

# Fixed-size ensemble classifier system evolutionarily adapted to a recurring context with an unlimited pool of classifiers

Konrad Jackowski

Received: 29 February 2012 / Accepted: 15 January 2013 / Published online: 5 February 2013  
© Springer-Verlag London 2013

**Abstract** This paper presents a novel ensemble classifier system designed to process data streams featuring occasional changes in their characteristics (concept drift). The ensemble is especially effective when the concepts reappear (recurring context). The system collects information on emerging contexts in a pool of elementary classifiers trained on subsequent data chunks. The pool is updated only when concept drift is detected. In contrast to other ensemble solutions, classifiers are not removed from the pool, and therefore, knowledge of past contexts is preserved for future use. To ensure high classification performance, the number of classifiers contributing to decision-making is fixed and limited. Only selected elements from the pool can join the decision-making ensemble. The process of selecting classifiers and adjusting their weights is realized by an evolutionary-based optimization algorithm that aims to minimize the system misclassification rate. Performance of the system is evaluated through a series of experiments presenting some key features of the system.

**Keywords** Concept drift · Multiple classifier systems

## 1 Introduction

In the early days of machine learning research on classifier systems, it was assumed that the environment is stable, i.e., prior class probability and the conditional distribution of

objects in classes do not change over time. However, a recently emerged trend focuses on the range of applications in data stream mining [1] where some changes in data characteristics occur [2, 3]. One of the most well-known examples of this problem is SPAM detection and filtering. User perceptions of what is considered to be SPAM and the attack methods used, have evolved over time. Other well-known applications address problems of fraud detection [4], credit application [4], and market analysis [5].

The possibility of change naturally makes static classification systems irrelevant [6, 7]. To counteract this negative effect, an appropriate response is required, which should be based on continuous monitoring of incoming data and applying amendments to the systems. Knowledge of the nature of the change can help design an appropriate algorithm.

First, it is important to identify the places where drift may occur. From a probabilistic point of view, the changes can relate to any of the following parameters [8]:

1. Prior probability of classes, where the proportion of each class in the population changes (or new classes emerge [9]);
2. Conditional distribution of objects in classes, where values of the object features drift in the feature space;
3. Posterior probability that a given object belongs to a particular class.

Change in the first case is called virtual concept drift, while that in the other cases is called real concept drift [10]. Typically, drift can affect all these parameters.

Considering the possible reaction to drift, one of the main questions is how to effectively recognize the moment when drift appears. This question is crucial because a fast response to change helps avoid deterioration of the classifier performance [11, 12]. This problem is not trivial, as a

---

K. Jackowski (✉)  
Department of Systems and Computer Networks, Wrocław  
University of Technology, Wybrzeże Wyspiańskiego 27,  
50-370 Wrocław, Poland  
e-mail: konrad.jackowski@pwr.wroc.pl

drift detector must be able to distinguish precisely between drift and fluctuations caused by noise [2, 13].

Another important factor, which determines the possible reaction to drift, is the way in which changes appear. A well-known taxonomy in this regard lists the following types of drift [14]: (1) sudden, (2) gradual, and (3) incremental concept drift. In the first case, essential changes appear suddenly at a particular moment in time (e.g., changes in accent for speech recognition). In the other two cases, changes occur in an evolutionary manner and are spread over time (e.g., seasonal characteristics of morbidity). Knowledge of the type of drift allows selection of the appropriate scenarios for system adjustment. For example, replacing an old classifier with a new one, which has been trained on recent data, could be an effective strategy when abrupt drift affects the data. On the other hand, when changes are spread over time, evolutionary adaptation of the classifier may be a better alternative.

It is usually assumed that changes are not reversible, i.e., although the current state is not permanent and further drift can appear, the past context does not occur again. This fact allows the possible application of a different kind of forgetting mechanism, which can relate either to instances that arrive in the data stream, or to classifiers that are included in the ensemble system [15, 16]. However, yet another category of drift has been identified in the literature, Called recurring context, it relates to the situation when past contexts re-emerge. The moment of their recurrence and the sequence thereof are unknown [2, 17]. The prospect of context reappearance makes the forgetting strategy useless. If there is no knowledge of previous contexts, re-emergence of a past context is treated in the same manner as the emergence of a new one. On the other hand, reference to a previously acquired experience allows for significant savings in terms of time and cost of learning. One of the most popular approaches for this category of drift is the exploitation of ensemble classifiers [18], the strength of which stems from accumulating knowledge of contexts in a set of elementary classifiers and making a collective decision [19, 20].

Analysis of the nature of context drift and a review of existing solutions has lead to the following conclusions:

1. Preserving once acquired knowledge of contexts for future use, rather than irreversible forgetting, is a better solution when dealing with recurring contexts;
2. Evolutionary adaptation of the classifier to new conditions, rather than abrupt reconstruction of the system, works better when the drift coincides with natural fluctuations in the data caused by noise or other factors;
3. Weighted fusion of classifiers improves classification accuracy;

4. Limiting the number of classifiers contributing to decision-making prevents spoiling the classification by the irrelevant voices of outdated classifiers and also reduces processing time;
5. Drift detection should be robust and noise-resistant.

The Evolutionary-Adapted Ensemble (EAE), proposed in this paper, addresses the different and sometimes conflicting demands (points 1, 3, and 4 above). The main features of the algorithm are as follows:

1. The system gathers knowledge on the contexts in the form of elementary classifiers stored in a pool with unlimited size.
2. A collective decision is made by a committee of selected classifiers. The size of the committee is fixed and limited. The weighted fusion of the discriminating functions of the classifiers in the committee forms the basis of the decision.
3. The system continuously adjusts committee members and their weights while processing subsequent data chunks.
4. A new classifier is created and trained on the recently processed data chunk, if the system cannot maintain classification accuracy using the classifiers in the pool.
5. A learning algorithm is implemented as a compound optimization task. It uses an evolutionary algorithm that constantly processes the population of individuals representing the set of possible solutions.

As the EAE is a batch processing algorithm, it can be used in a wide range of applications without the strict resource and processing time limitations characteristic of online classifiers.

The rest of the paper is organized as follows: Section 2 provides a review of related works on classifier systems working under concept drift, while Sect. 3 provides details of the proposed EAE model and its learning algorithm. Results of the experimental evaluation of the algorithm performance are presented in Sect. 4. Section 5 concludes the paper and presents some guidelines for further work.

## 2 Related works

The first algorithms designed to deal with drifting data were STAGGER proposed by Schlimmer and Granger [21], IB3 proposed by Aha [22], and the suite of FLORA algorithms by Widmer and Kubat [2]. Since then, a plethora of solutions has been proposed, which, together with the growing interest in the domain, has resulted in an increasing number of publications. Basically we can divide these algorithms into four main groups: (1) online learners, (2) instance-based solutions, also called sliding-window-based

solutions, (3) ensemble approaches [18, 23], and (4) drift detection algorithms.

The first group relates to the family of algorithms that continuously update the classifier parameters while processing incoming data. Not all types of classifiers can act as online learners; they have to meet some basic requirements [24]:

- Each object must be processed only once in the course of training;
- The system should consume only limited memory and processing time irrespective of the execution time and amount of data processed;
- The training process can be paused at any time and its accuracy should not be lower than that of a classifier trained on batch data collected up to the given time.

Classifiers that fulfill these requirements work very fast and can adapt their model in a very flexible manner. Among the others, the following are the most popular online learners: naive Bayes, Neural Networks [25], and Nearest Neighbor [22]. A more sophisticated solution can be found in [26], that is, the CVFDT algorithm, an extended version of the ultra fast decision tree, which ensures consistency with incoming data by maintaining alternative subtrees. CVFDT replaces the outdated tree when its respective alternative is more accurate. Selected online learners have been incorporated into the Massive Online Analysis framework (MOA) [27], an open framework dedicated to the implementation and testing of online algorithms working with changing data streams.

The second group consists of algorithms that incorporate the forgetting mechanism. This approach is based on the assumption that the recently arrived data are the most relevant, because they contain characteristics of the current context. However, their relevance diminishes with the passage of time. Therefore, narrowing the range of data to those that were most recently read may help form a dataset that embodies the actual context. There are three possible strategies here:

- Selecting the instances by means of a sliding window that cuts off older instances [2];
- Weighting the data according to their relevance; and
- Applying bagging and busting algorithms that focus on misclassified instances [28, 29].

When dealing with the sliding window the main question is how to adjust the window size. On the one hand, a shorter window allows focusing on the emerging context, though data may not be representative for a longer lasting context. On the other hand, a wider window may result in mixing the instances representing different contexts. Therefore, certain advanced algorithms adjust the window size dynamically depending on the detected state

(e.g., FLORA2 [2] and ADWIN2 [30]). In more sophisticated algorithms, multiple windows may even be used [31]. In [32] authors present algorithm incorporating active learning method which automatically decides which data will be used as training samples.

