Artificial Intelligence 230 (2016) 108–133

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

A concept drift-tolerant case-base editing technique

Ning Lu[a], Jie Lu[a,*], Guangquan Zhang[a], Ramon Lopez de Mantaras[b]

[a] Faculty of Engineering and Information Technology, University of Technology Sydney, PO Box 123, Broadway, NSW 2007, Australia
[b] Artificial Intelligence Research Institute (IIIA-CSIC), Campus UAB, Bellaterra, Spain

A R T I C L E I N F O

Article history:
Received 18 June 2013
Received in revised form 18 August 2015
Accepted 21 September 2015
Available online 30 September 2015

Keywords:
Case-based reasoning
Case-base editing
Concept drift
Competence model

A B S T R A C T

The evolving nature and accumulating volume of real-world data inevitably give rise to the so-called "concept drift" issue, causing many deployed Case-Based Reasoning (CBR) systems to require additional maintenance procedures. In Case-base Maintenance (CBM), case-base editing strategies to revise the case-base have proven to be effective instance selection approaches for handling concept drift. Motivated by current issues related to CBR techniques in handling concept drift, we present a two-stage case-base editing technique. In Stage 1, we propose a Noise-Enhanced Fast Context Switch (NEFCS) algorithm, which targets the removal of noise in a dynamic environment, and in Stage 2, we develop an innovative Stepwise Redundancy Removal (SRR) algorithm, which reduces the size of the case-base by eliminating redundancies while preserving the case-base coverage. Experimental evaluations on several public real-world datasets show that our case-base editing technique significantly improves accuracy compared to other case-base editing approaches on concept drift tasks, while preserving its effectiveness on static tasks.

1. Introduction

Case-based Reasoning (CBR) is a problem-solving strategy that uses prior experience to understand and solve new problems. Unlike other model-based learning methods, which store past experience as generalized rules and objects, CBR systems store past experience as individual problem solving episodes [1] and delay generalization until problem-solving time. Despite several reported advantages of CBR in the literature [2] (e.g., CBR performs well with disjointed concepts, or CBR, as a lazy learner, is easy to update), large-scale and long-term CBR systems suffer from effectiveness and competence issues [3], which has led to increased awareness of the importance of Case-base Maintenance (CBM).

Case-base Maintenance refers to the process of revising the contents of a CBR system, thereby improving the system's efficiency and competence [4]. While CBM extends beyond the case-base to include all knowledge containers, in this paper, CBM is restricted to maintenance of the case-base only. In CBM, methods of revising the case-base (also called case-base editing) involve reducing the size of a case-base or training set while endeavoring to preserve or even improve generalization accuracy [5]. Two distinct areas in case-base editing have been identified and investigated: 1) competence enhancement, which aims to remove noisy cases, thereby increasing classifier accuracy; 2) competence preservation, which corresponds to redundancy reduction, i.e., removing superfluous cases that do not contribute to classification competence. Although several hybrid methods exist that search for a small subset of the training set, and simultaneously achieve the elimination of noisy

* Corresponding author.
E-mail addresses: Ning.Lu@uts.edu.au (N. Lu), Jie.Lu@uts.edu.au (J. Lu), Guangquan.Zhang@uts.edu.au (G. Zhang), mantaras@iiia.csic.es (R. Lopez de Mantaras).

http://dx.doi.org/10.1016/j.artint.2015.09.009
0004-3702/© 2015 Elsevier B.V. All rights reserved.

and superfluous cases, competence enhancement and competence preservation methods can be combined to achieve the same objectives as hybrid methods [6].

The issue of concept drift refers to the change of distribution underlying the data [7,8]. More formally, the issue of concept drift can be framed as follows: If we denote the feature vector as x and the class label as y, then a data stream will be an infinite sequence of (x, y). If the concept drifts, it means that the distribution of $p(x, y)$ between the current data chunk and the yet-to-come data is changing. If we decompose $p(x, y)$ into the following two parts as $p(x, y) = p(x) \times p(y|x)$, we can say that there are two sources of concept drift: one is $p(x)$, which evolves with time t, and can also be written as $p(x|t)$, and the other is $p(y|x)$, the conditional probability of feature y given x [9]. Learning under concept drift is considered to be one of the most challenging problems in machine learning research [10]; it is consequently the subject of increased attention [11].

At present, approaches for handling concept drift can be generally divided into three categories [12]: 1) instance selection (window-based), where the key idea is to select the instances that are most relevant to the current concept. Many case-base editing strategies in CBR – including this research – that delete noisy, irrelevant or redundant cases are also a form of instance selection. Since this research focuses on case-based techniques in dynamic environments where concept drift may occur, we will give a more detailed review of CBM methods in Section 2; 2) instance weighting, where each instance is assigned a weight to represent the decreasing relevance of existing training examples. These instances can be weighted according to their "age", or their competence with regard to the current concept [13]. Research into this category [14,15] mainly focuses on exploring a proper weighting schema; 3) ensemble learning (learning with multiple models) is reported to be the most popular and successful approach for dealing with concept drift [16]. It utilizes multiple models by voting [17,18] or selecting the most relevant model to construct an effective predictive model [19]. Generally, there are two ensemble frameworks: 1) horizontal ensemble [20,21], which builds on a set of buffered data chunks; 2) vertical ensemble [9,22], which builds on the most recent data chunk only. More recently, an aggregate ensemble framework, which could be seen as a hybrid approach of the two, has been proposed [23]. There is also research that maintains ensembles with different diversity levels [24,25]. The key idea behind this kind of research is that "before the drift, ensembles with less diversity obtain lower test errors. On the other hand, it is a good strategy to maintain highly diverse ensembles to obtain lower test errors shortly after the drift independent of the type of drift" [26].

Although concept drift has been an active research area in machine learning, little effort has been made in respect of CBM-related research. In real world applications, the case-base accumulates with usage over time, and the case distribution as well as the decision concepts underlying the cases may be subject to continuous change. This phenomenon poses additional challenges for existing case-base editing methods that implicitly assume that existing cases are drawn from a fixed yet unknown distribution, i.e., stationary assumption [22]. When concept drift occurs, past cases may not reflect current concepts; as a consequence, current case-base editing methods may preserve harmful cases. In addition, when class boundaries shift or novel concepts emerge, new cases representing novel concepts are more likely to be treated as noise and removed by competence enhancement algorithms, because they conflict with past concepts. This may seriously delay or even prohibit a case-based learner from learning new concepts. Finally, redundancy reduction is particularly challenging in domains with evolving decision boundaries. Most current competence preservation methods preserve only cases that lie on the decision boundaries. We argue that this is too aggressive and is inappropriate, particularly in the concept drift environment, for the following reasons: First, it will destroy the original case distribution, thus affecting the result of any change detection method which directly compares the case distribution. Second, it makes a learner too sensitive to noise, i.e., incorrectly retaining noisy cases as novel concepts, at the center of class definitions, will dramatically affect classification boundaries. Last, because "boundary cases distinguish one concept from another, while typical cases characterize the concept they belong to [27]", preserving a certain number of typical cases may also help to improve a CBR system, e.g., by improving case explanation. Motivated by the above-mentioned issues, while recognizing that it is not possible to know in advance whether there will be concept drift, this research proposes a novel case-base editing method that addresses both competence enhancement and competence preservation, and works well in both static and changing environments.

In this paper, we present a new case-base editing technique which takes both competence enhancement and competence preservation into consideration. For competence enhancement, we develop a Noise-Enhanced Fast Context Switch (NEFCS) algorithm to prevent noise from being included during case retention and to speed the context switch process in the face of concept drift. By taking advantage of our previous research on concept drift detection [28], our NEFCS algorithm minimizes the risk of discarding novel concepts. For competence preservation, we invent a Stepwise Redundancy Removal (SRR) algorithm that uniformly removes superfluous cases without loss of case-base competence. Experimental evaluations based on public real-world datasets show that our case-base editing technique demonstrates significant improvements in time-varying tasks and exhibits good performance on static tasks compared with the most common case-base editing methods.

This paper is organized as follows. Section 2 reviews the established literature on case-base editing techniques, as well as the research in CBR for tackling concept drift. Section 3 presents our proposed two-stage case-base editing technique. Section 4 evaluates each proposed algorithm and the overall technique for handling concept drift, using real datasets of both static and concept drift tasks. Section 5 summarizes our conclusions.

2. Literature review

In this section, we first briefly explain some of the proposed competence models as a prerequisite, because many case-base editing solutions have taken a new direction in the sense that they are guided by explicit models of competence. We then review the established literature on case-base editing. Lastly, we describe the main developments in handling concept drift in CBR, and identify the limitations of the current research.

2.1. Competence models

Competence is a measurement of how well a CBR system fulfils its goals. As CBR is a problem-solving methodology, competence is usually taken to be the proportion of problems that it can solve successfully. In the CBR community, competence measurement has been given much attention and many competence models [5,29–31] have been proposed. These proposed models have now become essential tools for use in all stages of system development and are reported to be particularly important for CBM, where knowledge is added, deleted and modified to affect system adaptation and improvement [32].

The first study on competence measures was conducted by Smyth and Keane [29], in which the local competence of an individual case was characterized by its coverage and reachability. The coverage of a case is the set of target problems that this case can solve. The reachability of a target problem is the set of cases that can be used to provide a solution for the target problem. Since it is impossible to enumerate all possible future target problems, in practice the case-base itself is assumed to be a representative sample of the underlying distribution of target problems. As a result, the coverage of a case is estimated by the set of cases that can be solved by its retrieval and adaptation (Definition 1), and the reachability of a case is estimated by the set of cases that can bring about its solution (Definition 2).

Definition 1. (See [29].) For a case-base $CB = \{c_1, c_2, \cdots, c_n\}$, given a case $c \in CB$, $CoverageSet(c) = \{c' \in CB : Solves(c, c')\}$, where $Solves(c, c')$ means that c can be retrieved and adapted to solve c'.

Definition 2. (See [29].) $ReachabilitySet(c) = \{c' \in CB : Solves(c', c)\}$, where $Solves(c', c)$ means that c' can be retrieved and adapted to solve c.

Definition 3. (See [30].) For $c_1, c_2 \in CB$, $SharedCoverage(c_1, c_2)$ iff $[RelatedSet(c_1) \cap RelatedSet(c_2)] \neq \phi$, where $RelatedSet(c) = CoverageSet(c) \cup ReachabilitySet(c)$.

Smyth and Keane's competence models characterize what a case contributes to its local competence. However, arguing that it is these retrieved cases that should be blamed for misclassifications, Delany and Cunningham [5] extended Smyth and Keane's competence models by introducing a liability set (Definition 4) to measure the effectiveness or certainty of a case's competence contribution.

Definition 4. (See [5].) $LiabilitySet(c) = \{c' \in CB : Misclassifies(c', c)\}$, where $Misclassifies(c', c)$ means that case c contributes in some way to the incorrect classification of target case c'.

2.2. Case-base editing methods

This section presents a literature review on both competence enhancement and competence preservation aspects of case-base editing.

2.2.1. Competence enhancement

Competence enhancement aims to remove noisy instances to increase classifier accuracy. Some research that selects a set of appropriate cases and results in more suitable decision boundaries can also be viewed as a form of competence enhancement. One of the earliest noise removal techniques is Wilson's noise reduction technique [33]. In their proposed Edited Nearest Neighbor (ENN) algorithm [33], a case that is incorrectly classified by its nearest neighbors will be deleted from the original case-base as noise. To detect and delete small clusters of noisy cases, Tomek [34] made two modifications to ENN: the Repeated ENN (RENN) algorithm repeatedly applies ENN to the case-base until no more cases can be deleted, while the "all k-NN" algorithm increases the values of k during each cycle of RENN. These two modifications can improve classification accuracy; however, there is a risk of removing entire small clusters which represent genuine concepts [35]. Ferri and Vidal [36] integrated the ENN with cross-validation, which randomly splits the training set into n-folds and then iteratively removes cases from each fold that cannot be correctly classified by cases from other folds. According to the Structural Risk Minimization (SRM) principle, Karacali and Krim [37] proved that to construct an NN classifier on a reference set, one should aim for as few points as possible, while maintaining zero classification error on the training dataset. They developed a case-base editing method which simultaneously minimizes the noise incurred and the case-base redundancy. Inspired by the fact that most classification errors occur in the regions where the domains of the two classes are closest to each other, their method first considers the nearest pairs of points from different classes. Delany and Cunningham

[5] criticized methods based on Wilson's noise reduction technique [33] because a misclassified case may not necessarily be a noisy case but could be classified incorrectly due to the retrieved cases which contribute to its classification. They extended Smyth's competence model [30] to include the liability set and proposed a Blame-Based Noise Reduction (BBNR) algorithm which reviews all cases that have contributed to misclassifications and removes a case if its deletion results in no loss of coverage. Pan et al. [38] invented a Kernel Greedy Case Mining (KGCM) algorithm which extracts a given number of most representative cases. This greedy algorithm operates in an extracted feature space where the intra-class scatter around target classes is minimized while the inter-class scatter is maximized. The selected cases aim to maximize a defined global diversity and therefore are widespread within the feature space. Another extension of the competence model is the complexity model [39], in which a case with complexity greater than a given threshold means that its neighborhoods contain a majority of different classes, therefore the case is treated as noise. Rather than proposing a novel case-base editing method, Cummins and Bridge [40] developed a framework which combines existing noise reduction algorithms and redundancy reduction algorithms in sequence or through a set of voting rules. By trying all possible combinations, they were able to create several highly effective composite case-base editing methods.

2.2.2. Competence preservation

Competence preservation corresponds to redundancy reduction which aims to remove superfluous cases that do not contribute to classification competence. Hart's Condensed Nearest Neighbor (CNN) approach [41] is probably the earliest and best-known redundancy reduction technique which incrementally adds cases that cannot be correctly classified by the current edited case-base to an empty case-base. CNN makes multiple passes through the original case-base until no more additions can be made. Although intuitive, CNN has been criticized for being sensitive to the order of cases examined and to noise [5]. To overcome these problems, several modifications to CNN have been made, such as the Reduced Nearest Neighbor (RNN) [42], the Selective Nearest Neighbor (SNN) [43], the Modified CNN (MCNN) [44], the Generalized CNN (GCNN) [45], the Fast CNN (FCNN) [6] and the Improved CNN (ICNN) [46]. Other approaches to case editing build a competence model of the training data and use the competence properties of the cases to determine which cases to include in the edited set. Smyth and Keane [29] developed a footprint deletion policy which divides cases into four categories by their competence contributions and then removes cases from a case-base, guided by the categories, until a limit on the case-base size is reached. McKenna and Smyth [47] proposed a family of competence-guided methods based on different combinations of four features: an ordering policy, an addition rule, a deletion rule and a competence update policy. Although they empirically showed that their method preserves the competency of a CBR system and outperforms a number of previous deletion-based strategies, their deletion policy was criticized by Zhu and Yang [48] for not always being able to guarantee the competence preserving property. Instead, Zhu and Yang suggested a case addition policy which recursively selects a case that leads to maximum coverage benefit if added into an initially empty case-base until a limit on the case-base size is reached. Besides the advantage of being able to achieve the maximized case-base coverage, they also proved that their algorithm can place a lower bound on the coverage of the resulting case-base. Another competence-based case-base editing method is the Iterative Case Filtering Algorithm (ICF) [49], in which a case with a coverage set size smaller than its reachability set size is believed to be far from the class borders and is removed. Because noisy cases were protected from removal by their deletion rule, Brighton and Mellish adopted Wilson's deletion rule [33] ahead of their ICF algorithm. Arguing that existing competence preservation techniques were very aggressive in their pruning of cases, Delany and Cunningham [5] presented the Conservative Redundancy Reduction (CRR) algorithm, which adopts a similar principle to ICF [49] but removes redundant cases by repeatedly selecting the case with the smallest coverage set which has not yet been covered.

