using the SEA dataset there is a big difference on the precision values of Fuzzy-Rec when comparing it with MRec method. The reason of this behaviour is the same than before: the fuzzy similarity function allows to make more precise choices of previously seen models. Also in this case the values obtained when using RCD are similar to Fuzzy-Rec.

Lastly, in the case of the hyperplane dataset Fuzzy-Rec is the method that provides better precision results, improving the behaviour of both RCD and MRec methods.

To sum up, we can conclude in this experiment that Fuzzy-Rec provides similar or even better precision values than MRec or RCD methods in all cases. There are no situation in which Fuzzy-Rec behaves worst than the other methods assessed.

2) Experiment 2: The goal of this experiment is to prove that the training instances needed by Fuzzy-Rec when drifts appear are fewer than the ones needed when using MRec and RCD.

The results of this experiments are shown in table II. We can see that except in the case of SEA dataset, Fuzzy-Rec needs fewer training instances than RCD. Comparing these results to the one presented in the previous experiment, we can state that Fuzzy-Rec makes a more efficient use of training instances, improving the precision values of RCD. The reason why Fuzzy-Rec needs more instances when using SEA dataset is due to the high threshold value established to determine similarity. This threshold forces the method to reuse models with a high similarity, which is not reached for this dataset. However, as it has been shown in table I, the precision values are similar to RCD.

Lastly, when comparing the training instances needed by Fuzzy-Rec with the ones needed by MRec, we can see that there are just some slightly differences. As in the previous case, the unique exception is when using the SEA dataset, because of the aforementioned reasons. However, although the instances usage is similar, when comparing these results with the precision values of the previous experiment (see table I), we can conclude that Fuzzy-Rec makes an optimal selection of previously seen models. It is important to note the case of MRec when dealing with SEA dataset, because comparing it with Fuzzy-Rec we can see that although the former makes a lower use of training instances, the precision values obtained drop significantly; in contrast, Fuzzy-Rec while needing more instances provides the better precision results among all the methods assessed. The behaviour of MRec and Fuzzy-Rec with SEA dataset reinforce the idea that the latter provides a more appropriate selection of similar models.

VI. Conclusions and Future Lines of Research

In this paper the Fuzzy-Rec system, has been described as a mechanism to deal with concept drift in recurring situations.

The main contributions of Fuzzy-Rec are:

1) The implementation of a new similarity concept function using fuzzy logic techniques, which helps in the assessment of similarity between concepts in an improved way.
2) The development of Fuzzy-Rec as a wrapper mechanism, allowing it to be used in an easy way with different base learners and drift detector methods. Furthermore, this wrapper mechanism allows the behaviour of the similarity concept function to be parametrized depending on the needs of each dataset or real-world environment.

Fuzzy-Rec has been tested on different synthetic and real datasets, and comparisons have been made with other similar context-aware algorithms able to deal with drift recurrence. The main conclusions obtained from those experiments are that:

- Fuzzy-Rec is the method that provides the better balance between training instances needed and precision values obtained.
- Fuzzy-Rec needs a low rate of training instances as long as it reuses previously seen models.
- The fuzzy similarity function helps to find the most appropriate model without loosing precision.
- Fuzzy-Rec does not decrease the precision values when comparing it with similar methods.

Future lines of research are: i) analysis of a loss function to penalize bad similarity calculus; ii) implementation of a common fuzzy similarity function in distributed environments.

In both cases, an improvement on precision values is foreseen.

References

[1] M. Gaber, A. Zaslavsky, and S. Krishnaswamy, "A survey of classification methods in data streams," *Data Streams*, pp. 39–59, 2007.

[2] A. Tsymbal, "The problem of concept drift: definitions and related work," *Computer Science*

TABLE I
DATASETS PRECISION

	RCD		MRec		Fuzzy-Rec	
Dataset	Acc.	Kappa	Acc.	Kappa	Acc.	Kappa
Elec2	79.2 ± 7.18	56.22 ± 15.43	81.87 ± 5.66	61.71 ± 12.46	81.87 ± 5.66	61.71 ± 12.46
Sensor	85.29 ± 13.92	84.87 ± 14.32	80.99 ± 19.74	80.46 ± 20.32	85.25 ± 14.64	84.82 ± 15.15
SEA	85.06 ± 2.09	67.77 ± 4.41	64.63 ± 22.9	37.31 ± 33.52	85.08 ± 2.04	67.85 ± 4.34
Hyperplane	68.46 ± 10.97	36.91 ± 21.92	80.41 ± 9.99	60.79 ± 19.98	81.61 ± 9.1	63.19 ± 18.18

TABLE II
INSTANCES USED

Dataset	RCD	MRec	Fuzzy-Rec
Elec2	57.41%	16.72%	16.69%
Sensor	21.11%	16.62%	14.23%
SEA	48.6%	6.81%	84.92%
Hyperplane	68.46%	19.41%	20.61%

Department, Trinity College Dublin, 2004. [Online]. Available: http://www.cs.tcd.ie/publications/tech-reports/reports.04/TCD-CS-2004-15.pdf

[3] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," *Lecture Notes in Computer Science*, pp. 286–295, 2004.

[4] I. Žliobaitė, "Learning under concept drift: an overview," *Technical Report. Faculty of Mathematics and Informatics, Vilnius University: Vilnius, Lithuania.*, 2010. [Online]. Available: http://arxiv.org/abs/1010.4784

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

[6] J. Gama, *Knowledge Discovery from Data Streams*, 1st ed. Chapman & Hall/CRC, 2010.

[7] J. Gama and P. Kosina, "Tracking Recurring Concepts with Meta-learners," in *Progress in Artificial Intelligence: 14th Portuguese Conference on Artificial Intelligence, Epia 2009, Aveiro, Portugal, October 12-15, 2009, Proceedings*. Springer, 2009, p. 423.