In object weighting algorithms the relevance of the instance is used to calculate its weight, which is usually inversely proportional to the time that has passed since the instance was read [15, 33].

The third group consists of algorithms that incorporate a set of elementary classifiers [34, 35, 42]. The idea of ensemble systems is not new and their effectiveness has been proven in static environments [19]. It has been shown that a collective decision can increase classification accuracy because the knowledge that is distributed among the classifiers may be more comprehensive. This premise is true if the set consists of diverse members [36]. In static environments, diversity may refer to the classifier model, the feature set, or the instances used in training. In a changing environment diversity can also refer to the context. This makes ensemble systems interesting for researchers dealing with concept drift.

Several strategies are possible for a changing environment:

1. Application of *dynamic combiners*, where individual classifiers are trained in advance and their relevance to the current context is evaluated dynamically while processing subsequent data. The level of contribution to the final decision is directly proportional to the relevance. This strategy was used in [37, 38]. The drawback of this approach is that all contexts must be available in advance; emergence of new unknown contexts may result in a lack of experts.
2. *Updating the ensemble members*, where each ensemble consists of a set of online classifiers that are updated incrementally based on the incoming data [27, 28, 39, 43]. Some advanced approaches exploit different types of classifiers to increase diversity [40].
3. *Structural changes in the ensemble*, where ensemble classifiers are evaluated dynamically and the worst one is replaced by a new one trained on the most recent data.

Among the most popular ensemble approaches, the following are worth noting: the Streaming Ensemble Algorithm (SEA) [41] or the Accuracy Weighted Ensemble (AWE) [42]. Both algorithms keep a fixed-size set of classifiers. Incoming data are collected in data chunks, which are used to train new classifiers. If there is a free space in the ensemble, a new classifier joins the committee. Otherwise, all the classifiers are evaluated based on their accuracy and the worst one in the committee is replaced by a new one if the latter has higher accuracy. The SEA uses a majority voting strategy, whereas the AWE uses the more

advanced weighted voting strategy. A similar formula for decision-making is implemented in the Dynamic Weighted Majority (DWM) algorithm [43]. Nevertheless, unlike the former algorithms, the DWM modifies the weights and updates the ensemble in a more flexible manner. The weight of the classifier is reduced when the classifier makes an incorrect decision. Eventually the classifier is removed from the ensemble when its weight falls below a given threshold. Independently, a new classifier is added to the ensemble when the committee makes a wrong decision.

The final group consists of algorithms that address the question of when drift occurs. Not all classification algorithms dealing with concept drift require drift detection. Some evolving systems continuously adjust the model to incoming data [44]. This technique is called implicit drift detection [11] as opposed to explicit drift detection methods that raise a signal to indicate change [11, 12]. The detector can be based on changes in the probability distribution of the instances [9, 45, 46] or classification accuracy [47, 48, 49].

Among the machine learning methods dealing with concept drift, a new class has recently emerged [50], comprising algorithms that process data streams featuring a recurring context. Two additional requirements are imposed on algorithms in this class:

- The system should maintain knowledge of previously emerged contexts, and
- It should be effective in recognizing contexts and switching to a valid one.

Both issues can be effectively solved by an ensemble system. For example, in [51] the authors propose collecting context oriented learners in a “global set” of classifiers along with their selection procedure. The ensemble consists of classifiers that achieve arguably better results than a random classifier.

To address the second issue, certain algorithms use a conceptual representation of contexts. This idea originates from Turney’s definition of context-sensitive features [52] and the early work by Widmer [53] on meta-learning algorithms. It is based on mapping attributes into contextual clues representing some characteristic features of the respective context. Changes in the clues may indicate drift. In [54], the authors exploit an additional similarity measure between context representations for weighting a classifier’s contribution to the ensemble. The method presented in [55] incorporates a stream clustering algorithm, which aims to group incoming data batches automatically based on their conceptual representation. An incremental classifier is created (and updated if needed) for each cluster/context and stored in the pool. A similar solution can be found in [56], but here classifiers in the pool are adaptively weighted according to their performance on a recent data batch.

### 3 Evolutionary-adapted ensemble for concept drift

Techniques that are based on forgetting (applied implicitly or explicitly) have limited usefulness in applications that process streams with a recurring context because the information on past contexts is lost over time. Probably the most appropriate for this sort of application is the ensemble-based approach, in which the knowledge of contexts can be stored in a set of elementary classifiers. The possibility of the context reappearing demands that the classifiers are not removed from the ensemble. On the other hand, the ensemble cannot contain too many experts, because irrelevant voters could spoil the collective decision. A possible solution is as follows: all classifiers are permanently stored in the pool of available classifiers, but only selected classifiers join the ensemble.

Selecting the experts and adjusting their weights based on their accuracy seems to be intuitive and certainly easy to implement. However, the model for selection falls into the category of greedy optimization algorithms, which do not guarantee reaching an optimal solution as they tend to fall into local minima. Therefore, it seems reasonable to use various heuristic optimization algorithms aimed at minimizing the misclassification rate of the entire ensemble system.

Among the many possible techniques for making a collective decision, majority voting is regarded as the most intuitive. However, it is not appropriate when dealing with voters of diverse competence. Weighted voting allows much higher flexibility and therefore, is much more satisfactory.

The final question is when the pool should be updated. Periodic unconditional updates lead to an infinitely increasing pool. Conversely, updates triggered by a drift detector update (assuming that the number of contexts is finite and strictly limited) extend the pool size to a limited degree only.

Taking all these issues into consideration, we propose imposing the following assumptions on the proposed EAE algorithm:

1. The EAE is intended to work in a changing environment with a recurring context. It should gather knowledge of the emerging contexts in a pool of elementary classifiers.
2. The EAE should choose a subset of classifiers from the pool to create a decision-making committee with a fixed size.
3. The EAE committee should make a decision according to the weighted fusion of the discriminating function strategies [19].
4. Selecting the committee members and adjusting their weights should be realized in continuous training routines based on processing subsequent data chunks. The process should aim to minimize the EAE misclassification rate.

- 5. The training process should automatically detect new and recurring contexts.

### 3.1 Ensemble classifier model

We start by introducing some basic terms and symbols. Assume that we have a pattern recognition problem with  $M$  classes. Let  $\Pi^\Psi$  denote the pool of all elementary classifiers and  $K$  denote the pool size.

$$\Pi^\Psi = \{\Psi_1, \Psi_2, \dots, \Psi_K\} \tag{1}$$

Let  $\Xi^\Psi$  denote the set of indexes of classifiers that are taken from the pool to join the ensemble.

$$\Xi^\Psi = \{c_1, c_2, \dots, c_E\} \tag{2}$$

According to assumption 2 given above,  $E \leq K$ .

Let  $W^\Psi$  denote the set of weights assigned to the ensemble members

$$W^\Psi = \{w_1, w_2, \dots, w_E\}, \tag{3}$$

and  $d_{e,i}(x)$  denote the discriminating function of the classifier from  $\Pi^\Psi$  pointed to by index  $c_e$  supporting the  $i$ th class [19].

Now, the formula describing the decision made by the ensemble is given by (4).

$$\bar{\Psi}(x) = \arg \text{Max}_{i=1}^M \sum_{e=1}^E w_e d_{e,i}(x), \tag{4}$$

where  $\bar{\Psi}$  denotes the ensemble classifier response, that is, class label.

According to assumption 4, the learning procedure should minimize the misclassification rate of the ensemble. The rate can be estimated over the learning set, which consists of pairs of instances together with the associated correct class label.

$$LS = \{(x_1, j_1), (x_2, j_2), \dots, (x_N, j_N)\}, \tag{5}$$

where  $x_n$  denotes the feature vector of the  $n$ th instance in the set and  $j_n$  its class label.

### 3.2 Objective function

The objective function to minimize the misclassification rate of the ensemble is given by (6).

$$Q^{\bar{\Psi}} = \frac{1}{N} \sum_{n=1}^N L(\bar{\Psi}(x_n), j_n) = \frac{1}{N} \sum_{n=1}^N L\left(\arg \text{Max}_{i=1}^M \sum_{e=1}^E w_e d_{e,i}(x), j_n\right), \tag{6}$$

where  $L$  is the loss function

$$L(a, b) = \begin{cases} 1 & a \neq b \\ 0 & \text{else} \end{cases} \tag{7}$$

Searching for the minimum of the objective function (6) is a complex optimization problem. The solution can be found by selecting the ensemble members and setting their weights at the same time. To do so, the algorithm has to update two variables in Eq. (6):

- the set of selected classifiers  $\Xi^\Psi$ , and
- their weights stored in  $W^\Psi$ .