2.3. Case-based reasoning for handling concept drift

The first attempt to handle concept drift with a case-based technique is the Instance-Based learning Algorithm 3 (IB3) [50], which monitors each case's accuracy and retrieval frequency. IB3 prevents noisy cases by deferring the inclusion of an instance in the case-base until it is proven reliable by means of a confidence interval test, where the number of successful classifications is assumed to be binomially distributed. A case already included in the case-base is permanently removed if its accuracy is significantly less than its class's observed frequency. However, IB3 has been criticized for being suitable only for gradual concept drift, and for its costly adaptation process [8].

The Locally Weighted Forgetting (LWF) algorithm [51], which reduces the weights of the k-nearest neighbors (k-NN) of a new case and discards a case if its weight falls below a threshold θ, was believed to be one of the best adaptive learning algorithms in its time. However, the LWF algorithm has lower asymptotic classification in non-varying conditions. Klinkenberg [52] also showed in his experiments that instance weighting techniques tend to over-fit the data, thus performing worse than analogous instance selection techniques.

Salganicoff [53] introduced the Prediction Error Context Switching (PECS) algorithm in his research, which achieved good performance in both time-varying and static tasks. The PECS algorithm was similar to IB3 in the sense that it also tracks each case's accuracy and adopts the same confidence interval test to determine noise, although PECS does not normalize its lower bound confidence interval with respect to the overall observed frequency of the class. Despite this, the PECS algorithm differs from IB3 in several ways: First, any new observation is immediately included in the case-base. Second, the PECS algorithm calculates a case's accuracy based only on its latest l predictions. Third, rather than permanently deleting a case as noise, the PECS algorithm deactivates it and tracks its accuracy for reactivation purposes. Experiments show that the PECS algorithm improves robustness over IB3 on time-varying tasks at the cost of increased storage requirements. However, PECS was originally designed to improve performance in concept drift problems, where noise along with incoming observations was not considered. As a result, all noisy observations are retained first and can only be removed with a deferment. PECS has also been criticized for its unlimited memory assumption [54], by only disabling cases but not deleting them.

Delany et al. [55] suggested a two-level learning for handling concept drift. In level-1, they used a Competence-Based Editing (CBE) method [5], which is a hybrid of Blame Based Noise Removal (BBNR) and Conservative Redundancy Reduction (CRR) to manage the case-base periodically. Specifically, BBNR analyzes all cases that have contributed to misclassification and removes cases if their deletion results in no coverage loss; CRR repeatedly selects a case with the smallest coverage set that cannot be correctly solved. The authors compared their method with full case-base without management, as well as window-based updating, and demonstrated several improvements. However, a hybrid of two CBM methods designed for a static environment does not guarantee effective learning under concept drift. In the worst case, novel concepts can be consistently discarded, especially with gradual concept drift. In addition, the BBNR algorithm has difficulties in removing small groups of noisy cases. For example, two noisy cases that have each other in their coverage sets can correctly classify each other but can cause the misclassification of all other nearby cases; these small groups of noisy cases can never be removed by the BBNR, even though they continue to provide incorrect classification results. This phenomenon can be caused by outdated cases when concept drift occurs. Last but not least, BBNR neglects the competence model maintenance issue. Referring to an ill-matched competence model may lead to the mistaken preservation of noisy cases, i.e., if we have previously removed a noisy case c' that happens to be a member of the coverage set c that we are currently considering. If we do not keep the competence model up-to-date, c' will still be considered even though we know it is a noisy case. In level-2 learning, Delany et al. periodically reselected features to completely rebuild a CBR system. Level-2 learning [55] is beyond the scope of a case-base editing method, but it is a model rebuilding strategy that can be plugged into any instance selection technique. Later studies conducted by Delany and Bridge [56] suggested a feature-free distance measure which showed further improvement in accuracy on the same datasets in their experiments.

Beringer and Hüllermeier [54] presented an Instance-Based Learning on Data Streams (IBL-DS) algorithm that autonomously controls the composition and size of the case-base. This IBL-DS algorithm is based on three modification rules: 1) when the size of the case-base exceeds its limit, the oldest instance will be removed; 2) when concept drift is reported using Gama's detection method [57], a large number of instances will be deleted in a spatially uniform but temporally skewed way. The number of cases to be removed depends on the discrepancy between the minimum error rate and the error rate of the 20 most recent classifications; 3) for every retained new case whose class dominates in a test range, all neighbors in a candidate range that belong to a different class will be removed. Although Beringer and Hüllermeier's method is instructive in that it differentiates its instance selection strategy based on whether concept drift is reported, their instance deletion strategy might be difficult to implement in problem domains such as spam filtering where all features may be binary. Removing cases temporally may also result in loss of case-base competence, e.g., deleting rare but correct cases.

Indrė Žliobaitė [58] invented a family of training set formation methods named FISH (uniFied Instance Selection algoritHm). The FISH family dynamically selects a set of relevant instances as the training set for the current target instance. The key concept of the FISH family is to linearly combine distances in time and feature space for training set selection. It includes three modifications: FISH1, FISH2 and FISH3. In FISH1, the size of a training set is fixed and set in advance. FISH2, which is considered to be central to the family, varies the size by selecting a training set that achieves the best accuracy based on leave-one-out cross validation. In FISH3, the relative importance of time and space distance is determined through an additional loop of cross validation. Although the FISH family is reported to be capable of cooperating with any base classifier, the validation set is chosen based on nearest neighbors to the current target instance.

Other research discusses the cost and availability of new labeled training data and proposes methods for handling concept drift with limited labeled instances [59–61]. There is a clear difference between the research in this paper and these researches, however. The present research primarily focuses on a case-base editing (instance selection) technique that continuously improves or at least preserves the effectiveness and efficiency of a classifier in unknown situations (e.g., whether concept drift occurs, the type of concept drift), when it is provided with newly labeled cases. Experiments to investigate the effect of the availability and the number or proportion of newly labeled cases are not performed.

3. A new case-base editing approach

Traditional case-base editing methods either omit the evolving nature of many real world scenarios by implicitly making a stationary assumption [9] or are incapable of simultaneously addressing effectiveness and competence issues. Motivated by these two problems, we present a two-stage CBM process. In Stage 1 learning (enhancement), we propose a NEFCS algorithm, which targets the removal of noise in a dynamic environment. In Stage 2 learning (preservation), we propose a SRR algorithm, which removes redundant cases in a recursively uniform way while preserving case-base coverage. We evaluate both proposed case-base editing methods respectively using real-world data on static tasks and time-varying tasks, as well as evaluating the overall learning performance.

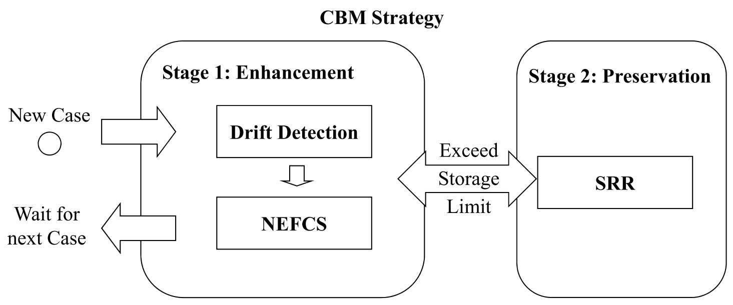

Fig. 1. A two-stage CBM process.

3.1. Problem description

Our technique is related to both CBM and data stream classification under concept drift. We assume there is a case-based system monitoring a live stream of new instances, where new labeled instances can be retained for future reasoning. Since concept drift may occur and noise may persist, we want to endow this case-based system with the ability to learn and refine accordingly, while restricting its case-base size rather than allowing it to expand limitlessly. Our solution consists of three modules: 1) competence-based drift detection [28], which compares the case distribution between two sliding windows of recent cases. When concept drift is detected, it also identifies the competence area where the distribution changes most significantly; 2) Noise-Enhanced Fast Context Switch (NEFCS), which enhances the system's learning capacity under concept drift; 3) Stepwise Redundancy Removal (SRR) method which controls the size of the case-base to tackle the performance issue.

The overall CBM process is shown in Fig. 1. For each new labeled instance, Drift Detection performs a competence-based concept drift detection [28]. NEFCS adapts the case-base on the basis of concept drift detection, thereby continuously improving the system's classification accuracy during its use. When the case-based system exceeds its storage limit, SRR will be triggered to shrink the case-base while preserving the case distribution as much as possible.

3.2. Competence-based drift detection

Our CBM strategy is motivated by the idea that when there is no concept drift, old cases can help to identify noise and improve accuracy; on the other hand, when concept drift is evident, new instances are more representative of the novel concept, while old cases that conflict may be obsolete. A change detection module is therefore adopted in our proposed CBM approach. This section reviews and extends the theoretical foundation of the competence-based drift detection method [28], which not only detects a concept drift but also identifies competence regions in which the concept drifts most severely.

Definition 5. (See [28].) Given a case base *CB*, and the case samples $S_i \subseteq CB$ $(i = 1, 2)$. For any *Related Set* $r \in \Re^{CB}(S_i)$, denote $\Re = \{r\}$, we define the *density* of r with regard to S_i as

$$w_i^*(r) = \frac{1}{|S_i|} \times \sum_{c \in S_i} \frac{|\Re \cap \Re^{CB}(c)|}{|\Re^{CB}(c)|}$$

where $\Re^{CB}(S_i)$ is the set of all related sets which contain at least one case $c \in S_i$, i.e., *Related Closure* of S_i.

Definition 6. (See [28].) Given a case base *CB*, and the case sample sets $S_i \subseteq CB$ $(i = 1, 2)$, denote the power set of $\Re^{CB}(CB)$ as $\wp(\Re^{CB}(CB))$. Considering $\wp(\Re^{CB}(CB))$ as the measurable space $\mathcal{A}$, for $A \in \mathcal{A}$, we define the *competence-based empirical weight* of S_i with regard to A over *CB* as

$$S_i^{CB}(A) = \sum_{r \in A \cap \Re^{CB}(S_i)} w_i^*(r)$$

In essence, the competence-based drift detection method partitions the problem space into a group of overlapping *related sets*, and then estimates the empirical probability over a competence area represented by A – a sub-set of $\Re^{CB}(CB)$ through *competence-based empirical weight*. The higher the weight is, the larger is the proportion of cases in S that support the selected competence area $-A$.

Definition 7. For two case sample sets $S_1, S_2 \subseteq CB$ and a *Related Set* $r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2)$, letting $\Re = \{r\}$, we define the *RelatedSet-based empirical distance* of r between S_1 and S_2 as

$$d_r^{CB}(S_1, S_2) = \left| S_1^{CB}(\Re) - S_2^{CB}(\Re) \right|$$

Theorem 1. *With the definition of the RelatedSet-based empirical distance, the competence-based empirical distance between S_1 and S_2, $d^{CB}(S_1, S_2) = \sup_{A\in\mathcal{A}} |S_1^{CB}(A) - S_2^{CB}(A)|$ can be computed as*

$$d^{CB}(S_1, S_2) = \sum_{r\in P} d_r^{CB}(S_1, S_2) \quad \text{or} \quad d^{CB}(S_1, S_2) = \sum_{r\in Q} d_r^{CB}(S_1, S_2),$$

where

$$\begin{aligned}
&P = \left\{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2) : S_1^{CB}(\Re) - S_2^{CB}(\Re) > 0\right\};\\
&Q = \left\{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2) : S_1^{CB}(\Re) - S_2^{CB}(\Re) < 0\right\};\\
&\sum_{r\in P} d_r^{CB}(S_1, S_2) = \sum_{r\in Q} d_r^{CB}(S_1, S_2).
\end{aligned}$$

Proof. For each $A \in \mathcal{A}$, we have

$$\begin{aligned}
S_1^{CB}(A) - S_2^{CB}(A) &= \sum_{r\in A\cap\Re^{CB}(S_1)} w_1^*(r) - \sum_{r\in A\cap\Re^{CB}(S_2)} w_2^*(r)\\
&= \sum_{r\in A\cap\Re^{CB}(S_1)} \sum_{r\in\Re\cap\Re^{CB}(S_1)} w_1^*(r) - \sum_{r\in A\cap\Re^{CB}(S_2)} \sum_{r\in\Re\cap\Re^{CB}(S_2)} w_2^*(r)\\
&= \sum_{r\in A\cap\Re^{CB}(S_1)} S_1^{CB}(\Re) + 0 - \sum_{r\in A\cap\Re^{CB}(S_2)} S_2^{CB}(\Re) - 0\\
&= \sum_{r\in A\cap\Re^{CB}(S_1)} S_1^{CB}(\Re) + \sum_{r\in A\setminus\Re^{CB}(S_1)} S_1^{CB}(\Re) - \sum_{r\in A\cap\Re^{CB}(S_2)} S_2^{CB}(\Re) - \sum_{r\in A\setminus\Re^{CB}(S_2)} S_2^{CB}(\Re)\\
&= \sum_{r\in A} \left(S_1^{CB}(\Re) - S_2^{CB}(\Re)\right)
\end{aligned}$$

Thus, to obtain the supremum of $|S_1^{CB}(A) - S_2^{CB}(A)|$ with regard to A, we should let A be either

$$\begin{aligned}
&P = \left\{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2) : S_1^{CB}(\Re) - S_2^{CB}(\Re) > 0\right\} \quad \text{or}\\
&Q = \left\{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2) : S_1^{CB}(\Re) - S_2^{CB}(\Re) < 0\right\}.
\end{aligned}$$

Correspondingly, the *competence-based empirical distance* between S_1 and S_2 is

$$d^{CB}(S_1, S_2) = \sum_{r\in P} d_r^{CB}(S_1, S_2) \quad \text{or} \quad d^{CB}(S_1, S_2) = \sum_{r\in Q} d_r^{CB}(S_1, S_2).$$

On the other hand,

$$\begin{aligned}
&\sum_{r\in P} d_r^{CB}(S_1, S_2) - \sum_{r\in Q} d_r^{CB}(S_1, S_2)\\
&\quad = \sum_{r\in P} \left|S_1^{CB}(\Re) - S_2^{CB}(\Re)\right| - \sum_{r\in Q} \left|S_1^{CB}(\Re) - S_2^{CB}(\Re)\right|\\
&\quad = \sum_{r\in P} \left(S_1^{CB}(\Re) - S_2^{CB}(\Re)\right) + \sum_{r\in Q} \left(S_1^{CB}(\Re) - S_2^{CB}(\Re)\right)\\
&\quad = \sum_{r\in\Re^{CB}(S_1)\cup\Re^{CB}(S_2)} \left(S_1^{CB}(\Re) - S_2^{CB}(\Re)\right)\\
&\quad = \sum_{r\in\Re^{CB}(S_1)} S_1^{CB}(\Re) + \sum_{r\in\Re^{CB}(S_2)\setminus\Re^{CB}(S_1)} S_1^{CB}(\Re) - \sum_{r\in\Re^{CB}(S_1)\setminus\Re^{CB}(S_2)} S_2^{CB}(\Re) - \sum_{r\in\Re^{CB}(S_2)} S_2^{CB}(\Re)\\
&\quad = 1 + 0 - 0 - 1 = 0.
\end{aligned}$$

Thus $\sum_{r\in P} d_r^{CB}(S_1, S_2) = \sum_{r\in Q} d_r^{CB}(S_1, S_2)$. □

Compared with the *competence-based empirical distance* [28], which defines the distance between two sample sets, the *Relatedset-based empirical distance* defines the distance between two sample sets on a particular relatedset. Theorem 1 proves that the *competence-based empirical distance* is the sum of the *relatedSet-based empirical distance* over all partitions (*related sets*) that are in favor of either S_1 or S_2 (with higher weight).

