Neural Comput & Applic (2012) 21:671–682
DOI 10.1007/s00521-011-0663-x

ICONIP2010

# Adaptive cascade of boosted ensembles for face detection in concept drift

**Teo Susnjak · Andre L. C. Barczak · Ken A. Hawick**

Received: 18 April 2011 / Accepted: 5 June 2011 / Published online: 17 June 2011

**Abstract** We propose an adaptive learning algorithm for cascades of boosted ensembles that is designed to handle the problem of concept drift in nonstationary environments. The goal was to create a real-time adaptive algorithm for dynamic environments that exhibit varying degrees of drift in high-volume streaming data. This we achieved using a hybrid of *detect-and-retrain* and *constant-update* approaches. The uniqueness of our method is found in two aspects of our framework. The first is the manner in which individual weak classifiers within each cascade layer of an ensemble are clustered during training and assigned a competence value. Secondly, the idea of learning optimal cascade-layer thresholds during runtime, which enables rapid adaptation to dynamic environments. The proposed adaptive learning method was applied to a binary-class problem with rare-event detection characteristics. For this, we chose the domain of face detection and demonstrate experimentally the ability of our algorithm to achieve an effective trade-off between accuracy and speed of adaptations in dense data streams with unknown rates of change.

**Keywords** Ensemble-based learning · Adaptive learning · Concept drift · Nonstationary environments · Boosting · Cascades · AdaBoost · Face detection

T. Susnjak (✉) · A. L. C. Barczak · K. A. Hawick
Massey University, Albany, New Zealand
e-mail: T.Susnjak@massey.ac.nz

A. L. C. Barczak
e-mail: A.L.Barczak@massey.ac.nz

K. A. Hawick
e-mail: k.a.hawick@massey.ac.nz

## 1 Introduction

The underlying assumption of classical approaches for training classifiers has been that their operating domains are stationary [1, 2]. However, with the advent of streaming data and long life classification systems, it has become clear that these assumptions no longer hold. Consequently, the inadequacy of traditional techniques for application on emerging domains has become apparent [3, 4].

The data streams in dynamic domains such as e-commerce, economic and financial data analysis, sensor systems, email spam filtering and a host of other burgeoning fields, possess distributions and class descriptors that constantly change with time, due some *hidden context* [5]. Therefore, the *target concepts* from the original training data may no longer correspond with the current representations of the target concept. As a result of an increasing loss of relevance between the concepts' representations and current snapshots of data, a deterioration in the runtime accuracies ensues. This phenomenon is referred to as *concept drift* [6].

Variations in data can take a sudden or a gradual form and may also be recurring, whereby previously active concepts reappear in a randomly cyclical fashion. Although an increased understanding about the anticipated types of drifts improves the chances of devising a successful adaptive strategy [7], often the magnitude and frequency of drifting contexts are not known a priori. To a large degree, training effective classifiers is contingent on formulating strategies based on foreseen types of changes inherent to an operating domain. Due to this, training accurate classifiers for *unpredictable* nonstationary environments is nontrivial [8, 9] and is still an open problem.

The situation is further compounded by the fact that many target concepts we monitor for in streaming data are also rare-occurring [2]. Biased-class distribution problems of this kind can hinder and impose significant limitations on accuracies attainable by standard learning methods [10] and are a considerable ongoing challenge [2]. This situation is encountered in a large and growing list of domains such as adversarial monitoring environments like network intrusion, surveillance, and transaction fraud detection.

In addition to the trends that have given rise to expanding numbers of available data with seemingly unlimited and continuously flowing streams, there has also been a tendency for these streams to be dense and high-speed [11]. This poses difficulties for not only realizing real-time event detection, but even more so for modeling and handling concept-drifts in a timely manner. Failure to expeditiously update models to time-evolving environments will eventually result in a decrease in prediction accuracy [2]. For instance, this can be critical in robotics where there is a reliance on computer vision components. The incoming data are high-speed and large scale, with drifting concepts caused by different lighting conditions. A problem domain of this kind requires real-time processing and adaptation in order for time-sensitive tasks such as obstacle avoidance and planning to be effective [12].