The final issue that has to be solved is the implementation of an effective detector. The proposed procedure updates the pool when one of the following conditions is satisfied:

1. The size of the pool is smaller than the desired ensemble size. This condition is satisfied at the beginning of stream processing when the first batches of data are processed;
2. The ensemble misclassification rate increases while processing a subsequent data batch. This situation is likely to occur when the EAE training procedure cannot find appropriate experts in the pool to maintain the accuracy of the ensemble.

There are two possible reasons for an increased ensemble misclassification rate: (a) a new context is emerging, and (b) incoming data are spoiled by noise. In the first case, it would be more desirable to create not one, but a set of classifiers, which would replace all outdated classifiers in the committee with experts in the given context. In the second case (i.e., when the context remains the same), it is not obligatory to refresh the entire committee. Therefore, it would be more appropriate to create just a single expert and evaluate the ensemble performance. If needed, a further expert could be added in the next step.

### 3.3 Evolutionary-adapted ensemble

All the above considerations led to the implementation of the EAE learning procedure as an evolutionary algorithm [57]. Apart from the author’s extensive experience with exploiting evolutionary algorithms for learning ensemble systems [58], other, even more important conditions affected the choice. As stated in the introduction, we believe that evolutionary adaptation of the ensemble to changing conditions may improve the stability of training. Therefore, an evolutionary algorithm seems to fit perfectly. The EAE consists of functions common to all evolutionary algorithms: initialization of the population, and mutation and crossover operators.

During optimization, which is realized by the EAE, the population of individuals is processed in a series of

generations. Each individual represents a possible solution, which is encoded in the form of compound chromosomes. A chromosome consists of two components (8):

$$\text{Ch} = \begin{cases} W^\Psi \\ \Xi^\Psi \end{cases} = \begin{cases} [w_1, w_2, \dots, w_E] \\ [c_1, c_2, \dots, c_E] \end{cases}. \quad (8)$$

### 3.4 Initialization of the population

Usually evolutionary algorithms begin with initialization of the population where chromosomes are filled with randomly selected values. In the EAE, the procedure was modified because: (1) when the first data chunk is processed, the algorithm starts with an empty pool; and (2) the population processed with the  $n^{\text{th}}$  data chunk should be used as the initial population when the next chunk arrives. Pseudo code for the *Init Population* routine is given in Listing 1.

The procedure begins by creating a population of empty individuals or filling the population with individuals remaining after processing the previous data chunk (lines 12 and 13, respectively). If the number of classifiers in the pool is smaller than the ensemble size, a new classifier is created (lines 17–19) and its index is added to all the chromosomes with a random weight (lines 20–24).

```

1. Function Init Population
2. Input:
3.     Previous population,
4.     Pool Learning Set
5.     Pool
6.     Ensemble Size
7.     Population Size
8. Output:
9.     Population
10. Begin
11.     If (Previous population is empty)
12.         Create Population
13.     Else
14.         Population = Previous population
15.     EndIf
16.     If (size of Pool < Ensemble Size)
17.         Create New Classifier
18.         Train Classifier with Pool Learning Set
19.         Add Classifier to Pool
20.         For each Individual in Population
21.             Increment size of  $\Xi$  and  $\Psi$ 
22.             Set Individual. $\Xi$ (last) to New Classifier
23.             Set Individual. $\Psi$ (last) to random number
24.         EndFor
25.     EndIf
26. End

```

### 3.5 Data stream

It is important to mention how data chunks read from the stream are pre-processed before they are used in the EAE. An incoming data chunk is randomly split into four separate datasets:

1. Pool training set—used for training elementary classifiers,
2. Ensemble training set—used for ensemble learning,
3. Validating set—used for overtraining detection,
4. Testing set.

The size of the learning sets is crucial for maintaining high accuracy and avoiding the risk of overfitting.

However, it is hard to answer the question: what is the optimal size of the set, since the answer strongly depends on the complexity of the recognition tasks (i.e., number of classes, number of features, distribution of the classes in the feature space, and extent and frequency of context drift). For the purpose of the experiments presented in this paper, we assume that each set consists of at least 100 instances. Nevertheless, it must also be emphasized, that creating separate learning sets has been proposed to ensure the highest level of generalization. In the case of limited data or a rapidly changing environment when the chunk size should be as small as possible, only one set can be used for training and the size of the validation set can also be reduced by half.

### 3.6 Mutation and crossover operators

Adjusting chromosome components is realized by means of the crossover and mutation operators. The former is the standard two-point crossover operator [57], which affects both chromosome components. As the crossover produces an offspring by exchanging parts of the parent chromosomes, the same classifier may appear at different positions in the chromosome which means that the same classifier is used twice. Nevertheless, we decided to tolerate this situation, since the computational complexity of a procedure that avoids redundancy, would be too high. If redundancy of one classifier in the chromosome reduces ensemble efficiency, the instance is removed from the population by the selection procedure.

The mutation operator (presented in Listing 2) is somewhat more complex and affects both chromosome components in a different manner. Usually mutation of vectors of real numbers is realized by adding some random noise generated with a Gaussian distribution. The EAE uses this procedure to update the weight vector in the chromosome. However, it should be remembered, that weights must be in the range between 0 and 1. Therefore, the vector is scaled at the end of mutation to satisfy this condition (lines 13, 14). The vector of indexes is mutated by replacing one randomly selected component with the index of another classifier. The mutation procedure is usually completely blind, i.e., there is no control over the changes, which are entirely random. However, based on the author's experience, in compound optimization problems, blind mutation could lead to loss of stability. Therefore, in the EAE, mutation of the indexes vector is slightly supported. The probability of classifier selection is directly proportional to its accuracy, i.e., it is more likely that a classifier with high accuracy will be chosen to join the ensemble during mutation. It must be emphasized that this does not mean that only the best classifiers can be selected. Selection is realized according

to roulette selection and weak classifiers could also join the ensemble (lines 10–13).

```

1. Function Mutation
2. Input:
3.   Parents
4.   Pool
5. Output:
6.   Children
7. Begin
8.   For each Individual in Parents
9.     Child = Individual
10.    Select randomly index in E
11.    Select new Classifier from Pool
12.     With Roulette Selection
13.    Set Child.E(index) to new classifier
14.    Set Child.W(index) to random number
15.    Normalize Child.W
16. EndFor
17. End
    
```

### 3.7 Fitness function

The quality of the ensemble represented by a particular individual is measured using a fitness function, which calculates the number of correctly classified samples stored in the ensemble training set using Eq. (6). Two additional functions were also implemented to detect overtraining and drift, respectively.

#### 3.7.1 Overtraining detector

Listing 3 gives the pseudo code for the overtraining detector. The procedure is intended to protect the learning process against overtraining [59]. It evaluates the best individual in the population using the validation set, which does not overlap with the learning set (lines 10, 11). The procedure is launched at the end of each EAE generation. The best individual in the population is evaluated using the validation set and the result is compared with the previous one. If the current result is better than the previous one, overtraining has not been detected and the population is saved as the last one not affected by overtraining (lines 12–15). Otherwise, overtraining has occurred and the population is not saved, although the training process is not cancelled. When the EAE finishes processing, the last saved population is returned as the final population.

```

1. Function Detect Overtraining
2. Input:
3.   Population
4.   Not over-fitted population
5.   Last Result
6. Output:
7.   Not over-fitted population
8.   Last Result
9. Begin
10.  Get Best from Population
11.  Set Result to Evaluate on Validating Set
12.  If (Result < Last Result)
13.    Last Result = Result
14.  Set Not over-fitted population = Population
15. EndIf
16. End
    
```

#### 3.7.2 Drift detector

Pseudo code for the drift detector is given in Listing 4. This routine aims to detect deterioration in EAE performance. Possible reasons and strategies were discussed at the beginning of this section. Two threshold parameters (drift and noise thresholds) are used to detect concept drift and other minor performance fluctuations. The procedure starts by evaluating the best individual using the ensemble learning set (lines 15, 16). The result is compared with that obtained while processing the previous data chunk. Slight deterioration in the accuracy (greater than the noise threshold but less than the drift threshold) could indicate incidental fluctuations in the data stream or the appearance of noisy data. In this case, only one new classifier is added to the pool (lines 17–19). Significant differences (greater than the drift threshold) could indicate occurrence of a new context. In this case, all classifiers in the ensemble are replaced (lines 20–22). Regardless of the reason, the procedure can, if necessary, update the pool with a new classifier and increase the number of generations to allow the new classifier to be selected to join the committee (lines 23–29).

```

1. Function Detect drift
2. Input:
3.   Population
4.   Pool Learning Set
5.   Ensemble Training Set
6.   Ensemble Size
7.   Last Result
8.   Drift Threshold
9.   Noise Threshold
10.  Update Counter
11. Output:
12.  Last Result
13.  Max Generation
14. Begin
15.  Get Best from Population
16.  Set Result to Evaluate on Ensemble Training Set
17.  If (Result > Last Result + Noise Threshold)
18.    Update Counter = 1
19.  EndIf
20.  If (Result > Last Result + Drift Threshold)
21.    Update Counter = Ensemble Size
22.  EndIf
23.  If (Update Counter > 0)
24.    Create Classifier
25.    Train Classifier with Pool Learning Set
26.    Add Classifier to Pool
27.    Set Max Generation to 2 * Max Generation
28.    Decrement Update Counter
29.  else
30.    Last Result = Result
31.  EndIf
32. End
    
```

### 3.8 EAE main routine

The EAE algorithm is controlled by the following set of parameters:

1. Size of the population. It is generally accepted that the larger the size of the population, the better the performance of the evolutionary algorithm. On the other hand, computational effort is directly proportional to size, and therefore, a compromise must be reached when selecting the size.