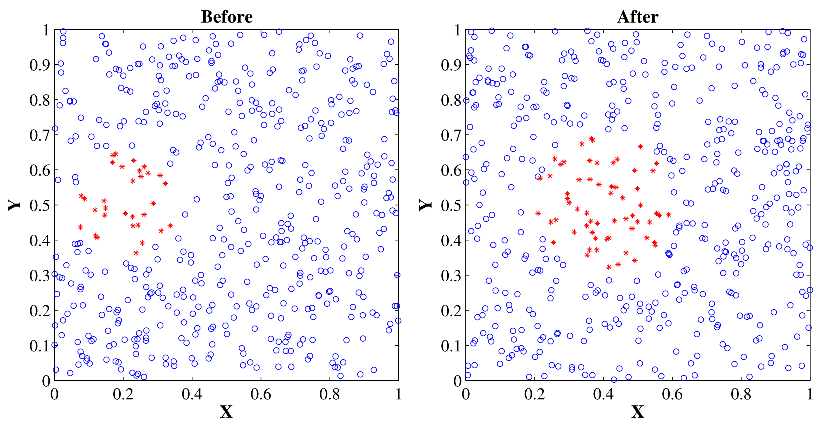

Fig. 2. Distributions of the moving circles before and after concept drift.

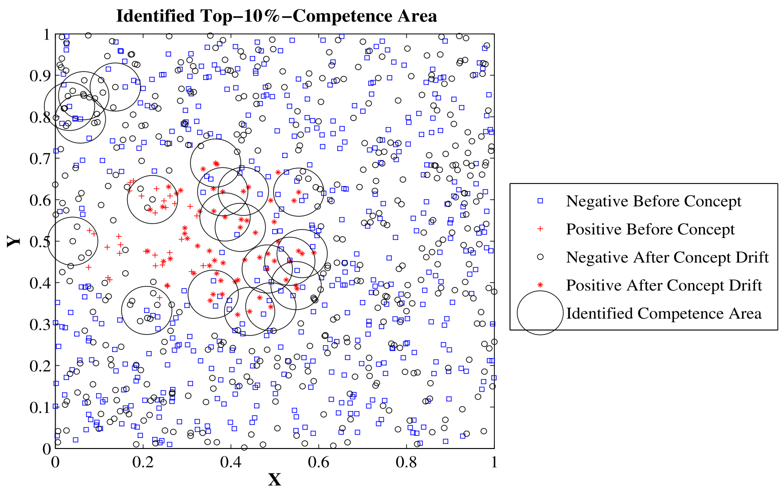

Fig. 3. Identified top-10%-competence area for circles.

Definition 8. If a set of *related sets* $\Re_p \subseteq \{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2) : d_r^{CB}(S_1, S_2) > 0\}$ meets the following three conditions

$$(1) \quad d_r^{CB}(S_1, S_2) \geq d_{r'}^{CB}(S_1, S_2) \quad \text{for } \forall r \in \Re_p \quad \text{and} \quad \forall r' \notin \Re_p$$

$$(2) \quad \sum_{r \in \Re_p} d_r^{CB}(S_1, S_2) \geq p \sum_{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2)} d_r^{CB}(S_1, S_2)$$

$$(3) \sum_{r \in \Re_p} d_r^{CB}(S_1, S_2) - \min_{r \in \Re_p}\left(d_r^{CB}(S_1, S_2)\right) < p \sum_{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2)} d_r^{CB}(S_1, S_2)$$

where $0 \leq p \leq 1$, then each element in $\Re_p$ is called *top-p-competence area.*

Since the largest *RelatedSet-based empirical distance* occurs in the top-p-competence areas, we treat these areas as the identified concept drift competence areas, where the most severe concept drift is believed to be occurring. To illustrate this, we visualize a simple moving circles dataset in Fig. 2, and depict the top-10%-competence areas identified in Fig. 3.

Clearly, we can see that the identified competence areas roughly highlight the exact problem space in which concept drift occurs. Also note that a small region has been selected in the top left corner, even though no real concept drift is taking place. This region has been identified because a significant increase in negative class has been detected, i.e., virtual concept drift. This phenomenon can be caused by bias in sampling. The results of both detection and description can be greatly improved if more representative samples are used for change detection. In addition, it worth pointing out that the smaller the p-value, the more confidence there will be in the identified areas, although fewer competence areas will be picked. Finally, in practice, areas undergoing real concept drift can be identified by translating the competence space back to the feature space and picking feature space where there is an emerging trend of positives in one sample set and an emerging trend of negatives in the other sample set.

Although any change detection technique can be adopted to accomplish this task, we choose the competence-based drift detection method [28], which uses the competence model as a space partition technique and compares the empirical distribution of two case windows. The main reason for adopting the competence-based change detection method is that it cannot only warn of possible concept drift but it can also highlight a small region of the problem space in terms of case-base competence, called the identified competence area (Definition 8), which is most likely to be affected by concept drift. This enhances one of the strengths of CBR for dealing with disjointed concepts, and also makes CBR more appropriate for dealing with local concept drift.

3.3. Noise-enhanced fast context switching

Our NEFCS method takes the results of competence-based change detection as input (whether there is a concept drift, and the identified competence area in which concept drift is detected), and aims to continuously improve the learning capability. NEFCS consists of three main processes: 1) Modified blame-based noise reduction (M-BBNR), 2) Context switching, and 3) Update competence model.

3.3.1. Modified blame-based noise reduction

Modified blame-based noise reduction (M-BBNR) can prevent a novel concept from being removed as noise based on the result of whether and where there is concept drift, which solves the problem of distinguishing noise from novel concepts.

Definition 9. Case $c' \in CB$ can be solved by CB (denoted as $Solves(CB, c')$) if there is a case $c \in CB$, $c' \neq c$, so that $c' \in CoverageSet(c)$.

Definition 10 *(Conditional BBNR rule).* If concept drift has been detected, a new case c will be safely deleted according to the BBNR rule, i.e.

$$\big|LiabilitySet(c)\big| > 0 \text{ (Definition 4), and } Solves\big(CB - \{c\}, c'\big) \text{ holds for } \forall c' \in CoverageSet(c)$$

only when c lies out of any identified concept drift competence area (i.e. $\forall r \in \Re_p \; c \notin r$).

M-BBNR applies the *conditional BBNR rule* to examine every new case c and determine whether it is noise. If there is concept drift and c lies in the identified concept drift competence areas (Definition. 8), c will be not be deleted, because it may represent a novel concept. Otherwise, c will be removed from the case-base if it fulfills the BBNR rule [5]. This helps to differentiate between new instances and prevent the possible inclusion of noise. If c is removed, then the next process (Context switching) will be skipped, otherwise only existing cases which conflict with c will be checked (Definition 11), which accelerates the process of learning with new concepts.

Let us denote the set of new cases that are deleted according to the *Conditional BBNR rule* as $\aleph$.

Theorem 2. *If a concept drift has been detected, the size of $\aleph$ is non-increasing when increasing p value which is used to define the concept drift competence areas in Definition 8.*

In addition, let NCB be the set of new cases that fulfill the BBNR rule, then it is trivially true that $\aleph = NCB$ when $p = 0$.

Proof. Assume $\{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2) : d_r^{CB}(S_1, S_2) > 0\} = \{r_1, r_2, \ldots, r_n\}$, and sort its elements (related sets) in descending order as $r_{i_1}, r_{i_2}, \ldots, r_{i_n}$ according to their *RelatedSet-based empirical distances*. To discover all the *top-p-competence areas* that meet the three conditions shown in Definition 8, we successively put these sorted related sets into $\Re_p$ starting with the first one r_{i_1} until the sum of *RelatedSet-based empirical distances* of all elements in $\Re_p$ is greater than or equal to

$$p \sum_{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2)} d_r^{CB}(S_1, S_2).$$

Given $0 \leq p_1 < p_2 \leq 1$, we have

$$p_1 \sum_{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2)} d_r^{CB}(S_1, S_2) \leq p_2 \sum_{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2)} d_r^{CB}(S_1, S_2)$$

then clearly $\Re_{p_2} \supseteq \Re_{p_1}$, that is, for any related set $r \in \Re_{p_1}$, $r \in \Re_{p_2}$ as well. For each case $c \in NCB$, if it is in one identified concept drift competence area corresponding to p_1, that is, there is a certain $r \in \Re_{p_1}$ so that $c \in r$, then this case must be in one identified concept drift competence area corresponding to p_2 as well, since $r \in \Re_{p_2}$. This means that if one case does not fulfill the *Conditional BBNR rule* when $p = p_1$, then it must also not fulfill it when $p = p_2$, so the size of $\aleph$ is non-increasing when the p value is increased from p_1 to p_2.

In particular, when $p = 0$, $p \sum_{r \in \Re^{CB}(S_1) \cup \Re^{CB}(S_2)} d_r^{CB}(S_1, S_2) = 0$. Because Condition 3 of Definition 8 cannot be met in any way. Thus $\Re_p = \emptyset$ and none of the cases in NCB can lie in any area of $\Re_p$, so $\aleph = NCB$. □

```
Local Variables:
aL, the conflicting list
CB, the case-base
CSet(c) , the coverage set of c
|LSet(c)| , size of the liability set of c
1:  sorted aL in descending order of |LSet(c)|
2:  For each x in aL
3:     CB = CB - {x}
4:     For each y in CSet(x)
5:        If CB contains y
6:           If y cannot be solved by CB
7:              CB = CB + {x}
8:              break
9:           End-If
10:       End-If
11:    End-For
12: End-For
```

Algorithm 1. Blame-based noise reduction algorithm.

The *conditional BBNR rule* imposes an additional condition on BBNR rule [5], i.e., any new case c will only be considered as noise and removed by the BBNR rule if c lies outside an identified concept drift competence area (does not represent a new concept). It is one of the reported strengths of the competence-based change detection method that it is able to identify small competence regions where concepts drift most severely. When working with other change detection methods which have no capacity to locate concept drifts, all new cases will be exempt from removal, which is also the case when $p = 0$ in Definition 8.

Definition 11. Case $c' \in CB$ is defined as a conflicting case by a new case c which is not in $\aleph$, if $c \in LiabilitySet(c')$.

To prevent unnecessary cost, the M-BBNR investigates only cases that conflict with c (Definition 11), instead of checking the entire case-base as the BBNR does. This largely improves the efficiency of the competence enhancement stage, because these retrieved cases have already been obtained when trying to solve c, therefore no additional case retrieval is required. Moreover, any case of the later window will be excluded from this list when there is concept drift if it lies in the identified competence area. There are two main reasons to exclude these cases: 1) there is still a chance that c may be noise, and the most recent cases in the same area can help to alleviate the effect; 2) to prevent the deletion of other recently retained cases which represent novel concepts. Lastly, every conflicting case will be examined for noise using the BBNR rule and will be removed if it fulfills this criterion.

The BBNR algorithm is given in Algorithm 1. We have made a minor modification to the original BBNR algorithm [5] in line-5 to ensure that no previously removed cases will be taken into coverage consideration.

3.3.2. *Context switching*

As BBNR has difficulty in removing groups of obsolete cases, NEFCS also refers to the effectiveness information to deal with concept drift. To track changes in accuracy due to concept drift without experiencing bias from a large number of historical observations derived from another concept, a shift register, which stores the latest l retrievals, is kept for each case as a form of effectiveness information. As a result, the shift register that keeps the latest l predictions of each retrievable case should be updated for each observed new case c, where a retrievable element c' to c means $sim(c, c') \geq sim(c, c'_k)$, where c'_k is the kth nearest neighbor to c.

NEFCS adopts the same confidence interval test used in the IB3 algorithm [50] and the PECS algorithm [53] to switch cases. This yields a confidence interval as calculated by (1). If the upper bound on the accuracy of a case c falls below the inactivation threshold $p_{\max}$, it is deactivated from future reasoning. However, this same example may eventually be moved back to the case-base if its lower bound accuracy rises above the agreement acceptance probability $p_{\min}$, as may the case when concepts drift cyclically.

$$confidence\ interval = \frac{p_i + z^2/2n \pm z\sqrt{p_i(1-p_i)/n + z^2/4n^2}}{1 + z^2/n} \tag{1}$$

where p_i is the calculated accuracy of c_i and n is the number of classification attempts of c_i. z is the confidence interval coefficient (either tabulated or computed).

3.3.3. *Update competence model*

This process is purely to ensure that the M-BBNR is referencing the correct competence model, i.e., the competence model is updated for case-addition [62] and removed cases will be purged from competence models.

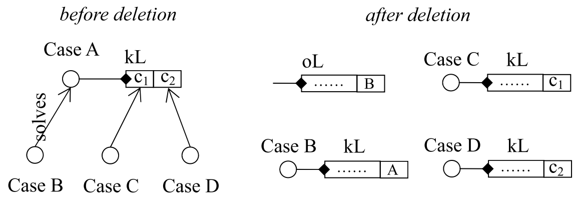

Fig. 4. An example of case deletion.

3.4. Stepwise redundancy removal

We adopt a similar schema to CNN [41] to guide our redundancy removal algorithm for competence preservation, that is, to obtain a sub-set of the original case-base that can successfully solve all removed cases. In this section, we will explain the SRR algorithm with the k-nearest neighbors (k-NN) rule, which makes it easier to apply to the nearest neighbor (NN) rule.

A major difference between SRR and other deletion-based competence preservation methods is that SRR recursively removes redundant cases in a competently uniform way, which we believe is more suitable for the concept drift problem. A similar approach used to handle the concept drift problem can be found in IBL-DS [54], which removes cases in a spatially uniform way. However, deleting cases uniformly in the feature space could be difficult to implement in high-dimensional data, such as spam filtering. In addition, IBL-DS suffers from the problem of losing case-base competence. Therefore, we suggest the uniform removal of redundant cases in the competence space. In addition, by taking advantage of the competence models, our method does not require multiple passes through the whole case-base, which saves the time incurred as a result of repeated case retrievals, as required in many competence preservation algorithms.

SRR starts by ordering the cases decreasingly by their reachability set, where a large reachability set indicates that a case is far from the class border [49] and works in a recursive manner. While running, SRR maintains three structures:

1. a preserved list – pL, which stores cases that cannot be removed;
2. a locked list – oL, which prevents a case from being deleted in the current round and will be cleared at the beginning of the next round;
3. a linked list for each case – kL, which links several previously removed cases to a case that solves them.

Stepwise Redundancy Removal works in a recursive manner. During each round, SRR continuously examines an unlocked case c with the largest reachability set until all cases are locked or preserved. A case c will be removed if c and all cases in the linked list (kL) of c can still be correctly solved without c. When c is removed, the closest $(k+1)/2$ cases that solve c are locked and linked to c. $(k+1)/2$ is chosen as the number of cases to lock because it is the minimum number of cases required to secure a correct classification of c, therefore no case-base competence will be lost by discarding c. In addition, each case c'_i in the kL of c will be added to the kL of the closest $(k+1)/2$ cases that solve c'_i. Fig. 4 depicts an example in which case A is removed by the nearest neighbor rule. First, cases in the remaining case-base that can solve case A as well as cases c_1, c_2 are retrieved. As a result, cases B, C, and D are retrieved. Such retrievals are instant because of the competence model, i.e., *ReachabilitySet*. Successful retrieval means that case A and all cases to which case A links can still be solved by the remaining case-base without case A; therefore A can be safely removed (no competence loss) and case B will be locked temporarily (added to the oL list). Then, any case in the linked list of case A will be added to the kL of the corresponding case that solves it. Failure to retrieve any case of B, C, or D means that case A cannot be removed; therefore case A will be preserved. SRR will move to the next round when there are no more unlocked cases, and will stop automatically when all cases have been preserved. However, SRR can be stopped at any time since it ensures that all removed cases can be correctly solved at any time (no coverage loss). SRR is presented in Algorithm 2.