## 1.1 Past research

Ensemble-based techniques have been shown to be an effective and scalable approach to addressing concept-drifting challenges in streaming data [11, 13–15]. Ensemble-based systems consist of multiple classifiers that can be viewed as a *committee of experts*, whose individual votes are combined in order to formulate a final classification. The modularity they afford gives them the ability to retain relevant historical information as well as to effectively incorporate new knowledge [16].

Generally, ensemble-classifiers maintain relevance to current snapshots of/streaming data through either the *constant-update* or *detect-and-retrain* methods [17]. Most ensemble systems belong to the former category, which simply assumes that drift is always taking place and consequently, and they continuously adapt the classifiers. This strategy comes with the loss of responsiveness to sudden drifts and a high model update-cost [17], while also introducing the risk of modeling noise. Alternatively, the *detect-and-retrain* strategies actively monitor for changes in the underlying distribution of data and trigger a classifier update once a predetermined criteria or performance threshold is surpassed. As a result, they are more responsive to sudden drifts [17].

Ensemble-based systems maintain an updated state using a combination of strategies. These typically involve dynamic modifications of classifier weights and voting techniques, mechanisms that add new competent classifiers and remove the irrelevant, as well as approaches employing batch (offline) and instance-based (online) learning [7, 16].

Batch learning approaches train on entire partitions of incoming streaming data, instead of single data instances. Recent approaches [1, 14, 16, 18] train new classifiers for every newly generated batch of data. In these cases, both the structure of an ensemble and the weights of existing classifiers are altered. Wang et al. [2] and Scholz and Klinkenberg [15] recognize the high update costs associated with continuous global updates of a model in high-density data streams. They devise a framework for rapidly revising only relevant components of an ensemble and initiate the training of new classifiers only when required.

Nishida et al. [9] employ a hybrid approach that combines both batch and online learning. Kolter and Maloof [19, 20] propose pure online learning approaches that train and incorporate new experts into the ensemble when the ensemble makes a collective mistake. Pockock [21] proposes a more optimal extension of Oza's Online Boosting [22] algorithm, on the basis that the modified version does not assume that the streaming data are independent and identically distributed.

Huang et al. [23] and Pelossof et al. [24] develop sequential learning algorithms for cascaded face detectors through a strategy of continuous updates of ensemble weights. Grabner et al. [25] approach the problem of concept drift in face detection by developing a hybrid tracker combined with a classification system.

## 1.2 Motivation

Our aim is to propose a drift-handling algorithm that can adapt in real time to evolving concepts in large volume and high-speed streaming data. We devise a model that makes no assumptions about the types of drift in its operating domain and can therefore be equally robust to all forms of drift. Lastly, we are motivated by the challenges presented by rare-class distributions to classification in dynamic domains.

Due to the lacking of robustness of existing artificial data sets for testing drift handling [8], as well as their insufficient ability to test scalability [26], we were motivated to create our own real-world dataset. We chose to apply our algorithm to computer vision and more specifically to the task of face detection. By implementing exhaustive raster scanning of a given image frame using subwindows of increasing sizes, we were able to create a suitable problem domain. The data set provided us with a large, high-speed data stream, where accuracy and real-time drift response are paramount. In addition, the problem domain met our requirement of possessing a biased-class distribution as well as naturally occurring concept drifts of varying magnitude.

### 1.3 Our contribution

The following summarizes our paper's contribution:

1. We propose a novel drift-learning framework for cascaded ensembles within binary-class problems
2. We introduce a low update-cost approach for ensembles in a cascaded structure, which enables real-time adaptation in high-speed data streams
3. We present results on a real-world problem data set involving computer vision and face detection

The algorithm is formulated on the ideas of boosted cascades in the form of Barczak et al. [27]. The method constructs clusters of ensembles within each layer that function as *nested cascades*. The novelty of our contribution is found in the unique way that competence values assigned to each cluster of ensembles within a layer are used to create optimal *layer confidence thresholds*. We show how these layer confidence thresholds can be trained in order to cope with all forms of drift without global model updates. The algorithm demonstrates an effective trade-off between accuracy and adaptation speed, in time-evolving environments with unknown rates of change and can process large volume data streams in near real time.

### 1.4 Paper organization

The structure of this paper is as follows: Sect. 2 describes the theory and implementation of the stationary cascaded ensemble-classifier that serves as input to our concept learning algorithm. This is followed by the detailed description of the concept-drift-handling algorithm. Sections 3 and 4 outline the experiment design and its results, respectively, while Sect. 5 includes a discussion before concluding remarks.