2. Crossover fraction—the fraction of the population that passes through the crossover operator. The rest of the population undergoes mutation.
3. Elite count—the number of best individuals unconditionally promoted to the offspring population.
4. Stopping criteria—the number of generations (iterations).

Pseudo code for the proposed algorithm is given in Listing 5.

The routine is launched when a new data batch is read. In the beginning, the batch is split into pool and ensemble learning sets, and a validation set (line 9), and the population is prepared. The population (line 10) is then repeatedly processed in the main loop of the routine (lines 11–22).

In each generation (lines 11–21), the algorithm rates the individuals (line 12). The results form the basis for selecting the elite and parents, which undergo mutation or crossover (lines 13–16). The offspring population is then formed from the elite, mutated, and crossed individuals and the system checks whether overtraining has taken place. In the last repetition of the loop, the drift detector is launched (lines 19–21). If drift is detected, the number of generations is doubled in order to continue adaptation, because we wish to provide the opportunity of finding a solution by means of the newly added classifiers.

```

1. Program EAE
2. Input
3. Data Chunk
4. Previous Population
5. Output
6. Winner
7. Population
8. Begin
9. Split Data Chunk into
   Pool Training Set
   Ensemble Training Set
   Validating Set
10. Init population
11. Repeat Generation until Max Generation
12. Evaluate population on Ensemble Training Set;
13. Select Elite
14. Select Parents for mutation and crossover
15. Mutate
16. Crossover
17. Create child population
18. Detect Overtraining
19. IF (Last Generation)
20. Detect Drift
21. EndIf
22. EndRepeat
23. Set Winner to Best from Population;
24. End

```

## 4 Experiments

To evaluate the performance of the proposed algorithm, a series of experiments was carried out. The main objectives of the tests were as follows:

1. Experiment 1—assessment of whether the EAE can effectively adapt the ensemble to a changing environment while batch processing the data stream.

2. Experiment 2—estimation of the susceptibility of the EAE to overtraining and its ability to counteract this effect.
3. Experiment 3—assessment of how the pool size affects learning time.
4. Experiment 4—comparison of the performance of the EAE and selected competing methods.

### 4.1 Experimental setup

An experimental framework was implemented using Matlab R2010 and its OPTIMTOOL toolbox, including the genetic algorithm (GA) framework. The PRTools toolbox [60] was used for modeling the datasets and elementary classifiers.

To control the number of contexts and drift as well as its extent, five datasets without concept drift were downloaded from the UCI Machine Learning Repository [61]. For these datasets, artificial concept drift was generated. The drift generation routine is presented in the next subsection. The UCI datasets were used for Experiments 1, 2, and 3. To ensure reliable comparison analysis in Experiment 4, two datasets with real concept drift were used: the Electricity Demand dataset [62] and the SPAM dataset [55] from the Spam Assassin Collection (the Apache SpamAssassin Project—<http://www.spamassassin.apache.org/>). Details of the datasets are presented in Table 1.

In Experiments 1, 2, and 3, the  $k$ -nearest neighbor classifier model was used to create the pool of elementary classifiers. The number of nearest neighbors was automatically set in each classifier according to the optimization routine using leave-one-out error. To reduce the possibility of drawing too optimistic or pessimistic conclusions owing to the randomness of the learning procedure, all experiments were repeated 50 times. All data included in the figures and tables denote average values. In the fourth experiment, a naive Bayes classifier was used to fill the pool.

### 4.2 Drift generator

Details of the generator are given below:

1. As each dataset consists of a limited number of instances, its size was extended. Copies of objects were created and extra noise was added to the attributes to avoid recurrence of the same pattern in the data stream.
2. Drift was simulated by rotating selected pairs of features in 2D space. The angle of rotation represents the extent of drift. The smaller the angle, the lesser the drift. In the case of an odd number of features, one feature (remaining after selection of all pairs) was left unchanged. Random selection of the pairs was carried out separately for each drift generation.



**Table 1** Datasets used in the experiments

Name	Liver disorders	Auto MPG	Pima Indians Diabetes	Breast cancer	Biomed	Electricity demand	SPAM
Instances	345	398	768	699	699	45,312	699
Classes	2	2	2	2	2	2	2
Attributes	6	6	8	9	5	6	9,324
Concept drift	Synthetic	Synthetic	Synthetic	Synthetic	Synthetic	Real	Real

3. Each subset of instances representing a particular context was split into the desired number of data chunks, which were then sorted and merged into a single data stream.

Figure 1 illustrates the results of drift generation for the banana dataset. Four contexts were generated with a varying degree of drift (angles are given in radians): Context 1 (original), Context 2 (angle 0.5), Context 3 (angle 1.0), and Context 4 (angle 2.0).

### 4.3 Experimental results

#### 4.3.1 Experiment 1: EAE performance and stability of learning process

The objective of this experiment was to assess whether the EAE can effectively adapt the ensemble to a changing environment while batch processing a data stream.

Two datasets were used in the experiment: liver and auto MPG. To analyze the response of the EAE to context

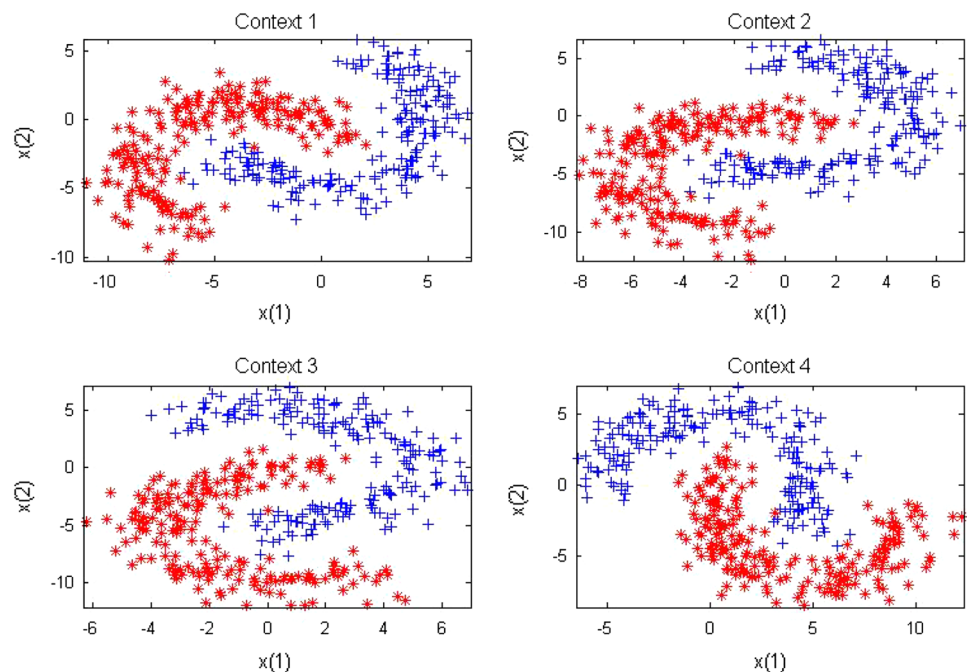
drift, three series of tests were conducted with a varying number of contexts and degree of drift (see Table 2).

For comparative analysis, apart from the EAE, two other classifier models were also tested.

1. Last added (LA)—a single elementary classifier is created and trained on the most recent data chunk.
2. Weighted voting ensemble (WVE)—a classifier consisting of a voting committee of elementary classifiers. When a new data chunk arrives, a new classifier is created and its accuracy is compared with all the classifiers in the committee. If the new classifier outperforms the worst one in the committee, the latter is removed and the new classifier takes its place. The weights of the classifiers are dynamically adjusted to the current context in proportion to their accuracy.