Clearly, the preserved list – pL 'preserves' essential cases whose removal will cause competence loss, i.e., lead to failure to solve removed cases. In addition, pL helps to speed up the redundancy removal process, because cases in pL will be excluded from inspection in further rounds which avoids the cost of repeatedly examining the same essential case in every round. The locked list – oL – ensures that redundant cases will be deleted in a competently uniform way. For each case deleted, a list of cases that solve the deleted case will be locked to protect them from being removed during the current round. This prevents cases in a particular region from being cleaned quickly, even if they are at the center of class definitions. The linked list – kL – is the key to ensuring that there is no loss in competence for any deletion during multiple rounds. Since SRR removes cases gradually, having a subset of the case-base that correctly solves all cases at the beginning of each round cannot guarantee that this subset will solve all cases of the original case-base. For example, during the first round, case A is removed, and case B, which solves case A, is locked. During the next round, if case B is removed simply because case C, which solves case B, can be retrieved, there will be a risk of failing to solve case A. Therefore, for any case c, SRR maintains a kL to store cases for which c is retrieved to solve. To claim a case c can be removed, it must be possible to correctly solve c and all cases to which c links by the remaining case-base.

The differences between SRR and existing competence preservation methods are dramatic. First, existing methods take either a case addition or case deletion approach, while the approach in SRR is more like a hybrid of the two, because

```
Local Variables:
CB, the case-base
pL, the preserved list
oL, the locked list
kL(c), the linked list of c
RSet(c), the ReachabilitySet(c)
1:  sort CB in descending order of RSet(c) size
2:  set oL to empty
3:  While |CB-pL-oL| > 0
4:      set x to first case in CB-pL-oL
5:      If x and kL(x) can be solved by CB - {x}
6:          CB = CB - {x}
7:          For each y in kL(x)
8:              link y to the cases that solve y
9:          End-For
10:         link x to cases that solve x
12:         lock cases that solve x
13:     Else
14:         pL = pL + x
15:     End-If
16: End-While
17: If CB exceeds the size limits and |oL| > 0
18:     goto: 2
19: Else
20:     exit
21: End-If
```

Algorithm 2. Stepwise redundancy removal algorithm.

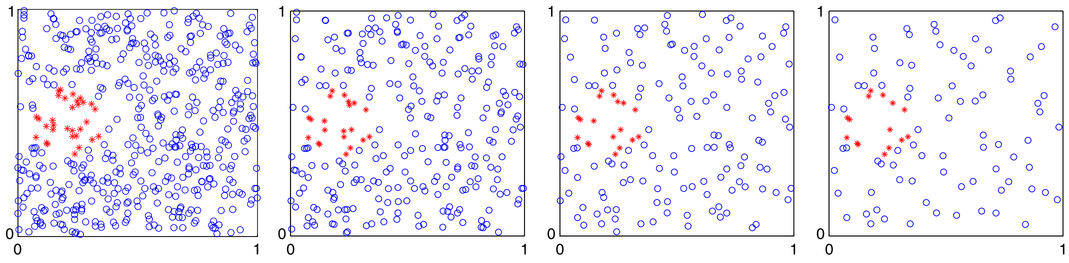

Fig. 5. Redundancy reduction process of SRR.

SRR can also “add” a case and prevent it from being deleted through the preserved list. Second, SRR uniformly removes redundancy through its locking mechanism. This not only facilitates case explanation [27] but also makes the learning of novel concepts smoother, without leaving a large blank in the feature space. Third, for each deletion, SRR guarantees that there will be no case coverage loss. As a result, SRR can be stopped at any time when the size limit is fulfilled, which gives the decision maker much more control over the size of the edited case-base.

We illustrate the redundancy reduction process of SRR during each loop with simple circle data [63] in Fig. 5, where *star* and *circle* indicate two different classes. We assume the 3-NN rule is used for classification.

We compare the deletion process with another iterative redundancy reduction algorithm, ICF [49], in Fig. 6 using the same dataset. Note that in Figs. 5 and 6, only three rounds were performed and visualized; more cases can be removed if the process is continued.

It can be seen that both methods remove redundancy gently and gradually. However, SRR operates in a uniform way, without leaving a large gap in the case of concept drift that may seriously affect the classification boundary.

To illustrate that SRR preserves its effectiveness for static tasks, we also compare the results with NUN-CNN, which sorts cases in ascending order of the nearest unlike neighbor (NUN) distance [64], ICF [49] and CRR [5]. Fig. 7 shows the results of minimized case-bases, i.e., no further cases can be deleted. It can be seen that when the case-base size is minimized for static tasks, SRR also focuses on defining decision boundaries because SRR shares the same principle as CNN [41]. However, the competently uniform approach taken by SRR exhibits a significant advantage (particularly where concept drift takes place) with better coverage while cases are being deleted. In addition, SRR can be stopped at any time, while the process of addition-based algorithms like NUN-CNN [64] and CRR [5] is completely uncontrollable. This means that SRR also exhibits the potential to intelligently adjust the case-base size in the face of concept drift.

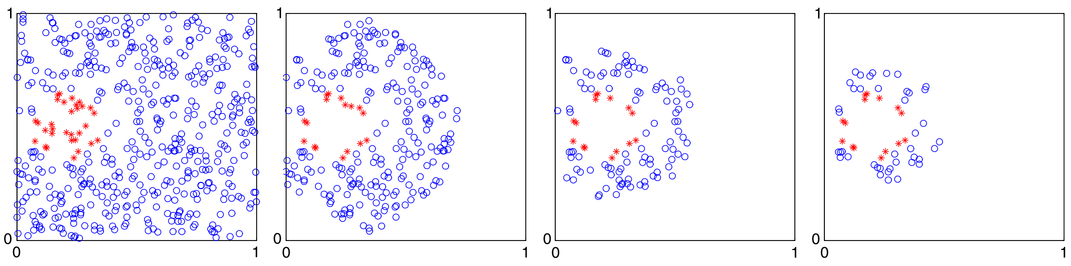

Fig. 6. Redundancy reduction process of ICF.

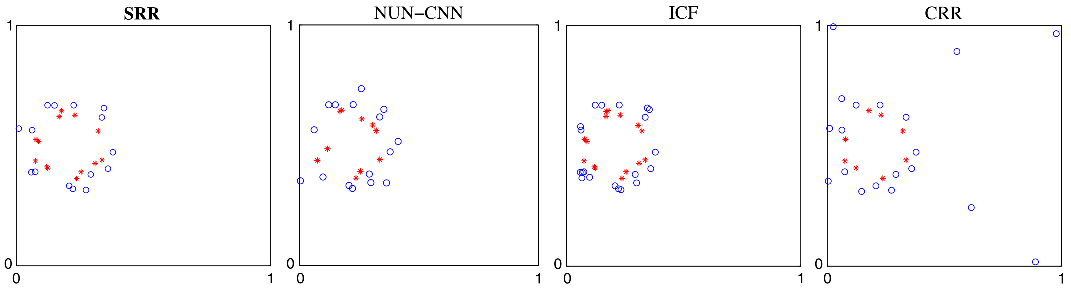

Fig. 7. Comparison of minimized results of SRR, NUN-CNN, ICF, CRR.

4. Experimental evaluation

This section provides a comprehensive experimental evaluation of the proposed two-stage CBM approach. In this section, we first evaluate the NEFCS algorithm and the SRR algorithm separately as independent CBM methods targeting competence enhancement and competence preservation respectively. We then evaluate the proposed two stage CBM approach for handling concept drift. All evaluations are based on real-world datasets.

4.1. Evaluating NEFCS

As it is not possible in practice to say whether concept drift will occur at some time in the future, a robust CBM strategy must be extensible to both static and changing environments. In this section, we evaluate the NEFCS as Stage 1 learning for both static tasks and time-varying tasks. We compare the results with other state-of-the-art competence enhancement approaches for both static and time-varying tasks.

Experiment 1 *(Spam filtering, static).* We choose to conduct studies on five public static spam filtering datasets, parts of which are used in the original work for testing the BBNR [5]. These datasets can be downloaded from http://www.comp.dit.ie/sjdelany/Dataset.htm. Each dataset consists of 1000 emails (500 spam/500 legitimate), extracted from an individual's email account. In these datasets, the feature is represented as F:XXXX where F is the feature type and XXXX is the actual text of the feature. Based on the unique combination of feature type and feature text, each email e_i is represented by a vector of features $e_i = \langle x_1, x_2, \cdots, x_n \rangle$, where each feature is binary. With all features weighted equally, the similarity between two emails $sim(e_x, e_y)$ is measured by the proportion of matched features, as shown in (2).

$$sim(e_x, e_y) = \frac{\sum_{i=1}^{n}(x_i \wedge y_i)}{\sum_{i=1}^{n}(x_i \vee y_i)} \tag{2}$$

where $e_x = \langle x_1, x_2, \cdots, x_n \rangle$, $e_y = \langle y_1, y_2, \cdots y_n \rangle$, x_i and y_i are binary features.

In the domain of spam filtering, the accuracy (or error rate) is not an adequate measurement of effectiveness since it does not give full transparency with regard to the number of False Positives (FPs) and False Negatives (FNs) that occur, where a False Positive (FP) is a legitimate email that is incorrectly classified as spam, which is believed to be significantly more serious than a False Negative (FN), i.e., a spam email that is incorrectly classified as a legitimate email. As a result, we use the False Positive Rate (FPR), which is calculated by $FPR = \frac{FP}{N} = \frac{FP}{(FP+TN)}$, the False Negative Rate (FNR), which is calculated by $FNR = \frac{FN}{P} = \frac{FN}{(FN+TP)}$, and the balanced average error (Err), which is defined as $Err = \frac{FPR+FNR}{2}$, to evaluate our algorithm.

For each dataset, we use 10-fold cross-validation. We divide the dataset into ten stratified divisions or folds. Each fold in turn is considered as a testing set, with the remaining nine folds acting as the training set. We sum the number of FPs and FNs for each testing fold and training set combination, and make an overall evaluation of each individual dataset.

We compare our NEFCS algorithm with other CBM approaches designed for time-varying tasks, including IB3 [50], PECS [53] and FISH2 [58], and for static tasks, including BBNR [5] and ENN [33], as well as Full Case-base (Full CB) with and without new case retention (update). Since we find that Full CB with update leads to slightly better results than Full CB without update, all the competence enhancement algorithms compared, with the exception of IB3 and NEFCS, immediately retain a new case after classification. Note that any new case is de-activated in IB3; in NEFCS, we distinguish a new case based on whether it might represent a novel concept and whether it fulfills the BBNR rule.

Competence enhancement algorithms designed for static tasks, such as BBNR and ENN, can be applied directly to each training set. However, for NEFCS, PECS and IB3, which remove noise during their classification processes, we perform leave-one-out-classification to edit the training set. That is, for each case in the training set, we first remove it from the training set or de-activated list and then try to classify it with each algorithm's successive learning mechanism. By doing this, we are able to initialize the de-activated list and classification accuracy for each case in advance.

To construct the competence model, a case c is considered to solve a case c' if $sim(c, c') > sim(c', c'_{nun})$, where c'_{nun} is the nearest unlike neighbor (NUN) of c'. This is the same method of constructing the competence model as described in ICF [49]. To compare case distribution, the window size selected is half the training set size, that is, 450, so that change detection can be started from the outset. We build a separate competence model using leave-one-out-classification with cases in the two windows for change detection purposes, where a case c is considered to solve a case c' if $sim(c, c') > sim(c', c'_{nun})$, where c'_{nun} is the NUN of c', and c is one of the retrieved cases for solving c'. There are three reasons for us to use a different way of constructing the competence model for change detection purposes: 1) While performing change detection, the competence model should reflect the current competence contribution of each case. While performing case-base editing, the competence model should reflect the maximum contribution of each case, since cases may be removed at any time. 2) We want to restrict the region of SharedCoverage (Definition 3) and focus each case's contribution on its closest and most related competence region. 3) It avoids re-searching the whole case-base to determine local competence. Using the actual retrieved cases can speed up the process of updating the competence model, thus facilitating online change detection.

As the problem space is very sparse, the confidence level chosen for confidence intervals is 80% for IB3, PECS and NEFCS. Other parameters required by PECS and NEFCS are set empirically, with $l = 10$, $p_{\max} = p_{\min} = 0.5$. The reason for this choice is that if a case provides more correct classifications than incorrect classifications, it will be added to the case-base, otherwise it will be de-activated. We did not test different parameter choices since NEFCS chooses the same parameters as PECS, and we believe this makes the comparison sufficiently fair.

We used the same classifier as Delany et al. [5] for all implemented techniques; that is, k-NN with $k = 3$ using unanimous voting, where an email is classified as spam if and only if all three retrieved emails are spam.

Finally, we force the FISH2 algorithm to start with a training set of 200 instances. Because the problem space is very sparse, starting with an extremely small training size $N = k$ makes the original FISH2 quite unsuitable for the spam filtering domain. Changing the starting size from $N = 3$ to $N = 200$ significantly improves its performance across all datasets; however a starting training size that is too large will contravene the original idea of FISH2, therefore 200 is chosen.

The results of comparing NEFCS with BBNR, PECS, FISH2, ENN, IB3, Full CB with and without update across the five datasets, and the overall average results of all datasets, are shown in Fig. 8 and Fig. 9 respectively. We also show percentage values for Err, FPR and FNR in both figures.

The results can be summarized as follows:

- No noise removal method consistently wins out. The behavior of different methods largely depends on the datasets.
- To compare multiple algorithms over multiple datasets, we run the Friedman test suggested by Demšar [65] which indicates a significant difference in Err at a confidence level of 95%. By setting NEFCS as the control classifier, we conduct the Bonferroni–Dunn test as a post-hoc test. As a result, NEFCS shows significant improvement over FISH2 ($z = 2.78$) and IB3 ($z = 2.9$) at a confidence level of 95%. Because FISH2 forces older instances to decay their retrieval over time (by increasing their distances to the current target instance in time space), we claim that FISH2 is not suitable for static tasks with high dimensionality.
- Because NEFCS, PECS and IB3 all adopt the same confidence interval test, we also conduct a Nemenyi post-hoc test to compare these three algorithms. As a result, NEFCS ($z = 2.69$, 95% confidence) and PECS ($z = 2.1$, 90% confidence) show significant improvement over IB3. This indicates that the post-addition strategy in IB3 may not be suitable for the spam filtering domain, where the problem space is very sparse. Another observation is that consistently retaining new cases can also assist effectiveness, although not much.
- It is worth noting that ENN has the worst FPR. We think this is due to legitimate emails being more likely to appear as several disjointed concepts, which causes a misclassified legitimate email to be removed by ENN but more likely to be preserved by the BBNR rule.
- NEFCS achieves similar results to BBNR on these static datasets. The differences on FPR, FNR and Err rate are not statistically significant.
- NEFCS has an Err rate that is at least as good as or better than PECS on four of the five datasets, and improves the average results on both the FPR and FNR. Although this improvement is not significant, it should nevertheless be safe to claim that using the BBNR rule to distinguish new cases will not deteriorate the effectiveness of PECS on static datasets.