## 2 Proposed framework

We first describe the structure and the method for training a cascaded classifier on *static* data sets, which serves as input to our concept-drift-handling algorithm. This is then followed by a detailed description of the proposed algorithm together with related research.

### 2.1 Building the static cascaded ensemble

Barczak et al. [27] initially proposed a classifier training structure termed *Parallel Strong Classifier* within the same *Layer* (PSL), seen in Fig. 1. This was the extension of the work by Viola and Jones [28], who produced the seminal cascade of boosted ensembles (CoBE), seen in the same figure. The central idea of the work by Barczak et al. was to transform the CoBE into a cascade of layers that contained within them a supplementary nested *intra-layer* cascade. In effect, this created a quasi two-dimensional cascaded framework. This novelty was motivated by the desire to reduce overall training runtimes by accelerating

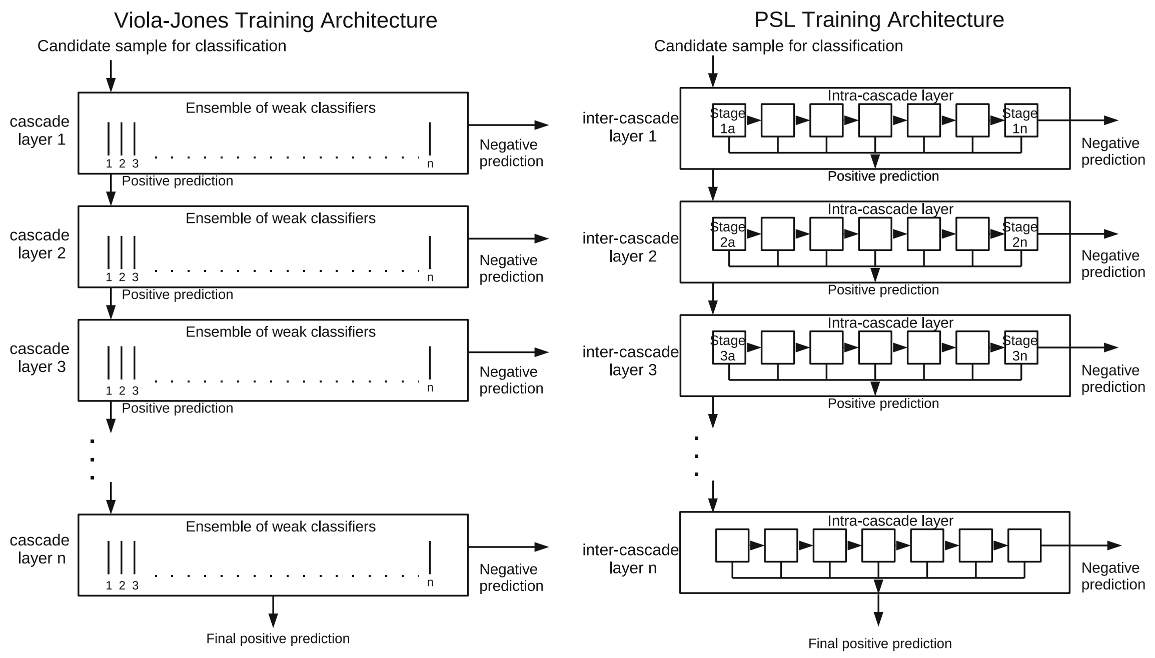

**Fig. 1** The standard cascade structure of Viola–Jones and the PSL structure [27], in which each stage within a layer is comprised of a predetermined maximum number of weak classifiers

convergence rates of hit and false acceptance rates to their specified layer targets. Furthermore, the goal was to design a strategy capable of guaranteeing 100% hit rates at every stage of a cascade without artificial threshold adjustments that are found in other approaches.

The training method for the PSL proceeds as follows: at the commencement of training for each layer of a cascade, all positive and negative samples are made available to the boosting algorithm. After a predetermined number of boosting rounds, the training for a given intra-layer stage ceases and a new independent one begins with all sample weights reset. Subsequently, each succeeding intra-layer stage is provided with all negative training samples, while only misclassified positive samples from the previous rounds are supplied for re-training until enough intra-layer stages are constructed to correctly classify all positive samples (Fig. 2).