Results of the experiment are presented in Figs. 2, 3, 4, 5, 6 and 7. The three labels on the  $x$ -axis denote (from top to bottom): chunk number, context number, and occurrence number.

**Fig. 1** Scatter plots of example contexts created for the banana dataset by the drift generator



**Table 2** Drift details for Experiment 1

	Contexts	Drift extent (rotation angle)	Chunks per context	Ensemble size	Chunk size (instances)
Series 1	2	1.57	5	5	300
Series 2	4	0.78			
Series 3	6	0.52			

From an analysis of the results, we observed the following:

1. LA was the most stable and resistant to drift. This is not surprising since LA was trained on the most recent data. Nevertheless, fluctuations in the accuracy thereof can be observed. There are two possible reasons for this: (1) low representativeness of the data chunk, and (2) randomness of the data division into sets (for learning and testing). The drawback of LA is that in almost all cases it had the highest error rate. An obvious conclusion is that a correctly carried out classifier fusion can improve the accuracy of the system. Both the EAE and WVE proved their effectiveness and took advantage of the collective decision.
2. The emergence of new contexts caused a sudden increase in the error rate of both ensembles. With the processing of subsequent batches, the error was gradually reduced, owing to the collection of additional knowledge in the form of classifiers filling the ensemble. A marked difference in the reaction of the EAE and WVE to drift can be observed in the second half of the time series (when recurrence of contexts took place). The EAE remained stable, whereas the WVE still showed fluctuations at the time of drift. The reason is that the WVE can deal with only a limited

number of classifiers and rejected ones cannot be reused. On the other hand, the EAE can always select trained classifiers from the pool. This proves that the EAE can effectively deal with recurring contexts.

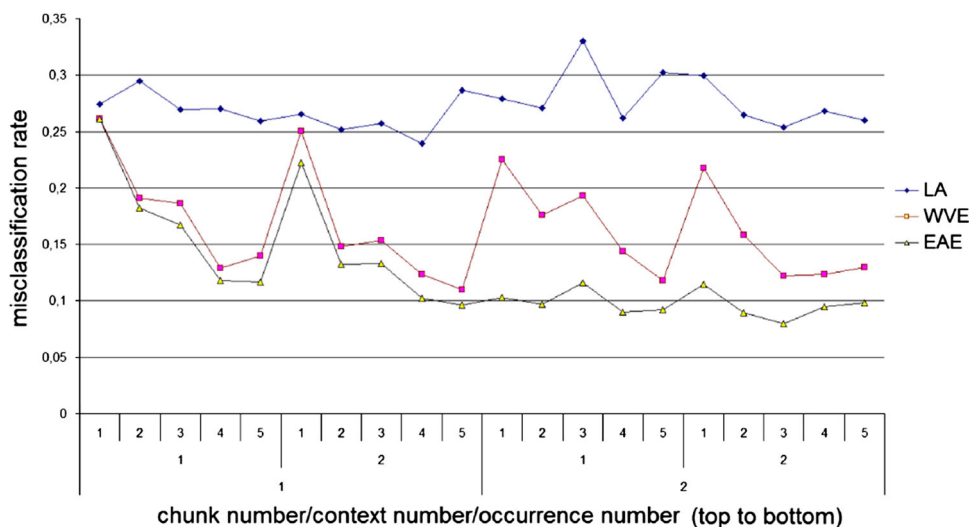
3. Of the tested classifiers, the EAE always achieved the smallest error rate. Adjustment of weights based on classifier accuracy, as occurs in the WVE, is easy to compute, but this does not guarantee an optimal solution. Therefore, subordination of the training process to minimize the ensemble error can result in even higher accuracy. The heuristic algorithm applied in the EAE proved its effectiveness in this regard.

4.3.2 Experiment 2: EAE ability to counteract overtraining

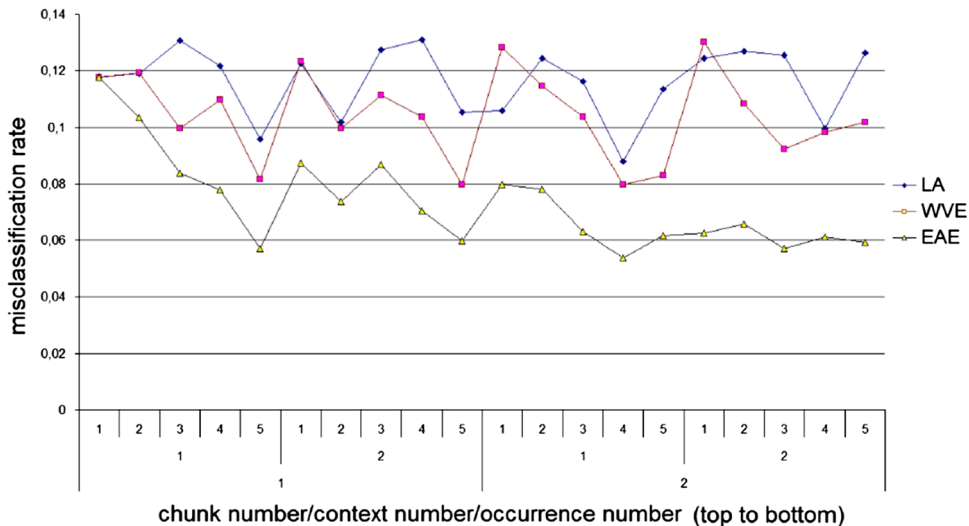
A common shortcoming of compound classifiers is their susceptibility to overtraining, which leads to the loss of generalization ability. The objective of the second experiment was to assess whether the EAE is able to effectively counteract this tendency.

The experiment was conducted on the breast cancer dataset. Three contexts with three chunks per context were generated. The EAE consisted of three classifiers. The following three EAE error rates were compared:

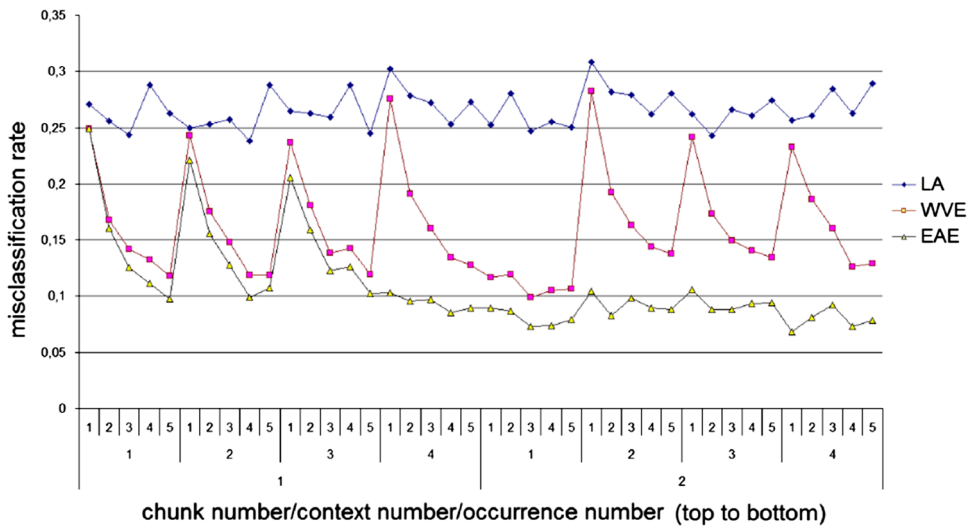
**Fig. 2** Misclassification rate for liver dataset (Series 1)



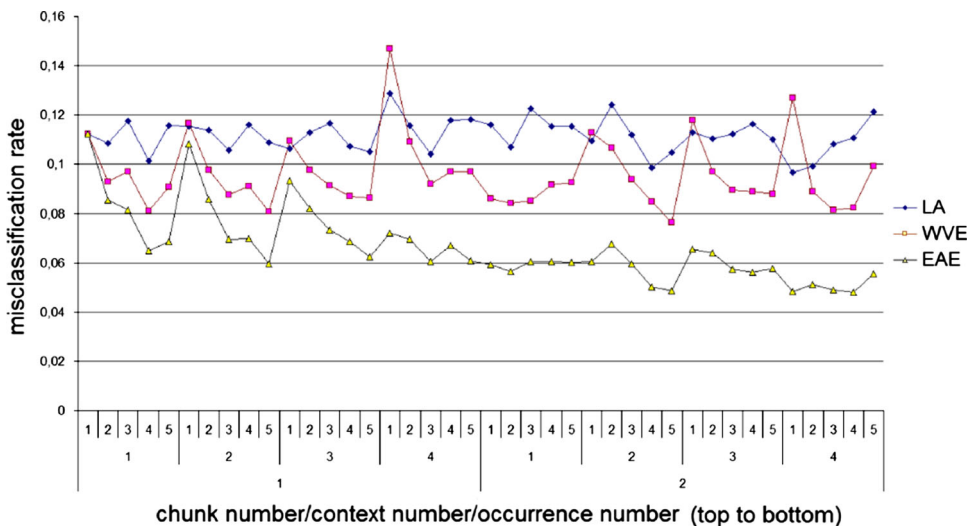
**Fig. 3** Misclassification rate for auto MPG dataset (Series 1)