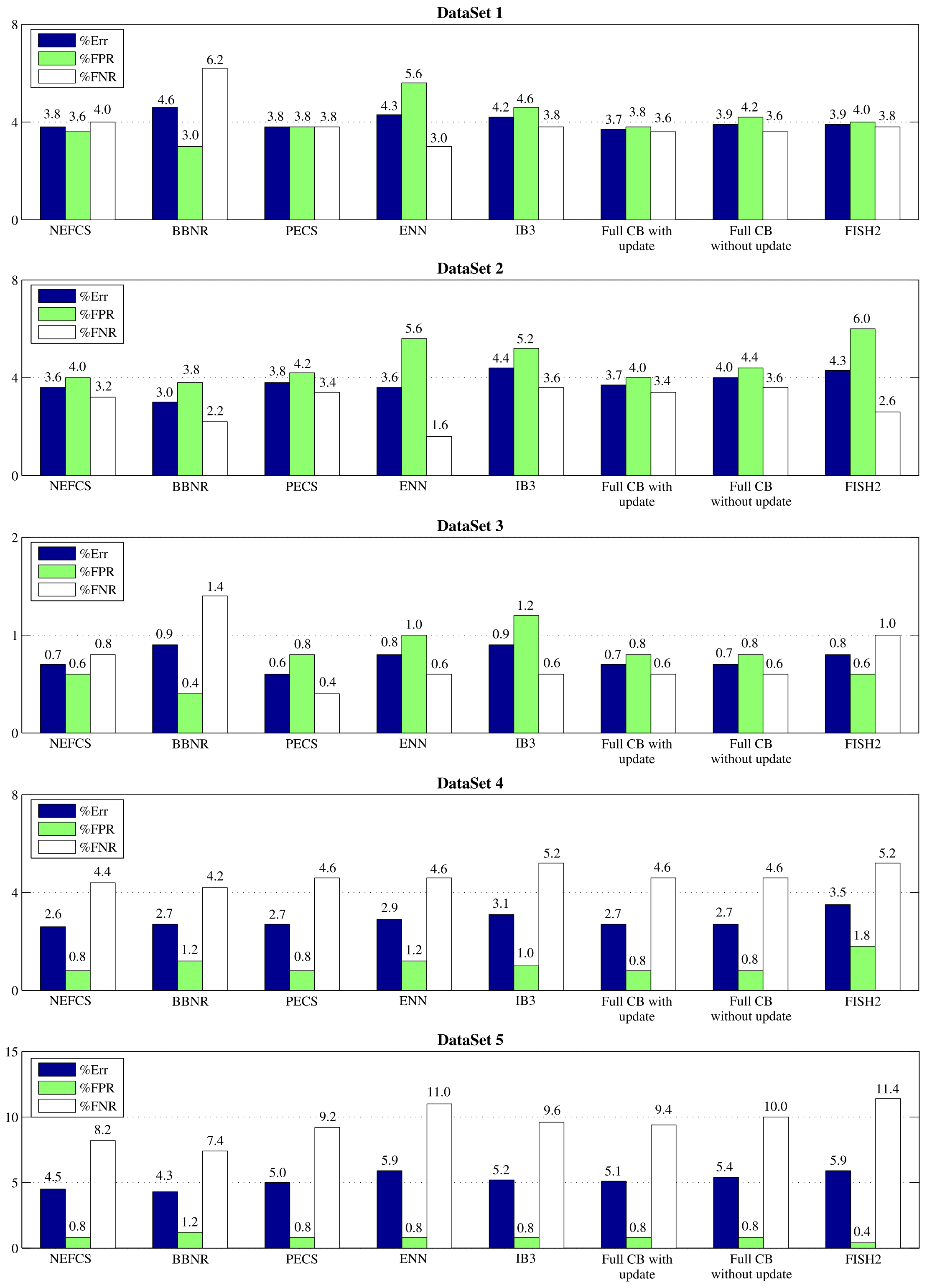

Fig. 8. Results on spam filtering static datasets.

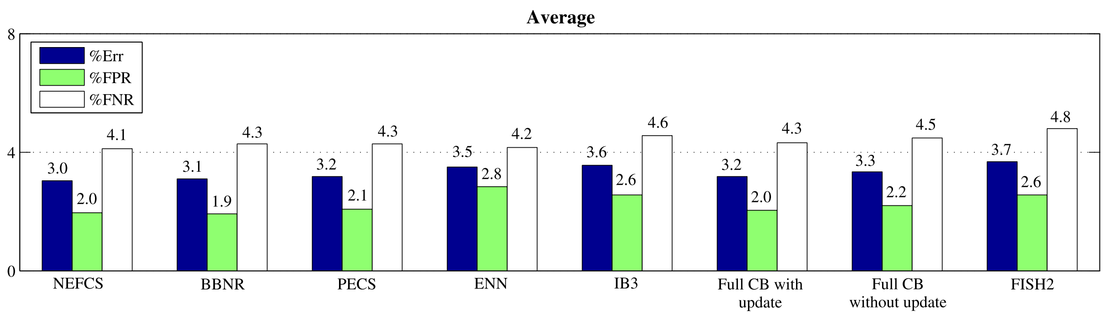

Fig. 9. Average results on spam filtering static datasets.

Table 1
Average training time (training) and execution time (classification) for each email (in seconds).

	NEFCS	BBNR	PECS	ENN	IB3	FISH2	Full CB
Training	180	116	17	115	19	N/A	N/A
Classification	0.5	0.1	0.2	0.1	0.2	0.2	0.1

Table 2
Summary of concept drift spam filtering datasets.

	Feb'03	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan'04	Total
Dataset 1													
Spam		629	314	216	925	917	1065	1225	1205	1830	576		8902
Non-spam		93	228	102	89	50	71	145	103	85	105		1071
Dataset 2													
Spam	142	391	405	459	406	476	582	1849	1746	1300	954	746	9456
Non-spam	151	56	144	234	128	19	30	182	123	113	99	130	1409

To demonstrate the efficiency of each of the algorithms compared, we list the average training time required as well as the execution time required to process each email in Table 1.

In general, competence-based methods (NEFCS, BBNR) need extra time for training, mainly because of building competence models. By contrast, on-line learning methods (NEFCS, PECS, IB3) need more time for classification, because they embed logic during the classification process to continuously improve effectiveness. However, it is rather unfair to make a straight comparison, because the training process is completely different and NEFCS is used in the reasoning process rather than being run once on the training set. In real world applications, the execution time is the actual time required to classify and learn incrementally for each new case. The extra execution time required by NEFCS implies a possible limitation of our proposed algorithm in application domains that require very fast performance. However, because the execution time of case-based algorithms depends heavily on the size of the case-base, performing redundancy reduction can help to improve the overall efficiency.

Experiment 2 *(Spam filtering, concept drift).* To evaluate how NEFCS behaves in real-world time-varying tasks, we choose to conduct studies on two concept drift spam filtering datasets [55] that can be downloaded from http://www.comp.dit.ie/sjdelany/Dataset.htm. Each dataset consists of more than 10,000 emails (spam or legitimate) collected over a period of approximately one year. A training set of 1000 cases (500 spam/500 legitimate), is given for each dataset. The remaining data is used to test our algorithm over time. Table 2 shows a summary of the test data across each month for both datasets.

Similar to Spam Filtering – static, each email e_i is represented by a vector of features $e_i = \langle x_1, x_2, \cdots, x_n \rangle$, where each feature is binary, and we use the same similarity as described in Experiment 1.

Since error rate is not a good metric for skewed datasets, the most common effectiveness metrics [66] for spam filtering are used to evaluate effectiveness, where Legitimate Recall (LR) is defined as $LR = \frac{TN}{TN+FP} = \frac{TN}{N}$, and Legitimate Precision (LP) is defined as $LP = \frac{TN}{TN+FN}$, where FP means legitimate emails that are incorrectly classified as spam, and FN means spam emails that are incorrectly classified as legitimate.

In this experiment, we again compare our algorithm with two closely related concept handling methods, IB3 [50] and PECS [53], as well as other well-known competence enhancement methods designed for static tasks, ENN [33] and BBNR [5], and the Full Base which performs no noise removal at all. Similar to Experiment 1, all the algorithms compared in this experiment, with the exception of IB3 and NEFCS, immediately retain a new case after classification, which leads to significantly better results. ENN and BBNR are applied monthly to remove any resulting noisy cases, because it is not feasible to check the whole case-base after every single case retention. We do not make a comparison with the original

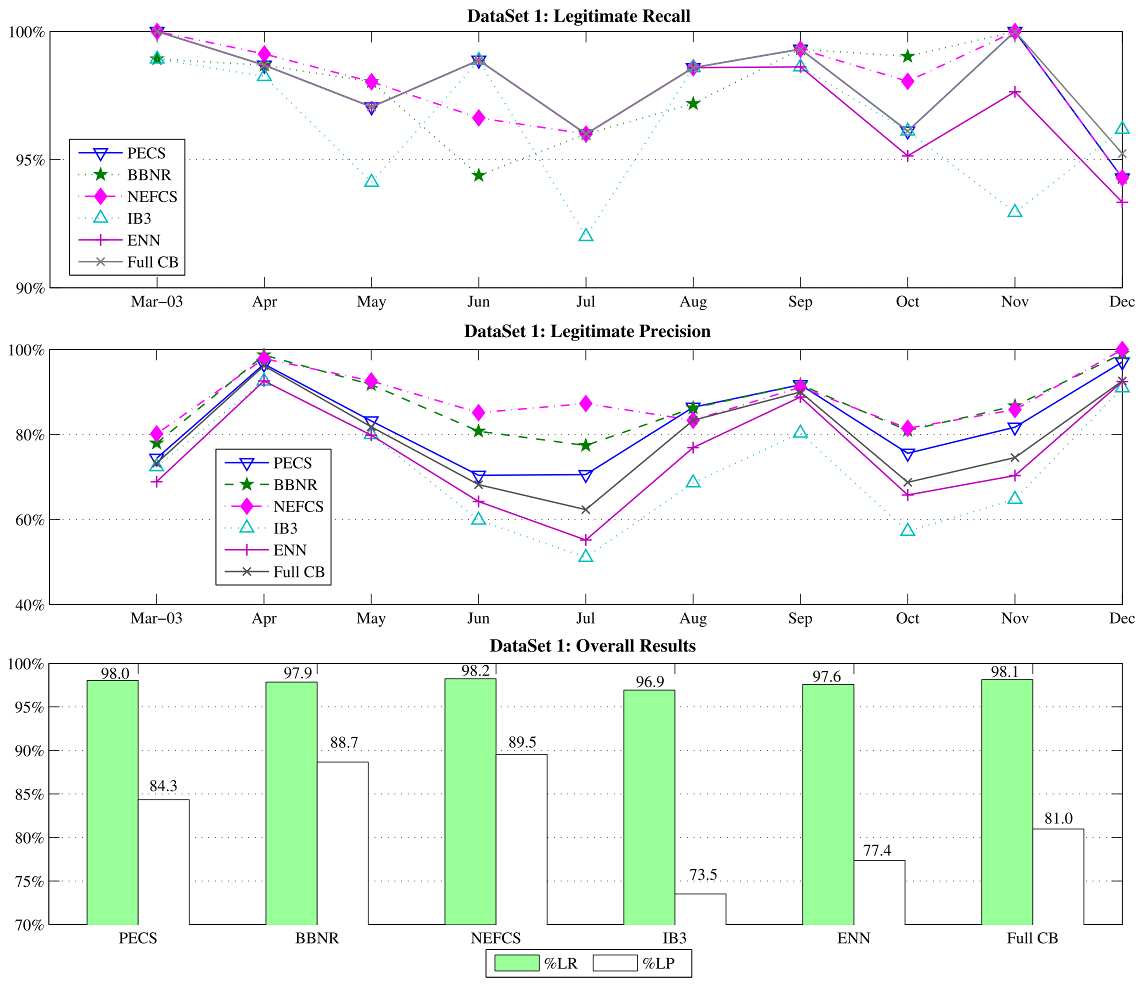

Fig. 10. Results of Stage-1 learning on spam filtering concept drift dataset 1.

work of Delany et al. [55], since their proposed CBE algorithm is a hybrid method of both competence enhancement and competence preservation. We intentionally leave this comparison for Section 4.3.

We construct the competence model in the same way as described in Experiment 1. To detect concept drift, we adopt two sliding windows. Each window contains all the emails received during the last 30 days. As a result, we do not detect concept drift in the first two months and assume there is no concept drift. Lastly, we use the same classifier and parameters as described in Experiment 1. The classification results for each month for both datasets are shown respectively in Fig. 10 and Fig. 11.

From these results, we conclude that:

- NEFCS improves the overall results on both LR and LP for dataset 1 over all the algorithms compared.
- To compare multiple algorithms over multiple datasets, we perform the same statistical tests as in Experiment 1. The result indicates a significant difference in LP at a confidence level of 99%, as well as a significant difference in LR at a confidence level of 95% across all compared algorithms. Again, by setting NEFCS as the control classifier, the Bonferroni–Dunn test reports a significant improvement in LP over ENN ($z > 10$), IB3 ($z > 10$), Full CB ($z = 7.9$), and PECS ($z = 3.7$) at a confidence level of 95%, and a significant improvement in LR over IB3 ($z = 4.6$) at a confidence level of 95%.
- Although the Bonferroni–Dunn post-hoc test is not able to pick up the difference between NEFCS and BBNR, a separate Wilcoxon signed-rank test comparing NEFCS and BBNR reports a significant improvement in LP at a confidence level of 95%, with no loss in LR (the difference is not statistically significant).
- NEFCS mainly improves LP. This is because we are using k-NN with unanimous voting. As a result, mistakenly retaining one or two noisy spam emails in the same competence area will not affect the classification result for a legitimate email; however, mistakenly retaining a legitimate email will dramatically affect the classification result for a spam email.

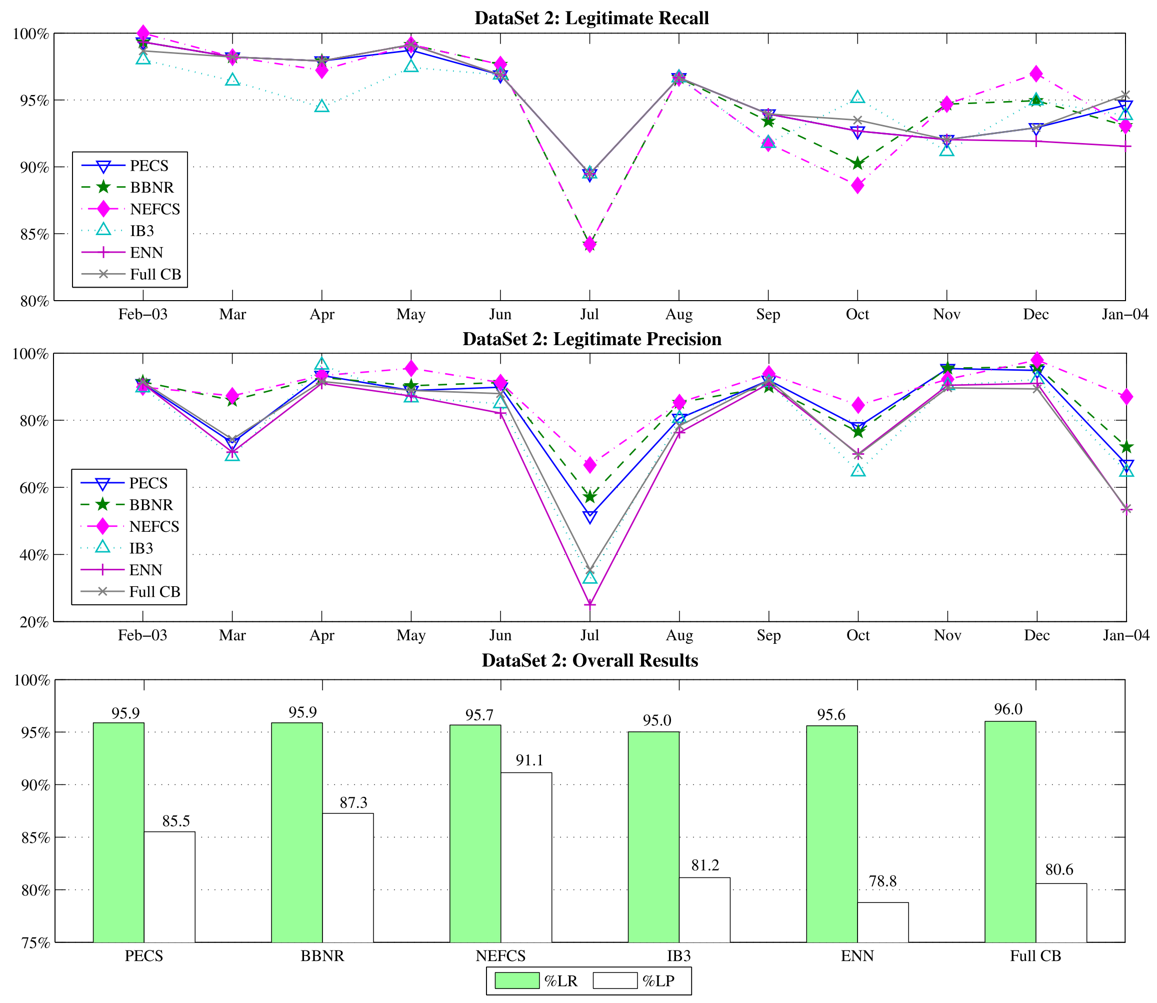

Fig. 11. Results of Stage-1 learning on spam filtering concept drift dataset 2.