The convergence to layer targets is rapidly realized since the size of the positive training data set decreases rapidly following each intra-layer stage, leaving considerably fewer but also more difficult positive samples to be learned. At detection time, any instance is classified as a positive if any one intra-layer stage predicts it as a positive, while a negative prediction is attained only when a unanimous vote among all intra-layer stages is reached.

The idea of a dual-cascaded structure was taken further in Susnjak et al. [29], in order to implement positive sample bootstrapping. This enabled the utilization of potentially massive positive data sets, without the learning algorithm being exposed to all samples explicitly, a task that would otherwise be computationally too expensive. The research showed that it was possible to repeatedly train each intra-layer stage using only a small subset of the positive data set. After constructing each intra-layer stage, the total positive data set would be sampled in order to create a new positive subset. This set would only consist of positive samples that were misclassified by all *preceding* intra-layer stages. The experiments demonstrated that it was not only possible to implement bootstrapping, but that the diversity of each intra-layer stage increased. Since each intra-layer stage learned on mostly nonoverlapping positive data sets, each one came to specialize at predicting more distinct portions of the distribution of the target concept. Algorithm 1 details the training procedure.

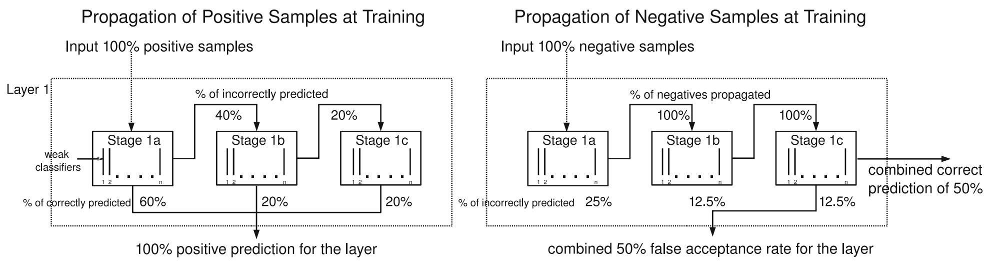
Propagation of Positive Samples at Training
Input 100% positive samples
Layer 1
% of incorrectly predicted
40%
20%
Stage 1a
Stage 1b
Stage 1c
weak classifiers
% of correctly predicted
60%
20%
20%
100% positive prediction for the layer
Propagation of Negative Samples at Training
Input 100% negative samples
% of negatives propagated
100%
100%
Stage 1a
Stage 1b
Stage 1c
combined correct prediction of 50%
% of incorrectly predicted
25%
12.5%
12.5%
combined 50% false acceptance rate for the layer

**Fig. 2** The propagation of positive and negative training samples within each layer of a PSL nested cascade

**Algorithm 1** PSL with Positive Sample Bootstrapping

```
Given: $i$ = cascade layer, $j$ = intra-layer stage, $M$ = total training set size, $m$ =
       training set size where $m < M$, $P$ = total positive set, $N$ = total negative set,
       $neg\_subset_i^M$ = negative training dataset for layer $i$ consisting of $M$ samples,
       $pos\_subset_{ij}^M$ = positive training dataset for layer $i$ and intra-layer stage $j$
       consisting of $M$ samples, $h_k = k^{th}$ weak classifier
Input: $P^M$ = static dataset vector $\{(x_1, y_1), ...(x_M, y_M)\}$ where each sample $x$
       corresponds to the class label $y \equiv 1$, $N^M$ = static dataset vector
       $\{(x_1, y_1), ...(x_M, y_M)\}$ where each sample $x$ corresponds to the class label
       $y \equiv -1$
Output: PSL cascade $classifier_{ij}$

for $i$ = 0 to max cascade layers do
    $neg\_subset_i^m$ = Random_Select_Misclassified_Samples($N$, $m$, $classifier_i$)
    for $j$ = 0 to max intra-layer stages do
        $pos\_subset_{ij}^m$ = Random_Select_Misclassified_Samples($P$, $m$, $classifier_{ij}$)
        for $k$ = 0 to max weak classifiers do
            $h_k$ = Weak_Learn($pos\_subset_{ij}^m$, $neg\_subset_i^m$)
            Boost($pos\_subset_{ij}^m$, $neg\_subset_i^m$, $h_k$)
            Append($h_k$, $classifier_{ij}$)
            if target_hit_rate ≡ Classify($P$, $classifier_{ij}$) then
                break
        if layer_target_hit_rate ≡ Classify($P$, $classifier_{ij}$) then
            break
    if No_Valid_Samples($N$, classifier) then
        break
return classifer
```