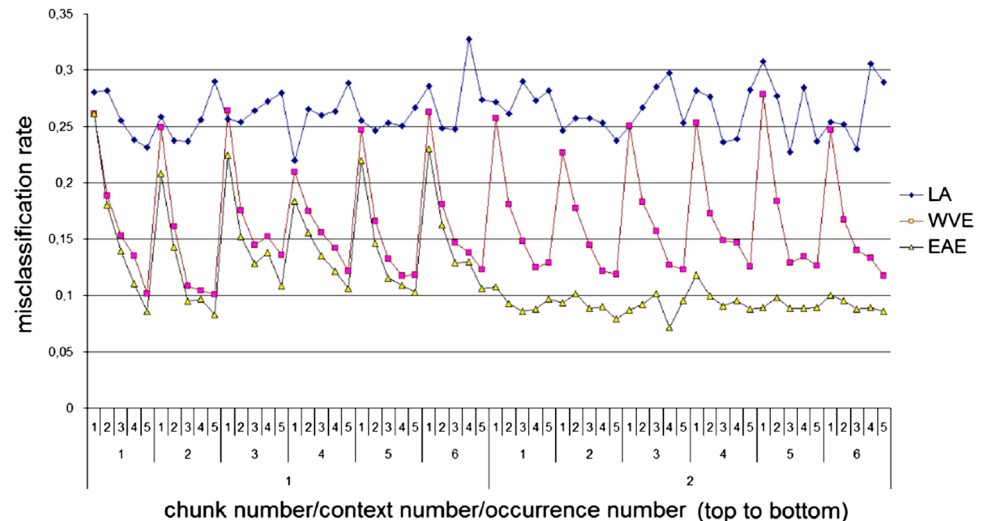
**Fig. 4** Misclassification rate for liver dataset (Series 2)



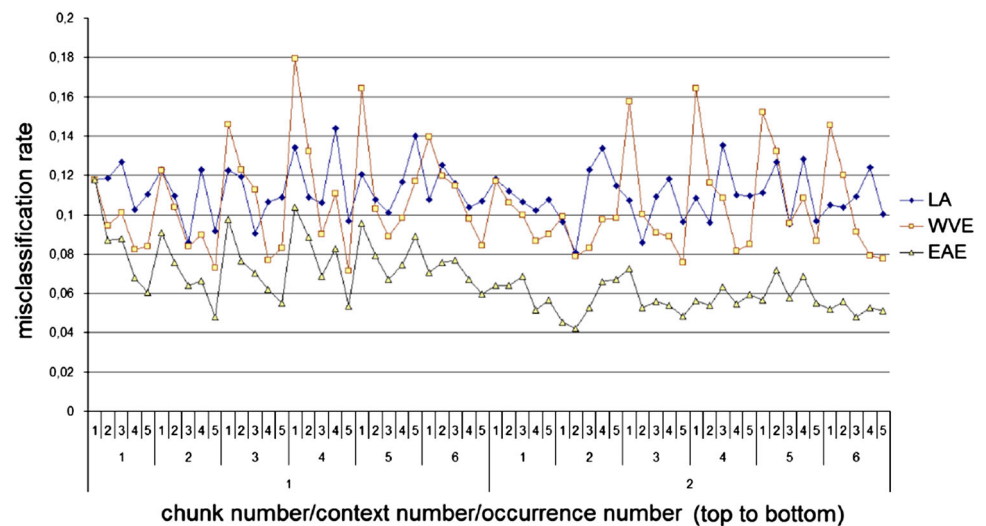
**Fig. 5** Misclassification rate for auto MPG dataset (Series 2)



**Fig. 6** Misclassification rate for liver dataset (Series 3)



**Fig. 7** Misclassification rate for auto MPG dataset (Series 3)



1. Error rate based on the learning set (LS);
2. Error rate based on the testing set with overtraining detector routine disabled (TS without detector);
3. Error rate based on the testing set with an active overtraining detector (TS with detector).

The results are depicted in Fig. 8.

The following observations were made:

1. The error rate based on the learning set is the smallest. This is not surprising; it is mentioned merely for reference.
2. The error rate based on the testing set without the overtraining detector is always slightly higher than that using an active detector. This proves that too high flexibility of the classifier may result in the loss of its generalization ability. Implementation of additional overtraining routines is a must in such systems. This proves that the detector implemented in the EAE is effective.

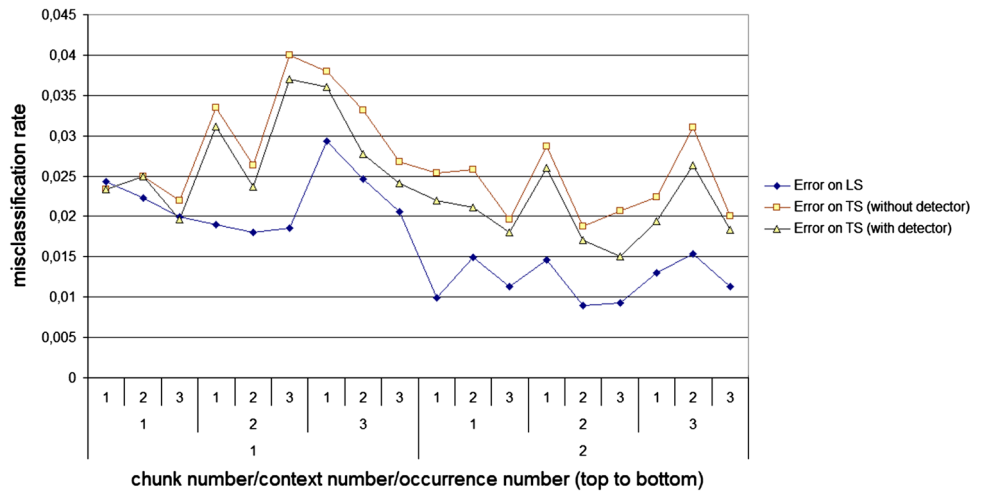
#### 4.3.3 Experiment 3: evaluation of EAE learning time

As previously mentioned, the EAE updates the pool when the drift detector identifies a significant change in the quality of the system. This situation occurs when new contexts appear in the stream. Therefore, together with the occurrence of the last context, the pool is filled with classifiers representing all possible contexts. It is expected that the processing time is directly proportional to the number of contexts, and increases with the emergence of new contexts until the last one is recognized. Thereafter, the time should remain constant.

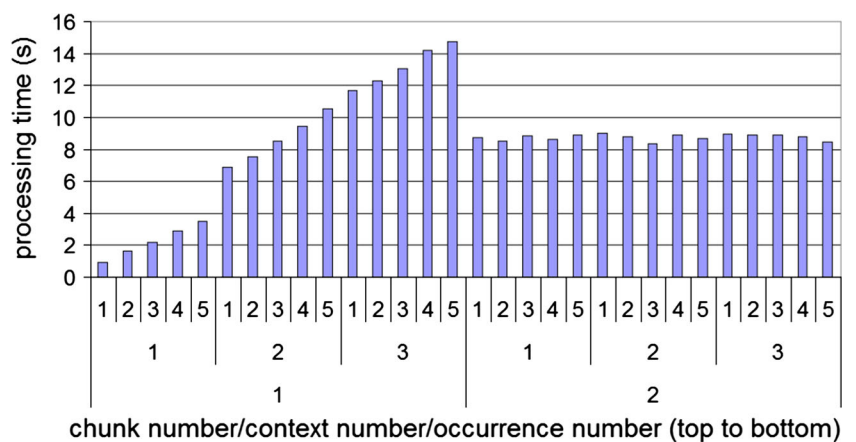
A series of experiments aimed at testing this hypothesis were carried out on the Pima Indians Diabetes dataset. Three contexts with five chunks per context were generated. The EAE algorithm consisted of an ensemble with five members. The results are illustrated in Figs. 9 and 10.