- NEFCS is less affected by unknown concept drift and can quickly react to it. The proof is that there is a decreasing proportion of legitimate email from May to July for dataset 1, and a sudden fall in the proportion of legitimate email in July for dataset 2. As a result, all algorithms suffer from a loss of LP during these periods. NEFCS catches this trend and performs the best of all the algorithms compared during these periods, even improving the LP in July for dataset 1.
- NEFCS lies between PECS and IB3 by selectively retaining new cases. Experimental evaluation reveals that this strategy is better when coping with concept drift than either retaining or de-activating all new cases.

Experiment 3 *(Weather, concept drift).* To evaluate how NEFCS performs in a different application domain, we conduct a comparison on a weather dataset, which is also used in [18,67] as a concept drift dataset and can be downloaded from ftp.ncdc.noaa.gov/pub/data/gsod. The weather dataset is a real world dataset compiled by the U.S. National Oceanic and Atmospheric Administration, which collects meteorological measurements from over 9000 weather stations worldwide. Each instance in the dataset contains eight daily meteorological measurements (such as temperature, pressure, wind speed, etc.). The learning task is to predict whether precipitation occurred on each day. Lastly, we choose the data from Offutt Air Force Base in Bellevue, Nebraska for the same reason mentioned in [67], i.e., it contains records and details of diverse weather patterns over a 50-year term, which presents a long-term precipitation classification drift problem. This dataset contains 18,159 instances in total, with 5698 (31%) positive (rain) and 12,461 (69%) negative (no rain). The experiment is setup as follows:

Instead of directly using the actual meteorological measurements, we extract four principal components, whose eigenvalues are greater than or equal to 1. The similarity between instances is then calculated based on Euclidean distance.

For this experiment, the size of the initial training set is set to 30, with no redundancy removal. The experiment is designed in this way for two reasons: 1) In Learn++.NSE [18], “each training batch consisted of 30 samples (days), with corresponding test data selected as the subsequent 30 days. Thus, the learner is asked to predict the next 30 days’ forecast, which becomes the training data in the next batch”; 2) FISH2 needs to select a training set from all the historical data. For

Table 3
Average accuracy of weather dataset.

NEFCS	BBNR	PECS	IB3	Learn++.NSE	ENN	FISH2	Full base
74.4	73.8	72.7	72.1	75.9	71.7	72.1	72.1

Table 4
Characteristics of UCI datasets.

Dataset	Number of instances	Number of features	Number of classes
Balance-scale	625	4	3
Breast tissue	106	9	6
Ecoli	336	7	8
Glass identification	214	9	7
Haberman's survival	306	3	2
IRIS	150	4	3
Transfusion	748	4	2
Vertebral – 2 classes	310	6	2
Vertebral – 3 classes	310	6	3
Wine	178	13	3
Wine Quality (red)	1599	11	11
Wine Quality (white)	4898	11	11
Yeast	1484	8	10
Zoo	101	16	7

BBNR and ENN, the noise removal process is performed every 1000 instances, which makes the comparison not entirely fair; however, it is impractical to examine the whole case-base for each new learnt instance.

Lastly, we use k-NN as the base classifier for all compared algorithms with $k = 3$ using majority voting. For this dataset, FISH2 is implemented as it is originally described [58] and all other parameter settings are the same as described in Experiment 1.

Unlike the spam filtering domain, there is no implication to differentiating the costs of misdetection and false alarm; therefore, we choose to measure all the compared methods through overall classification accuracy. Table 3 shows the comparative results.

In this experiment, Learn++.NSE achieves the best overall accuracy, followed by NEFCS and BBNR; Since there is no significant difference between the compared algorithms, and on the other hand the dataset bears no clear information about when concept drift occurs, only the overall accuracy is given for this experiment.

4.2. Evaluating SRR

Before we move on to the hybrid approach of competence enhancement and competence preservation for dealing with concept drift, we evaluate SRR as an independent method of competence preservation on static tasks, for the reason described at the beginning of Section 4.1.

Experiment 4 *(UCI data. Static).* To evaluate how SRR behaves in static tasks, we conduct studies on 14 static classification datasets downloaded from the UCI Repository for Machine Learning Databases. We compare the results with NUN-CNN [64], ICF [49], and CRR [5], and the benchmark is a full case-base with no redundancy reduction. The characteristics of these 14 datasets are reported in Table 4.

Since our goal is to compare different redundancy reduction approaches in various situations, but not to find the most appropriate case retrieval metric, we simply use Euclidean distance and weight all features equally,

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3}$$

where $x = \langle x_1, x_2, \cdots, x_n\rangle$, $y = \langle y_1, y_2, \cdots, y_n\rangle$.

We use the NN rule for classification. To construct the competence model, a case c is considered to solve a case c' if $d(c, c') < d(c', c'_1)$, where c'_1 is the NUN of c'. This is the same method of constructing a competence model as used in ICF [49]. For a fair comparison of all the algorithms, and to focus on the aspect of redundancy reduction, no algorithm performs a noise removal pre-process. Note that ICF originally adopts ENN as a noise removal pre-process.

Lastly, we use 10-fold cross-validation for each dataset. We divide the dataset into ten stratified divisions or folds. Each fold in turn is considered as the testing set with the remaining nine folds acting as the training set. We sum the number of misclassifications for each testing fold and training set combination and make an overall evaluation of each individual dataset. The results are shown in Table 5, with the best accuracy for all the algorithms compared highlighted in bold.

Table 5
The classification accuracy (Acc.) and storage requirements (Stor.) for each dataset.

Dataset	Benchmark		SRR		CRR		NUN-CNN		ICF	
	Acc.	Stor.	Acc.	Stor.	Acc.	Stor.	Acc.	Stor.	Acc.	Stor.
Balance-scale	78.2	100	69.0	41.2	63.8	42.2	68.0	31.7	**71.4**	41.2
Breast tissue	56.6	100	**62.3**	56.6	58.5	74.2	**62.3**	57.6	60.4	67.4
Ecoli	79.5	100	75.9	32.7	**76.8**	44.3	76.2	32.1	74.4	34.6
Glass identification	72.4	100	65.4	40.1	**68.7**	57.2	66.8	42.9	65.4	48.8
Haberman's survival	73.4	100	70.1	42.9	**70.6**	57.7	69.2	42.9	68.2	49.3
IRIS	96.0	100	**95.3**	11.7	94.0	16.2	94.7	11.8	94.0	46.3
Transfusion	68.3	100	66.6	55.3	**69.8**	66.4	65.5	42.8	66.8	69.1
Vertebral – 2 classes	83.6	100	81.9	28.3	81.0	42.1	81.0	26.9	**82.6**	30.1
Vertebral – 3 classes	83.2	100	78.1	28.9	79.0	43.5	78.7	27.0	**79.4**	30.9
Wine	77.5	100	74.7	37.6	**78.7**	49.5	74.7	37.8	74.2	44.1
Wine quality (red)	59.6	100	58.6	57.5	58.7	68.8	**58.9**	57.1	58.7	72.4
Wine quality (white)	59.7	100	58.6	58.9	**59.2**	69.1	58.7	58.8	58.1	78.0
Yeast	52.6	100	49.8	62.2	**50.3**	76.1	50.1	60.5	48.8	65.8
Zoo	97.0	100	94.1	15.8	**96.0**	22.9	94.1	16.6	**96.0**	72.4
Average	74.1	100	71.5	40.7	**71.8**	52.2	71.3	39.0	71.3	53.6

Table 6
Compare classification accuracy (Acc.) on the same level of storage requirements (Stor.)

Dataset	Benchmark		SRR		CRR	
	Acc.	Stor.	Acc.	Stor.	Acc.	Stor.
Balance-scale	78.2	100	**69**	42.2	63.8	42.2
Breast tissue	56.6	100	**58.5**	74.2	**58.5**	74.2
Ecoli	79.5	100	**78.3**	44.3	76.8	44.3
Glass identification	72.4	100	**70.1**	57.2	68.7	57.2
Haberman's survival	73.4	100	**72.4**	57.7	70.6	57.7
IRIS	96	100	**95.3**	16.2	94	16.2
Transfusion	68.3	100	66.9	66.4	**69.8**	66.4
Vertebral – 2 classes	83.6	100	**82.9**	42.1	81	42.1
Vertebral – 3 classes	83.2	100	**82.3**	43.5	79	43.5
Wine	77.5	100	77	49.5	**78.7**	49.5
Wine quality (red)	59.6	100	**59**	68.8	58.7	68.8
Wine quality (white)	59.7	100	58.9	69.1	**59.2**	69.1
YEAST	52.6	100	**50.8**	76.1	50.3	76.7
Zoo	97	100	**96**	22.9	**96**	22.9
Average	74.1	100	**72.7**	52.2	71.79	52.2

At first analysis, we find that the average accuracy over these 14 datasets is very similar. Although CRR has slightly better generalization accuracy, winning eight of the 14 datasets, SRR and NUN-CNN require 10% less storage on average compared to CRR and ICF.

Because different algorithms result in different levels of accuracy and storage requirements, it is difficult to conduct a fair comparison. We decide to further compare SRR with CRR. We force the SRR algorithm to stop when it reaches the same storage as CRR. This is also a claimed advantage of our SRR algorithm, which allows the size of the case-base to be controlled. The results are shown in Table 6. Again, the best accuracies are highlighted in bold.

This time, SRR wins nine of the 12 datasets and behaves the same on two datasets. Even though this improvement is not statistically significant according to a Wilcoxon signed-rank test ($p = 0.08$), it would be safe to claim that SRR behaves as well as CRR for static tasks. In addition, we find that a redundancy reduction algorithm alone is unable to improve generalization accuracy on static tasks.

To show the efficiency of our SRR algorithm, we list the execution time required to minimize the case-base by each of the algorithms compared for a number of datasets, as illustrated in Table 7. For algorithms that require competence models, such as CRR, SRR, ICF, the execution time also includes the time required to build these competence models. In some real world applications, where competence models may be built and maintained for multiple purposes, it is not necessary to rebuild competence models when performing redundancy reduction. These running times were obtained from our unoptimized C# code on a laptop PC with a 2.3 GHz Intel i5 processor and 4 GB memory.

From the above table, we conclude that competence-based redundancy removal algorithms exhibit a visible advantage over NUN-CNN because they avoid the repeated cost of searching the case-base by referring to competence models. The execution time grows exponentially as the size of the case-base grows, which is mainly the result of the increasing number of case comparisons required to generate competence models. In addition, ICF is much slower than CRR and SRR, because as an iterative algorithm, ICF constantly needs to rebuild competence models. By contrast, SRR avoids this cost by maintaining its three unique lists. Finally, although CRR is slightly faster than SRR, the two algorithms are very close in execution time.

Table 7
Compare the execution time (milliseconds).

DataSet	Number of instances	CRR	SRR	NUN-CNN	ICF
IRIS	150	10	11	15	11
Ecoli	336	39	50	94	69
Balance-scale	625	92	131	183	165
Transfusion	748	112	130	369	337
YEAST	1484	804	875	3483	2757
Wine quality (white)	4898	11,847	12,806	52,639	38,728

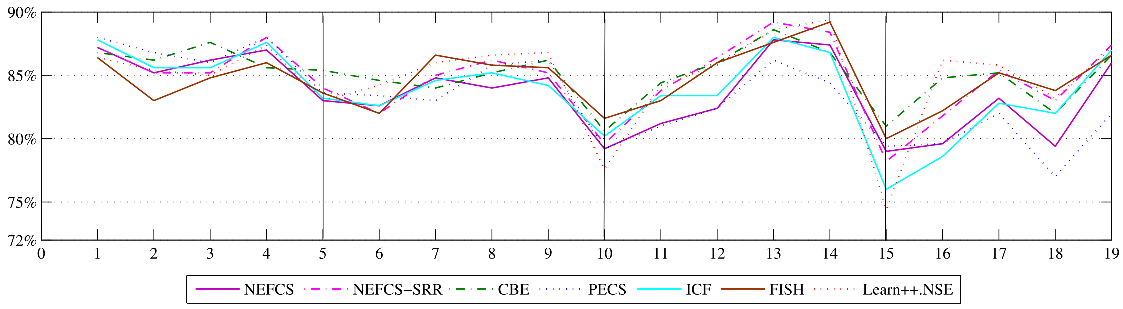

Fig. 12. Classification accuracy over sequential data blocks for the synthetic SEA data.

4.3. *Evaluating overall approach*

In this section, we will first evaluate our proposed NEFCS-SRR approach on two public online artificial concept drift datasets (http://www.win.tue.nl/~mpechen/data/DriftSets/) to reveal its behavior towards different types of concept drift (e.g., sudden or gradual). We then conduct experiments on real world data and compare our NEFCS-SRR approach with the most popular CBM algorithms as well as a recent ensemble approach [18].

Experiment 5 *(SEA concepts, sudden concept drift).* There are four blocks of data with different concepts. Each block contains 2500 random points of three-dimensional feature space. The three features have values randomly generated in the range [0; 10), and only the first two features are relevant. In each block, a data point belongs to class 1, if $f_1 + f_2 \le \theta$, where f_1 and f_2 represent the first two features, and θ is a threshold value for the two classes. Threshold values for the four data blocks are 8, 9, 7 and 9.5 in sequence. 10% class noise is introduced into each block of data by randomly changing the class value of 10% of instances.

The experiment is set up as follows: The first 500 data points from the first block are used as the training set. The remaining 2000 points, together with the other three blocks, are classified and learnt incrementally. The window size is chosen as 500 points. The case-base size limit is set to 1000, except for FISH2 and Learn++.NSE which do not perform redundancy removal. For CBE [55] and ICF [49], new instances are learnt immediately, while a noise removal process is performed every 500 points. With the k-NN rule, where $k = 5$, we record classification accuracy for every 500 points, therefore a sudden concept drift will occur at the 5th, 10th and 15th records respectively. Fig. 12 reports the accuracy of the different algorithms evaluated in this experiment.

From these results, we conclude that:

- There is a significant difference by Friedman test ($p < 0.01$) in classification accuracy across all the algorithms compared. By setting NEFCS-SRR as the control classifier, we conduct the Bonferroni–Dunn test as a post-hoc test. The result reveals a significant improvement over NEFCS at a confidence level of 95% ($z = 2.74$). Therefore, we claim that the proposed SRR algorithm helps to improve the learning capability for sudden concept drift.
- Although there is no statistical difference between NEFCS-SRR, FISH2 and CBE, visually we notice that just as sudden concept drift occurs (on the 5th, 10th and 15th records), NEFCS-SRR experiences a sharper drop in accuracy than FISH2 and CBE. This is probably because NEFCS-SRR adopts a concept drift detection algorithm and re-acts differently whether or not this is a reported concept drift. Therefore, before a concept drift can be detected, NEFCS-SRR will try to modify the case-base to preserve the previous learnt concept. However, NEFCS-SRR recovers after the concept drift with a higher slope than CBE and eventually achieves the best overall accuracy.