[8] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: an application to email filtering," *Knowl. Inf. Syst.*, vol. 22, no. 3, pp. 371–391, Mar. 2010.

[9] Y. Yang, X. Wu, and X. Zhu, "Mining in anticipation for concept change: Proactive-reactive prediction in data streams," *Data mining and knowledge discovery*, vol. 13, no. 3, pp. 261–289, 2006.

[10] ——, "Combining proactive and reactive predictions for data streams," in *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM, 2005, p. 715.

[11] W. Kosinski, P. Prokopowicz, and D. Slezak, "Calculus with fuzzy numbers," in *Proceedings of the Second international conference on Intelligent Media Technology for Communicative Intelligence*, ser. IMTCI'04. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 21–28.

[12] I. Žliobaitė, A. Bifet, M. M. Gaber, B. Gabrys, J. Gama, L. L. Minku, and K. Musial, "Next challenges for adaptive learning systems," *SIGKDD Explorations*, vol. 14, no. 1, pp. 48–55, 2012.

[13] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM New York, NY, USA, 2001, pp. 97–106.

[14] W. 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 New York, NY, USA, 2001, pp. 377–382.

[15] S. Ramamurthy and R. Bhatnagar, "Tracking recurrent concept drift in streaming data using ensemble classifiers," in *Proc. of the Sixth International Conference on Machine Learning and Applications*, 2007, pp. 404–409.

[16] J. Bartolo Gomes, E. Menasalvas, and P. Sousa, "Tracking recurrent concepts using context," in *Rough Sets and Current Trends in Computing, Proceedings of the Seventh International Conference RSCTC2010*. Springer, 2010, pp. 168–177.

[17] D. Brzeziński and J. Stefanowski, "Accuracy updated ensemble for data streams with concept drift," in *Proceedings of the 6th international conference on Hybrid artificial intelligent systems - Volume Part II*, ser. HAIS'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 155–163.

[18] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2013.

[19] 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, 2011.

[20] M. Muhlbaier, A. Topalis, and R. Polikar, "Learn++. nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, 2009.

[21] M. J. Hosseini, Z. Ahmadi, and H. Beigy, "New management operations on classifiers pool to track recurring concepts," in *Data Warehousing and Knowledge Discovery*. Springer, 2012, pp. 327–339.

[22] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recogn. Lett.*, vol. 33, no. 2, pp. 191–198, Jan. 2012.

[23] P. M. Gonçalves Jr and R. S. M. D. Barros, "RCD: A Recurring Concept Drift Framework," *Pattern Recogn. Lett.*, vol. 34, no. 9, pp. 1018–1025, Jul. 2013.

[24] P. Li, X. Wu, and X. Hu, "Mining recurring concept drifts with limited labeled streaming data," *ACM Trans. Intell. Syst. Technol.*, vol. 3, no. 2, pp. 29:1–29:32, Feb. 2012.

[25] J. a. B. Gomes, E. Menasalvas, and P. A. C. Sousa, "Learning recurring concepts from data streams with a context-aware ensemble," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11. New York, NY, USA: ACM, 2011, pp. 994–999.

[26] R. Klinkenberg and I. Renz, "Adaptive information filtering: Learning in the presence of concept drifts," in *Learning for Text Categorization*. Menlo Park, California: AAAI Press, 1998, pp. 33–40.

[27] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting Change in Data Streams," in *Proceedings of the Thirtieth International Conference on Very Large Data Bases - Volume 30*, ser. VLDB '04. VLDB Endowment, 2004, pp. 180–191.

[28] A. Dries and U. Rückert, "Adaptive Concept Drift Detection," *Stat. Anal. Data Min.*, vol. 2, no. 5–6, pp. 311–327, Dec. 2009.

[29] I. Adä and M. Berthold, "EVE: a framework for event detection," *Evolving Systems*, vol. 4, no. 1, pp. 61–70, 2013.

[30] K. Nishida and K. Yamauchi, "Detecting Concept Drift Using Statistical Testing," in *Proceedings of the 10th International Conference on Discovery Science*, ser. DS'07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 264–269.

[31] 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*. Citeseer, 2006, pp. 77–86.

[32] J. Mendel, "Fuzzy logic systems for engineering: a tutorial," *Proceedings of the IEEE*, vol. 83, no. 3, pp. 345 –377, mar 1995.

[33] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM New York, NY, USA, 2000, pp. 71–80.

[34] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965.

[35] E. Cox, "Fuzzy fundamentals," *Spectrum, IEEE*, vol. 29, no. 10, pp. 58 –61, oct. 1992.

[36] M. Harries, "Splice-2 comparative evaluation: Electricity pricing. Technical report, The University of South Wales," 1999.

[37] X. Zhu, "Stream Data Mining Repository - http://www.cse.fau.edu/~xqzhu/stream.html," 2010.

[38] G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis, 2007 - http://sourceforge.net/projects/moa-datastream/," 2007.

[39] P. Cingolani and J. Alcala-Fdez, "jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation," in *Fuzzy Systems (FUZZ-IEEE), 2012 IEEE International Conference on*, june 2012, pp. 1 – 8.

[40] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo: Morgan Kaufmann, 1995, pp. 338–345.

[41] R Development Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2010, ISBN 3-900051-07-0. [Online]. Available: http://www.R-project.org

[42] M. Hollander and D. A. Wolfe, *Nonparametric Statistical Methods*. Wiley–Interscience, 1999.

[43] T. Galili, "Post-hoc analysis for Friedman test. Code available in http://www.r-statistics.com/2010/02/post-hoc-analysis-for-friedmans-test-r-code," 2010.