The following are our observations based on the results:

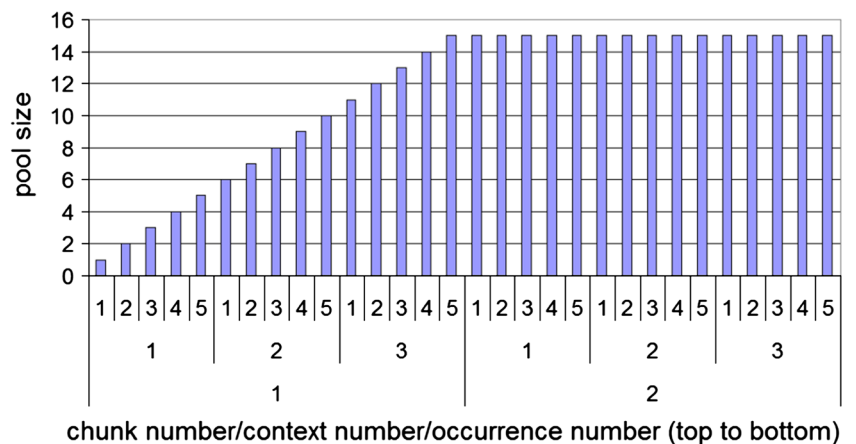
**Fig. 8** EAE misclassification rate for the Breast Cancer dataset based on learning set and testing set with and without overtraining detector



**Fig. 9** Processing time at subsequent learning steps (Pima Indians Diabetes dataset)



**Fig. 10** Size of pool at subsequent learning steps (Pima Indians Diabetes dataset)



1. With the arrival of new contexts in the stream, the drift detector initiates the procedure for updating the pool. This effect is visible on both graphs in the first half of the time series. The learning time and pool size increases linearly with an increase in the amount of processed data (number of new contexts).
2. Re-emergence of known contexts does not elicit a response from the detector. The size of the set and learning time remains almost unchanged. This effect is visible in the second half of the time series.
3. There is a 50 % reduction in learning time, which is visible in the second part of the time series results from the operation of the detector. As previously discussed, the detector doubles the number of operations when it detects a context (see the description of the drift detector in Sect. 3.7.2 for the details).

**Table 3** Results of comparative tests of EAE, DWM, AUE, and LA classifiers for two datasets with real concept drift

Dataset		AWE	AHT	LA	EAE	Statistical test (EAE significantly better than)
Electricity demand (90 chunks)	Average error	0,2673	0,2489	0,2624	0,2342	AWE AHT
	Error variance	0,0007	0,0009	0,0009	0,0060	LA
SPAM (23 chunks)	Average error	0,1967	0,2006	0,1783	0,1002	AWE
	Error variance	0,0031	0,0052	0,0030	0,0154	

It should be noted that there are also other factors affecting learning time, for example, number of attributes, number of classes, and types of elementary classifiers used in the ensemble. To understand how these factors affect processing time, we need to analyze how the EAE works. The algorithm is based on continuous processing of responses collected in a pool of classifiers.

The complexity of the recognition problem expressed in the number of attributes or the number of objects in the training set, directly affects processing time of the elementary classifiers. The EAE has access to these data only through the elementary classifiers. The EAE learning procedure repeatedly activates elementary classifiers in subsequent repetitions of the main loop. Thus, the EAE processing time is directly proportional to the number of classifiers, number of iterations, and size of the population. These relationships are linear. In summary, the EAE processing time is a linear function of the time required for learning and activation of elementary classifiers.

#### 4.3.4 Experiment 4: comparative analysis of the performance of EAE and three other methods

The purpose of the last experiment was to verify the effectiveness of the EAE compared with three other methods, namely, AWE [42], Adaptive Hoeffding Tree (AHT) with ADWIN [63], and LA, a single classifier trained when concept drift is detected. Tests were conducted using the two previously mentioned reference datasets with real concept drift (Electricity Demands and SPAM). It should be noted that the sets differ greatly in terms of the numbers of examples and attributes describing the objects.

To ensure an objective evaluation of the classifiers, an interleaved chunk test-then-train strategy was employed. In other words, each incoming chunk (except the first one) was first used for testing, as it contains instances never seen before and not previously used for training. Thereafter, the chunk was used for training.

The datasets were divided into packets each containing 400 examples. Thus, the Electricity set consisted of 89 packages and the SPAM set consisted of 22 packets.

A naive Bayes classifier was used as the elementary classifier model. The size of the voting committee in the EAE and AWE was 5.

A one-sided *t* test with a 0.05 confidence level was used to test the statistical significance of the results [64], which are presented in Table 3.

The following observations were made based on the results. For both datasets, the EAE had the smallest average misclassification rate of the tested classifiers. However, it should be noted that in all cases the EAE's error variance was much higher than that of the other methods. The high variance indicates that the EAE has a relatively smaller ability to maintain stability of the learning process. This can be due to randomness of the GA. The learning algorithm definitely needs further improvement to ensure higher stability. However, the statistical tests show that in some cases the difference between the EAE and other methods is statistically significant. This leads us to conclude that the EAE could be an interesting option for systems processing streams with drifting data.

## 5 Conclusions

In this paper we proposed a new recognition system designed to process data streams characterized by the presence of recurring contexts. The classifier model assumes that knowledge of all the contexts that have emerged in the data stream is permanently stored in a pool of elementary classifiers. This approach eliminates the need to re-create elementary classifiers for recurring contexts, which is a drawback of algorithms that eliminate outdated classifiers. In addition, it also allows for easy and quick adaptation of the system. These assumptions can be satisfied, if the applied learning algorithm is able to efficiently recognize the moment of drift and the emerging context.

A specifically designed evolutionary algorithm was also presented and implementation details of the genetic operators and learning procedures were discussed in detail. In addition, the effectiveness of the proposed algorithm was verified in an experiment that analyzed the system's response to changing contexts (Experiment 1). The results

show that the EAE reacts quickly to emerging changes, and re-emergence of familiar context does not cause even the slightest inaccuracy in the classification. Because the EAE model has great flexibility, which is characteristic of complex recognition systems, there are concerns that it could lose its generalization ability. To prevent this, an overtraining detector was implemented and its effectiveness was confirmed through a second experiment.

The paper also presented an estimation of computational complexity. Three main factors affect processing time: size of the classifier pool, size of the population, and number of generations. The relationship between these factors and time is linear, as confirmed in the third experiment. It is noteworthy that the results do not support the classification of the EAE as an online algorithm. This was not the intention of the authors, who assumed that it belongs to the class of batch algorithms.

The final experiment compared the effectiveness of the EAE with other algorithms designed to work in the presence of concept drift. Two benchmark datasets containing real concept drift were used for this purpose and the results were verified using statistical tests. We confirmed that the EAE compares favorably with other algorithms.

By summarizing the results of all the experiments, we can conclude that the EAE can be used successfully in applications where there is no need for online classifiers.

**Acknowledgments** This work was supported by The Polish National Science Centre under the grant N N519 576638 which is being realized in years 2010–2013.