Experiment 6 *(Rotating hyperplane, gradual concept drift).* To evaluate the ability of our approach to handle gradual concept drift, we apply it to the commonly used synthetic benchmark data based on a rotating hyperplane. A hyperplane in d-dimensional data is denoted by equation $\sum_{i=1}^{d} a_i x_i = a_0$. Examples satisfying $\sum_{i=1}^{d} a_i x_i \ge a_0$ are labeled as positive, and $\sum_{i=1}^{d} a_i x_i < a_0$ as negative. Random samples are generated and uniformly distributed in space $[0, 1]^d$. Weights a_i are initial-

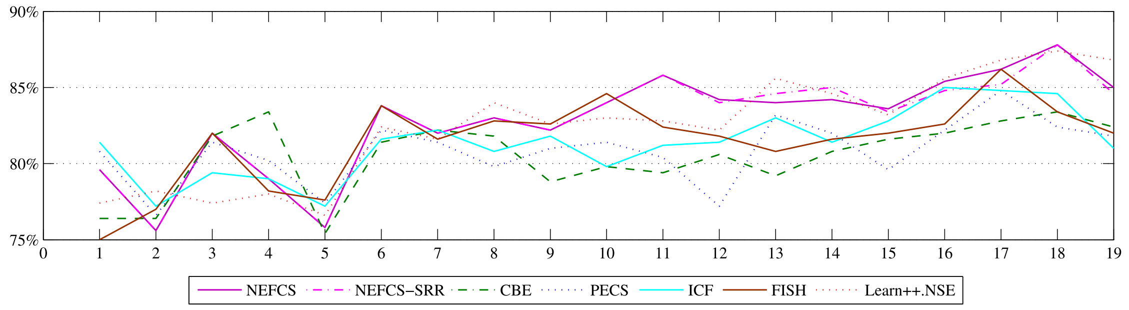

Fig. 13. Classification accuracy over sequential data blocks for the synthetic hyperplane data.

ized randomly in the range of [0, 1]. a_0 is set to $\frac{1}{2}\sum_{i=1}^{d} a_i$ so that the hyperplane divides the space into two parts of the same volume. Noise is introduced by randomly switching the labels of 5% of the examples.

Concept drift is simulated by two parameters. K specifies the total number of dimensions whose weights are changing. T specifies the magnitude of change (every N examples) for weights $a_1, \ldots, a_k$, and $s_i \in \{-1, 1\}$, $1 \le i \le K$ specifies the direction of change for each weight. Weights change gradually, i.e. adjusted by $s_i \times T/N$ after each example has been generated. Furthermore, there is a 10% possibility that the change will reverse its direction after every N examples; that is s_i will be replaced by $-s_i$ with a probability of 10%. Lastly, each time the weights are updated, we recompute $a_0 = \frac{1}{2}\sum_{i=1}^{d} a_i$ so that the class distribution is not changed. We choose one dataset of 10,000 instances and 10 dimensions, using $K = 5$, $T = 0.5$ and $N = 1000$. The experiment is set up in the same way as described in Experiment 4. Concept drift occurs gradually over all 10,000 instances (including the 500 instances for training). Fig. 13 reports the accuracy of the different algorithms evaluated in this experiment.

We perform the same statistical test as described above in Experiment 5. As a result, NEFCS-SRR shows a statistically significant improvement over CBE ($z = 3.75$), PECS ($z = 3.08$) at a confidence level of 95%. Compared with Experiment 5, NEFCS and NEFCS-SRR exhibit strong effectiveness against gradual concept drift. This is because when concept drifts gradually, the adopted concept drift detection method continuously reports a small region in which the concept changes, which enables our proposed approach to perform a well-targeted editing strategy.

Experiment 7 *(Spam filtering, concept drift).* To evaluate our proposed case-base editing technique, we make a comparison with other CBM approaches that target both competence enhancement and competence preservation, including CBE [55], ICF [49] and TWF [53] using the concept drift dataset described in Experiment 2. We also compare our results with FISH2 [58] and another recent ensemble approach for handling concept drift [18]. We do not make a comparison with IBL-DS [54] since there is no easy way to remove instances in a spatially uniform but temporally skewed way for the spam filtering domain, where the number of dimensions is extremely high, and all features are binary.

We set up a case-base size limit for all algorithms except for FISH2 [58] and Learn++.NSE [18]. We do not incorporate any redundancy removal operation for FISH2 for the reason described previously in Experiment 3. We do not incorporate any redundancy removal operation for Learn++.NSE, because: 1) we intend to retain their original methods; 2) each base classifier in Learn++.NSE is relatively small, which meets the speed and storage requirements. We perform each approach's corresponding competence preservation method when the case-base exceeds its size limit. If we stop any applied competence preservation method at the pre-set case-base size, any forthcoming case retention will trigger the competence preservation process again, therefore ICF and SRR will continue to complete their current round during which the limit is fulfilled, while CRR as an addition-based method has its own stopping point. We retain 1500 cases for dataset 1 and 2500 cases for dataset 2. This limit is determined according to original work using these datasets [55], which reports that “the resulting size of the case-base after all the data has been applied (i.e., after ten months for dataset 1 and 12 months for dataset 2) is 1512 and 2518, respectively”. Considering that we are using k-NN with unanimous voting rather than the traditional k-NN rule, and a FP is significantly more serious than a FN, SRR locks the closest k cases, rather than $(k+1)/2$. Again, we force the FISH2 algorithm to start with a training set of 200 instances for the same reason described in Experiment 1, which generates significantly better results. For Learn++.NSE [18], a new base classifier (case-based) is trained monthly. The rest of the experiment setup is the same as described in Experiment 2.

The classification results of each month for both datasets are shown respectively in Fig. 14 and Fig. 15.

These results clearly demonstrate that our proposed NEFCS-SRR approach achieves the best score on LR and LP for both datasets. To further compare the results, we perform the same statistical test as described above in Experiment 4, which implies a statistically significant difference in LP and LR ($p < 0.01$) across all compared algorithms. The result of the Bonferroni–Dunn test indicates that NEFCS-SRR achieves a significant improvement on LP over ICF ($z = 4.39$), Learn++.NSE ($z = 5.8$) and TWF ($z = 3.18$) at a confidence level of 95%, and a significant improvement over CBE ($z = 2.42$) at a confidence level of 90%. Although there is no significant improvement over FISH2 in LP, NEFCS-SRR shows a significant improvement on LR over FISH2 ($z = 4.19$) and Learn++.NSE ($z = 3.06$) at a confidence level of 95%. This proves that our method accurately captures the concept drift and performs well on targeted reactions.

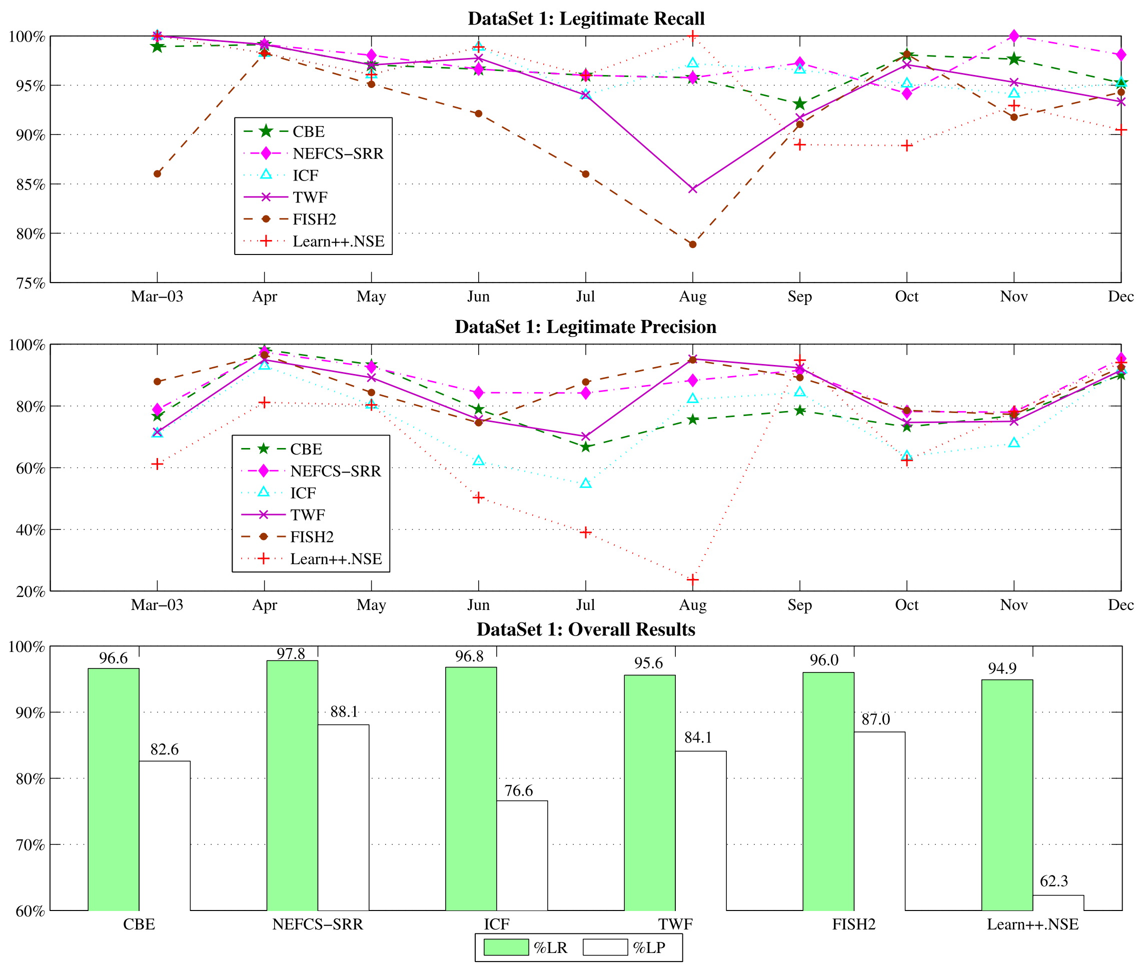

Fig. 14. Results of Stage-2 learning on spam filtering concept drift dataset 1.

An interesting finding is that although the ensemble approach has been reported as the most common and successful way of handling concept drift [16], Learn++.NSE leads to the worst result on most months for both datasets in this experiment, which is principally for the following two reasons: 1) The ensemble approach adopted in Learn++.NSE performed a postponed updating schema, i.e., a new base classifier was trained monthly. As a result, the system could not take advantage of the up-to-date feedback and make a timely adaptation. It would appear that an online learning schema is more suitable for the spam filtering domain where the feature space is large and sparse, and concept drift changes continuously; 2) The weight of each base classifier is calculated by a log operator which may generate an unlimited figure. As a result, a single base classifier can dominate the final prediction result. In our experiments at time t, the base classifier h_t trained with the latest cases D_t always achieved the best performance on D_t. Note that the weight of each base classifier is assigned based on its behavior on D_t; however, when concept drifts, the incoming dataset D_{t+1} may follow a different distribution of D_t. Unfortunately, Learn++.NSE does not incorporate a drift detection mechanism to cope with this situation.

Comparing NEFCS-SRR, CBE and ICF, we find that although ICF also adopts a competence-guided approach to the removal of redundant cases, the way in which it works may generate competence holes when concept drift occurs. Therefore, a conservative competence preservation method is recommended.

5. Conclusion and further study

Case-base Editing is an important aspect of CBM research that adopts an instance selection approach for handling concept drift. Editing the case-base properly can continuously tune a case-based learner according to concept drift.

In this paper, we first present a competence enhancement method – NEFCS, to prevent noise from being included and to quickly adapt the case-base according to concept drift by the safe removal of obsolete cases. Instead of blindly trusting all the most recent cases, NEFCS incorporates a concept drift detection method to differentiate the treatment on new cases based on whether there is concept drift. In addition, as an on-line learning algorithm, NEFCS can take immediate advantage of any system feedback, enabling a fast response to possible concept drift. Since it is almost impossible to increase the

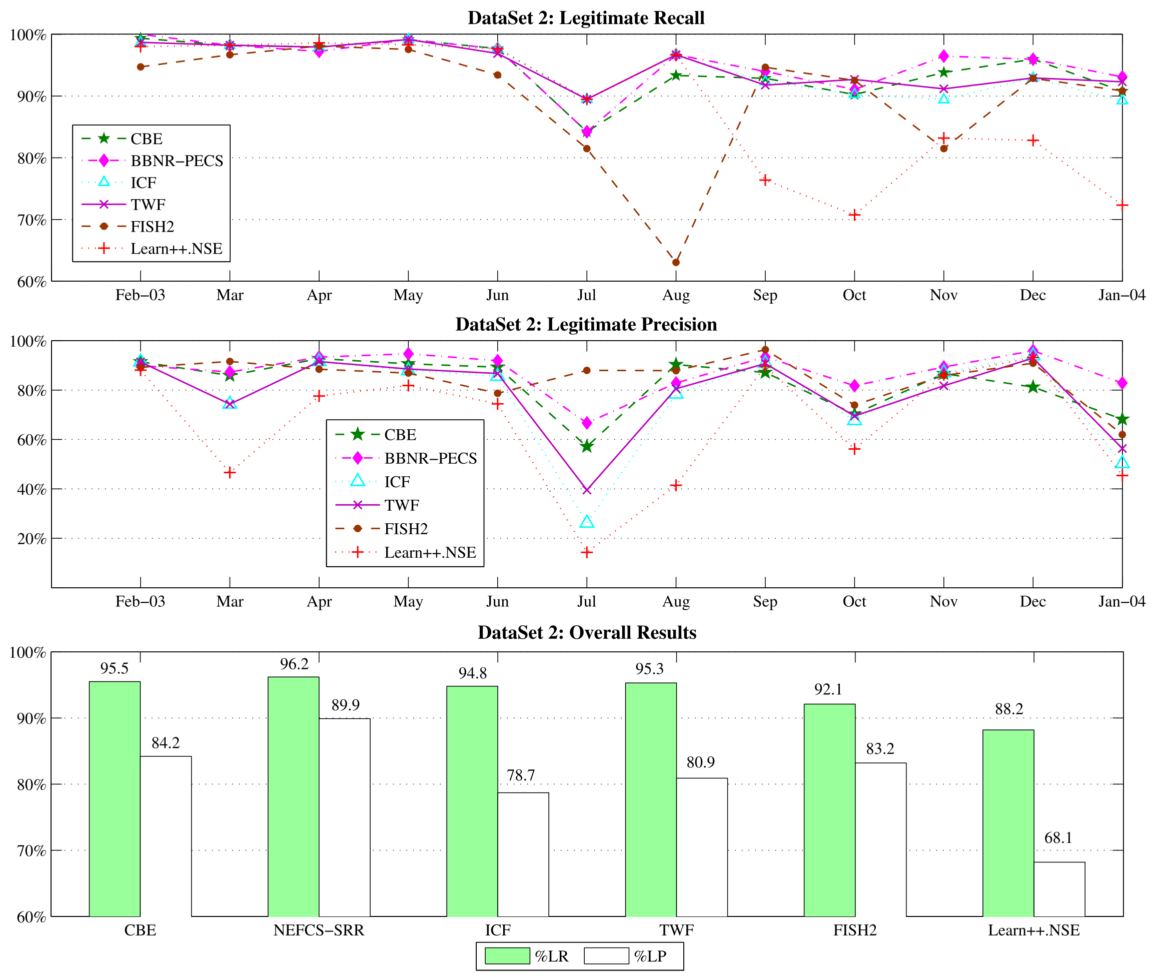

Fig. 15. Results of Stage-2 learning on spam filtering concept drift dataset 2.

case-base size without limit, we have also proposed a competence preservation method – SRR – to restrict the storage requirement. Two main features of SRR are that: First, SRR tries to preserve the case distribution during each iteration, which is very important for concept drift problems, e.g., to prevent competence holes, to aid concept drift detection. Second, as a competence-guided method, SRR preserves the competence of the case-base, i.e., it ensures that any removed instance can still be solved by the remaining case-base.

Several interesting findings have been revealed through our experiments: 1) FISH2 is not suitable for static tasks, because it forces to take into account the time factor whether or not concept drift occurs; 2) When sudden concept drift occurs, a certain degree of redundancy removal can help the learner to adapt more quickly to a novel concept; 3) Before concept drift can be detected, our proposed NEFCS-SRR tries to preserve previously learnt concepts; therefore compared to sudden concept drift it is believed to be more suitable for gradual concept drift; 4) In the spam filtering domain, where the problem space is very sparse, a competence-based instance selection technique has a clear advantage over other instance selection techniques; 5) Although each method behaves very differently based on the dataset, our proposed NEFCS-SRR approach constantly reports good overall accuracy across the datasets compared, and we therefore believe it is more general than the other methods compared; 6) Finally, in term of efficiency, all competence-based methods require extra time for maintaining the competence models; this makes a competence-based method inappropriate for applications that require real time decisions, such as network intrusion detection where an extremely high volume of packages arrives every second. However, for spam filtering, the effect of a little extra time is trivial.

From this research, we conclude that: 1) differentiating new cases according to drift detection is better than retaining or discarding all new cases indiscriminately; 2) a competence-guided, controllable, and conservative method is preferable to a non-competence-guided, uncontrollable, and unbalanced method of conducting competence preservation for time-varying tasks.

Instead of either retaining or removing a case completely, our next attempt will target the discovery of a fuzzy case weighting schema which will better balance the decision as to whether a case is noisy or whether it represents a novel concept. In addition, it would be interesting to consider our work in relation to the machine learning field, and to discover whether it is possible to plug our instance selection technique into other learning models.

Acknowledgements

The work presented in this paper is supported by the Australian Research Council (ARC) under Discovery Project DP150101645. We thank Weigang Zhao, Fang Dong and Anjin Liu for assistance with experiments. We would also like to show our gratitude to the editors and anonymous reviewers for comments that have greatly improved the manuscript.

References

[1] R. Lopez de Mantaras, D. McSherry, D. Bridge, D. Leake, B. Smyth, S. Craw, B. Faltings, M.L. Maher, M.T. Cox, K. Forbus, M. Keane, A. Aamodt, I. Watson, Retrieval, reuse, revision and retention in case-based reasoning, Knowl. Eng. Rev. 20 (3) (2005) 215–240.
[2] P. Cunningham, N. Nowlan, S.J. Delany, M. Haahr, A case-based approach to spam filtering that can track concept drift, in: ICCBR'03 Workshop on Long-Lived CBR Systems, Trondheim, Norway, Jun 24, 2003.
[3] M.K. Haouchine, B. Chebel-Morello, N. Zerhouni, Competence-preserving case-deletion strategy for case-base maintenance, in: 9th European Conference on Case-Based Reasoning, ECCBR '08, Trier, Germany, Sep. 1–4, 2008, pp. 171–183.
[4] D. Wilson, D.B. Leake, Maintaining case-based reasoners: dimensions and directions, Comput. Intell. 17 (2) (2001) 196–213.
[5] S.J. Delany, P. Cunningham, An analysis of case-base editing in a spam filtering system, in: 7th European Conference on Case Based Reasoning, ECCBR '04, Madrid, Spain, Aug. 30–Sep. 2, 2004, pp. 128–141.
[6] F. Angiulli, Fast nearest neighbor condensation for large data sets classification, IEEE Trans. Knowl. Data Eng. 19 (11) (2007) 1450–1464.
[7] G. Widmer, M. Kubat, Effective learning in dynamic environments by explicit context tracking, in: 6th European Conference on Machine Learning, ECML '93, Vienna, Austria, Apr. 5–7, 1993, pp. 227–243.
[8] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Mach. Learn. 23 (1) (1996) 69–101.
[9] P. Zhang, X. Zhu, Y. Shi, Categorizing and mining concept drifting data streams, in: 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, Las Vegas, NV, USA, Aug. 24–27, 2008, pp. 812–820.
[10] Q. Yang, X. Wu, 10 challenging problems in data mining research, Int. J. Inf. Technol. Decis. Mak. 5 (4) (2006) 597–604.
[11] M. Scholz, R. Klinkenberg, Boosting classifiers for drifting concepts, Intell. Data Anal. 11 (1) (2007) 3–28.
[12] C.-J. Tsai, C.-I. Lee, W.-P. Yang, Mining decision rules on data streams in the presence of concept drifts, Expert Syst. Appl. 36 (2) (2009) 1164–1178.
[13] A. Tsymbal, The problem of concept drift: definitions and related work, Technical report TCD-CS-2004-15, Trinity College Dublin, Ireland, 2004.
[14] Y. Li, Y. Wei, A. Kolesnikova, W.D. L, A new gradual forgetting approach for mining data stream with concept drift, in: International Symposium on Information Science and Engineering, ISISE '08, Shanghai, China, Dec. 20–22, 2008, pp. 556–559.
[15] Y. Koren, Collaborative filtering with temporal dynamics, in: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, Paris, France, June 28–July 1, 2009, pp. 447–456.
[16] W. Qu, Y. Zhang, J. Zhu, Q. Qiu, Mining multi-label concept-drifting data streams using dynamic classifier ensemble, in: 1st Asian Conference on Machine Learning, ACML '09, Nanjing, China, Nov. 2–4, 2009, pp. 308–321.
[17] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '09, Paris, France, June 28–July 1, 2009, pp. 139–148.
[18] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Trans. Neural Netw. 22 (10) (2011) 1517–1531.
[19] R. Klinkenberg, T. Joachims, Detecting concept drift with support vector machines, in: 17th International Conference on Machine Learning, ICML '00, Stanford, CA, USA, June 29–July 2, 2000, pp. 487–494.
[20] N. Street, Y. Kim, A streaming ensemble algorithm (SEA) for large-scale classification, in: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '01, San Francisco, CA, USA, Aug. 26–29, 2001, pp. 377–382.
[21] J.Z. Kolter, M.A. Maloof, Using additive expert ensembles to cope with concept drift, in: 22nd International Conference on Machine Learning, ICML '05, Bonn, Germany, Aug. 7–11, 2005, pp. 449–456.
[22] J. Gao, W. Fan, J. Han, On appropriate assumptions to mine data streams: analysis and practice, in: 7th IEEE International Conference on Data Mining, ICDM '07, Omaha, NE, USA, Oct. 28–31, 2007, pp. 143–152.
[23] P. Zhang, X. Zhu, Y. Shi, X. Wu, An aggregate ensemble for mining concept drifting data streams with noise, in: 13th Pacific-Asia Conference on Knowledge Discovery and Data Mining, PAKDD '09, Bangkok, Thailand, Apr. 27–30, 2009, pp. 1021–1029.
[24] L.L. Minku, X. Yao, DDD: a new ensemble approach for dealing with concept drift, IEEE Trans. Knowl. Data Eng. 24 (4) (2012) 619–633.
[25] V. Attar, P. Chaudhary, S. Rahagude, G. Chaudhari, P. Sinha, An instance-window based classification algorithm for handling gradual concept drifts, in: 7th International Workshop on Agents and Data Mining Interaction, AMDI '11, Taipei, Taiwan, May 2–6, 2012, pp. 156–172.
[26] L.L. Minku, A.P. White, X. Yao, The impact of diversity on online ensemble learning in the presence of concept drift, IEEE Trans. Knowl. Data Eng. 22 (5) (2010) 730–742.
[27] J. Zhang, Selecting typical instances in instance-based learning, in: 9th International Workshop on Machine Learning, ML '92, Aberdeen, Scotland, UK, July 1–3, 1992, pp. 470–479.
[28] N. Lu, G. Zhang, J. Lu, Concept drift detection via competence models, Artif. Intell. 209 (2014) 11–28.
[29] B. Smyth, M.T. Keane, Remembering to forget: a competence-preserving case deletion policy for case-based reasoning systems, in: 14th International Joint Conference on Artificial Intelligence, IJCAI '95, Montreal, Quebec, Canada, Aug. 20–25, 1995, pp. 377–382.
[30] B. Smyth, E. McKenna, Competence models and the maintenance problem, Comput. Intell. 17 (2) (2001) 235–249.
[31] R. Pan, Q. Yang, S.J. Pan, Mining competent case bases for case-based reasoning, Artif. Intell. 171 (16–17) (2007) 1039–1068.
[32] A. Smiti, Z. Elouedi, Overview of maintenance for case based reasoning systems, Int. J. Comput. Appl. 32 (2) (2011) 49–56.
[33] D.L. Wilson, Asymptotic properties of nearest neighbor rules using edited data, IEEE Trans. Syst. Man Cybern. 2 (3) (1972) 408–421.
[34] I. Tomek, Two modifications of CNN, IEEE Trans. Syst. Man Cybern. 6 (11) (1976) 769–772.
[35] D.R. Wilson, T.R. Martinez, Reduction techniques for instance-based learning algorithms, Mach. Learn. 38 (3) (2000) 257–286.
[36] F.J. Ferri, E. Vidal, Small sample size effects in the use of editing techniques, in: 11th IAPR International Conference on Pattern Recognition, ICPR '92, Hague, Netherlands, Aug. 30–Sep. 3, 1992, pp. 607–610.
[37] B. Karaçalı, H. Krim, Fast minimization of structural risk by nearest neighbor rule, IEEE Trans. Neural Netw. 14 (1) (2003) 127–137.
[38] R. Pan, Q. Yang, J.J. Pan, L. Li, Competence driven case-base mining, in: 20th National Conference on Artificial Intelligence, AAAI'05, Pennsylvania, USA, 2005, pp. 228–233.
[39] S. Craw, S. Massie, N. Wiratunga, Informed case base maintenance: a complexity profiling approach, in: 22nd AAAI Conference on Artificial Intelligence, AAAI-07, Vancouver, British Columbia, Canada, July 22–26, 2007, pp. 1618–1621.

[40] L. Cummins, D. Bridge, Maintenance by a committee of experts: the MACE approach to case-base maintenance, in: 8th International Conference on Case-Based Reasoning, ICCBR'09, Seattle, USA, July 20–23, 2009, pp. 120–134.
[41] P.E. Hart, The condensed nearest neighbor rule, IEEE Trans. Inf. Theory 14 (3) (1968) 515–516.
[42] G.W. Gates, The reduced nearest neighbor rule, IEEE Trans. Inf. Theory 18 (3) (1972) 431–433.
[43] G. Ritter, H. Woodruff, S. Lowry, T. Isenhour, An algorithm for a selective nearest neighbor decision rule, IEEE Trans. Inf. Theory 21 (6) (1975) 665–669.
[44] V.S. Devi, M.N. Murty, An incremental prototype set building technique, Pattern Recognit. 35 (2) (2002) 505–513.
[45] C.-H. Chou, B.-H. Kuo, F. Chang, The generalized condensed nearest neighbor rule as a data reduction method, in: 18th International Conference on Pattern Recognition, ICPR '06, Hong Kong, China, Aug. 20–24, 2006, pp. 556–559.
[46] X. Hao, C. Zhang, H. Xu, X. Tao, S. Wang, Y. Hu, An improved condensing algorithm, in: 7th IEEE/ACIS International Conference on Computer and Information Science, ICIS'08, Portland, OR, USA, May 14–16, 2008, pp. 316–321.
[47] E. McKenna, B. Smyth, Competence-guided case-base editing techniques, in: 5th European Workshop on Case-Based Reasoning, EWCBR '00, Trento, Italy, Sep. 6–9, 2000, pp. 235–257.
[48] J. Zhu, Q. Yang, Remembering to add: competence-preserving case-addition policies for case-base maintenance, in: 16th International Joint Conference on Artificial Intelligence, IJCAI '99, Stockholm, Sweden, July 31–August 6, 1999, pp. 234–239.
[49] H. Brighton, C. Mellish, Advances in instance selection for instance-based learning algorithms, Data Min. Knowl. Discov. 6 (2) (2002) 153–172.
[50] D.W. Aha, D. Kibler, M.K. Albert, Instance-based learning algorithms, Mach. Learn. 6 (1) (1991) 37–66.
[51] M. Salganicoff, Density-adaptive learning and forgetting, in: 10th International Conference on Machine Learning, ICML '93, Amherst, MA, USA, June 27–29, 1993, pp. 276–283.
[52] R. Klinkenberg, Learning drifting concepts: example selection vs. example weighting, Intell. Data Anal. 8 (3) (2004) 281–300.
[53] M. Salganicoff, Tolerating concept and sampling shift in lazy learning using prediction error context switching, Artif. Intell. Rev. 11 (1) (1997) 133–155.
[54] J. Beringer, E. Hüllermeier, Efficient instance-based learning on data streams, Intell. Data Anal. 11 (6) (2007) 627–650.
[55] S.J. Delany, P. Cunningham, A. Tsymbal, L. Coyle, A case-based technique for tracking concept drift in spam filtering, Knowl.-Based Syst. 18 (4–5) (2005) 187–195.
[56] S.J. Delany, D. Bridge, Catching the drift: using feature-free case-based reasoning for spam filtering, in: 7th International Conference on Case-Based Reasoning, ICCBR'07, Northern Ireland, UK, August 13–16, 2007, pp. 314–328.
[57] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: 17th Brazilian Symposium on Artificial Intelligence, SBIA '04, Sao Luis, Maranhao, Brazil, Sep. 29–Oct. 1, 2004, pp. 286–295.
[58] I. Žliobaitė, Combining similarity in time and space for training set formation under concept drift, Intell. Data Anal. 15 (4) (2011) 589–611.
[59] S. Huang, Y. Dong, An active learning system for mining time-changing data streams, Intell. Data Anal. 11 (4) (2007) 401–419.
[60] P. Lindstrom, S.J. Delany, B.M. Namee, Handling concept drift in text data stream constrained by high labelling cost, in: 23rd International Florida Artificial Intelligence Research Society Conference, FLAIRS'10, Florida, USA, May 19–21, 2010, pp. 26–31.
[61] P. Lindstrom, B.M. Namee, S.J. Delany, Drift detection using uncertainty distribution divergence, Evol. Syst. 4 (1) (2013) 13–25.
[62] B. Smyth, E. McKenna, An efficient and effective procedure for updating a competence model for case-based reasoners, in: 11th European Conference on Machine Learning, ECML '00, Barcelona, Catalonia, Spain, May 31–June 2, 2000, pp. 357–368.
[63] K. Nishida, K. Yamauchi, Detecting concept drift using statistical testing, in: 10th International Conference on Discovery Science, DS '07, Sendai, Japan, Oct. 1–4, 2007, pp. 264–269.
[64] B. Smyth, E. McKenna, Building compact competent case-bases, in: 3rd International Conference on Case-Based Reasoning, ICCBR '99, Seeon Monastery, Germany, July 27–30, 1999, pp. 329–342.
[65] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[66] K.R. Gee, Using latent semantic indexing to filter spam, in: 18th Annual ACM Symposium on Applied Computing, SAC '03, Melbourne, FL, USA, Mar. 9–12, 2003, pp. 460–464.
[67] G. Ditzler, R. Polikar, Incremental learning of concept drift from streaming imbalanced data, IEEE Trans. Knowl. Data Eng. 25 (10) (2013) 2283–2301.