## References

- Bifet A (2009) Adaptive learning and mining for data streams and frequent patterns. PhD thesis, Universitat Politècnica de Catalunya
- Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. *Mach Learn* 23(1): 69–101
- Zliobaite I (2010) Adaptive training set formation. PhD thesis, Vilnius University, Lithuania
- Hilas C (2009) Designing an expert system for fraud detection in private telecommunications networks. *Expert Syst Appl* 36(9): 11559–11569
- Black M, Hickey R (2002) Classification of customer call data in the presence of concept drift and noise. In: *Soft-Ware 2002, Proceedings of the 1st International Conference on Computing in an Imperfect World*, Springer, Berlin, pp 74–87
- Delany SJ, Cunningham P, Tsymbal A (2005) A comparison of ensemble and case-based maintenance techniques for handling concept drift in spam filtering. Technical Report TCD-CS-2005-19, Trinity College Dublin
- Cunningham P, Nowlan N (2003) A case-based approach to spam filtering that can track concept drift. In: *ICCB-2003 Workshop on Long-Lived CBR Systems*. Springer, London, pp 3–16
- Kelly MG, Hand DJ, Adams NM (1999) The impact of changing populations on classifier performance. In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining-KDD'99*, ACM Press, New York, USA, pp 367–371
- Markou M, Singh S (2003) Novelty detection: a review—part 1: statistical approaches. *Signal Process* 83:2481–2497
- Widmer G, Kubat M (1993) Effective learning in dynamic environments by explicit context tracking. *Proceedings of the 6th European Conference on Machine Learning ECML-1993*, Springer. Lecture Notes Comput Sci 667:227–243
- Kuncheva LI (2008) Classifier ensembles for detecting concept change in streaming data: overview and perspectives. In: *Proceedings of the 2nd Workshop SUEMA, ECAI 2008*, Patras, Greece, pp 5–9
- Mak L-O, Krause P (2006) Detection and management of concept drift, machine learning and cybernetics. In: *International Conference on*, 2006, pp 3486–3491
- Ouyang Z, Zhou M, Wang T, Wu Q (2009) Mining concept-drifting and noisy data streams using ensemble classifiers, artificial intelligence and computational intelligence, 2009. *AICI'09. International Conference*, vol 4, pp 360–364
- Tsymbal A (2004) The problem of concept drift: definitions and related work, Technical Report TCD-CS-2004-15. Department of Computer Science, Trinity College Dublin, Ireland
- Klinkenberg R, Renz I (1998) Adaptive information filtering: Learning in the presence of concept drifts. In: *Learning for text categorization*. AAAI Press, Marina del Rey, pp 33–40
- Klinkenberg R (2004) Learning drifting concepts: example selection vs. example weighting. *Intell data anal* 8:281–300
- Chen S, Wang H, Zhou S, Yu P (2008) Stop chasing trends: Discovering high order models in evolving data. In: *Proceedings of the 24th International Conference on Data Engineering*, 2008, pp 923–932
- Kuncheva LI (2004) Classifier ensembles for changing environments. In: *5th International Workshop on Multiple Classifier Systems, MCS 04, LNCS*, vol. 3077, Springer, Berlin, pp 1–15
- Kuncheva LI (2004) Combining pattern classifiers. *Methods and algorithms*. Wiley, New York
- Littlestone N, Warmuth MK (1994) The weighted majority algorithm. *Inf Comput* 108:212–261
- Schlimmer J, Granger R (1986) Incremental learning from noisy data. *Mach Learn* 1(3):317–354
- Aha DW, Kibler D, Albert MK (1991) Instance-based learning algorithms. *Mach Learn* 6(1):37–66
- Ouyang Z, Gao Y, Zhao Z, Wang T (2011) Study on the classification of data streams with concept drift. In: *2011 Eighth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, pp 1673–1677
- Domingos P, Hulten G (2003) A general framework for mining massive data streams. *J Comput Graph Stat* 12:945–949
- Polikar R, Udpa L, Udpa SS, Honavar V (2001) Learn++: An incremental learning algorithm for supervised neural networks. *IEEE Trans Syst Man Cybern Part C Appl Rev* 31(4): 497–508
- Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams. In: *7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, San Francisco, pp 97–106
- Bifet A, Holmes G, Pfahringer B, Read J, Kranen P, Kremer Hardy, Jansen H, Seidl T (2011) MOA: a real-time analytics open source framework. *ECML/PKDD* 3:617–620
- Bifet A, Holmes G, Pfahringer B, Kirkby R, Gavaldà R (2009) New ensemble methods for evolving data streams. In: *KDD'09 Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining*, ACM, pp 139–148
- Chu F, Zaniolo C (2007) Fast and light boosting for adaptive mining of data streams. In: Dai H, Srikant R, Zhang C (eds) *PAKDD*. Springer, Berlin, pp 282–292

30. Bifet A, Gavalda R (2006) Learning from time-changing data with adaptive windowing. Technical report, Universitat Politècnica de Catalunya, 2006. (<http://www.lsi.upc.edu/~abifet>)
31. Lazarescu M, Venkatesh S, Bui H (2003) Using multiple windows to track concept drift. Technical report. Faculty of Computer Science, Curtin University
32. Kurlj B, Woźniak M (2011) Learning Curve in Concept Drift While Using Active Learning Paradigm. In: Bouchachia A (ed) Adaptive and Intelligent Systems. Springer Berlin Heidelberg, pp 98–106
33. Koychev I (2000) Gradual forgetting for adaptation to concept drift. In: Proceedings of ECAI 2000 Workshop Current Issues in Spatio-Temporal Reasoning, pp 101–106
34. Stanley K (2003) Learning concept drift with a committee of decision trees. Technical Report UT-AI-TR-03-302, Computer Sciences Department, University of Texas
35. Tsymbal A, Pechenizkiy M, Cunningham P, Puuronen S (2008) Dynamic integration of classifiers for handling concept drift. *Inf Fusion* 1:56–68
36. Shipp CA, Kuncheva LI (2002) Relationships between combination methods and measures of diversity in combining classifiers. *Inf Fusion* 3(2):135–148
37. Littlestone N, Warmuth M (1994) The weighted majority algorithm. *Inf Comput* 108(2):212–261
38. Jacobs RA, Jordan MI, Nowlan SJ, Hinton GE (1991) Adaptive mixtures of local experts. *Neural Comput* 3:79–87
39. Nikunj CO (2000) Online ensemble learning. In: AAAI/IAAI. AAAI Press/The MIT Press, USA
40. Rodriguez J, Kuncheva L (2008) Combining online classification approaches for changing environments. In: SSPR/SPR, vol 5342 of LNCS, Springer, Berlin, pp 520–529
41. Street N, Kim Y (2001) A streaming ensemble algorithm (sea) for large scale classification. In: KDD'01 Proceedings of the 7th ACM SIGKDD international conference on knowledge discovery and data mining, ACM, pp 377–382
42. Wang H, Fan W, Yu PS, Han J (2003) Mining concept-drifting data streams using ensemble classifiers. In: Getoor L, Senator TE, Domingos P, Faloutsos C (eds) KDD, ACM, pp 226–235
43. Kolter J, Maloof M (2003) Dynamic weighted majority: a new ensemble method for tracking concept drift. In: ICDM, IEEE, pp 123–130
44. Zliobaite I (2009) Learning under concept drift: an overview. Technical report, Vilnius University, Faculty of Mathematics and Informatics
45. Gaber MM, Yu PS (2006) Classification of changes in evolving data streams using online clustering result deviation. In: Third international workshop on knowledge discovery in data streams, Pittsburgh, PA, USA
46. Salganicoff M (1993) Density-adaptive learning and forgetting. In: Proceedings of the 10th International Conference on Machine Learning, pp 276–283
47. Klinkenberg R, Joachims T (2000) Detecting concept drift with support vector machines. In: Proceedings of the 17th international conference on machine learning (ICML), San Francisco, CA, USA, pp 487–494
48. Baena-Garcia M, Campo-Avila J, Fidalgo R, Bifet A, Gavalda R, Morales-Bueno R (2006) Early drift detection method. In: 4th international workshop on knowledge discovery from data streams, pp 77–86
49. Kurlj B, Wozniak M (2012) Active learning approach to concept drift problem. *Log J IGPL* 20:550–559
50. Ramamurthy S, Bhatnagar R (2007) Tracking recurrent concept drift in streaming data using ensemble classifiers. In: Proceedings of the 6th international conference on machine learning and applications, pp 404–409
51. Sasthakar R, Raj B (2007) Tracking recurrent concept drift in streaming data using ensemble classifiers. In: ICMLA'07 Proceedings of the 6th international conference on machine learning and applications, IEEE Computer Society, Washington, DC, pp 404–409
52. Turney P (1993) Exploiting context when learning to classify. In: Proceedings of the European conference on machine learning (ECML-93), pp 402–407
53. Widmer G (1997) Tracking context changes through meta-learning. *Mach Learn* 27(3):59–286
54. Gomes JB, Ruiz EM, Sousa PAC (2011) Learning recurring concepts from data streams with a context-aware ensemble. In: Proceedings of SAC, 2011, pp 994–999
55. Katakis I, Tsoumakas G, Vlahavas I (2010) Tracking recurring contexts using ensemble classifiers: an application to email filtering. *Knowl Inf Syst* 22(3):371–391
56. Hosseini MJ, Ahmadi Z, Beigy H (2011) Pool and accuracy based stream classification: a new ensemble algorithm on data stream classification using recurring concepts detection. In: 2011 IEEE 11th International Conference on Data Mining Workshops, pp 588–595
57. Bäck T, Fogel D, Michalewicz Z (1997) Handbook of evolutionary computation. Oxford University Press, Oxford
58. Jackowski K, Wozniak M (2010) Method of classifier selection using the genetic approach. *Expert Syst* 27(2):114–128
59. Alpaydin E (2004) Introduction to machine learning. The MIT Press, Cambridge
60. Duin RPW, Juszczak P, Paclik P, Pekalska E, de Ridder D, Tax DMJ (2004) PRTools4, a Matlab toolbox for pattern recognition, Delft University of Technology, The Netherlands
61. UCI Machine Learning Repository. <http://www.archive.ics.uci.edu/ml/>
62. Harries M (1999) Splice-2 comparative evaluation: electricity pricing, Technical report, The University of South Wales, Australia
63. Bifet A, Gavalda R (2009) Adaptive learning from evolving data streams in IDA 2009
64. Dietterich TG (1998) Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Comput* 10(7): 1895–1924