Intuitively, the next step involved recognizing that *each* intra-layer stage, made up of a ranging number of weak classifiers, could be seen as an individual ensemble itself. Theoretically, the competence level of each intra-layer stage could then be determined based on its performance on any combination of the training or validation data sets as well as streaming data containing drifting concepts. The process of assigning confidence values to intra-layer stages transformed the nested cascade within each layer into ensembles of ensemble-clusters for which the previous vote-combination strategy had to be revised. The following section describes the procedure for assigning weights to each cluster of ensembles and their combination method, which forms the foundation for our concept-drift-handling system. There we also introduce the idea of learning *layer confidence thresholds*, which form the basis of our proposed algorithm.

## 2.2 Concept-drift learning algorithm

Algorithm 2 outlines in detail our concept-drift learning method. Once an initial dual-cascaded classifier according to the structure of Barczak et al. [27] has been trained on a static data set (Step 1), it is first validated against it to gather competence values for its constituent parts (Step 2). Since each intra-layer stage now becomes interdependent of others in the same layer and is no longer considered to be part of a rigid cascade structure, we refer to each one from now on as an *ensemble-cluster* as seen in Fig. 3.

In the validation process, it is the performance of each individual ensemble-cluster that determines the confidence weight assigned to it. This competence measure is calculated in the form of *alpha* values used by Freund and Schapire [30]:

$$h_{ij}\alpha = 0.5 \ln\left(\frac{1 - error_{ij}^{h}}{error_{ij}^{h}}\right)$$

where each confidence value $\alpha$ is assigned to a $j$th ensemble $h$ on an $i$th cascade layer. Usually, ensemble combination rules consist of a simple *weighted majority vote* as in:

$$H_i(x) = \text{sign}\left(\sum_{j=1}^{n} \alpha_{ij} h_{ij}(x)\right)$$

where $H$ represents a prediction for a layer $i$ using the sum of $n$ number of ensembles applied to on an instance $x$ whose prediction is determined by the sign. We modify this by defining a unique minimum confidence threshold value $thresh_i$ for every layer:

**Algorithm 2** Concept Drift Learning

```
     Given: $\alpha_j$ = vector of sums of confidences $\{z_1,...,z_{2^J}\}$ of ensemble-clusters for layer
            where $z \in \mathbb{R}$, $ec$ = ensemble-cluster classifier, $fp$ = false positive, $tp$ = true
            positive, $\epsilon$ = error rate, $thresh$ = layer threshold value, $\psi$ = false positives
            weight adjustment, $\phi$ = true positives weight adjustment
     Input : $SD_n$ = static dataset vector $\{(x_1,y_1),...(x_n,y_n)\}$ where each sample $x$
            corresponds to a class label $y \in \{-1,1\}$, $D_n^t$ = $t^{th}$ snapshot of non-stationary
            dataset containing $n$ samples,
Step
   1 $ec = Train\_Classifier(SD_n)$
     $Test\_Classifier(SD_n), ec)$            // test ensemble-clusters on static dataset
   2 foreach $i^{th}$ cascade layer do
         foreach $j^{th} ec_i$ do
             $\epsilon_{ij}^{ec} = \frac{1}{2}(fp_{ij} - 1 - tp_{ij})$        // calculate error for ensemble-cluster
             $\epsilon_{ij}^{conf} = \frac{1}{2} ln \frac{1-\epsilon_{ij}^{ec}}{\epsilon_{ij}^{ec}}$        // assign confidence for ensemble-cluster
   3 foreach $t^{th}$ snapshot of dataset stream $D_n^{1,..,T}$ do
         $\epsilon^t = ec(D_n^t)$                          // calculate error on current dataset
   4     if $\epsilon^t > \epsilon$ then
   5         $Adjust\_Initial\_Layer(ec_{ij}(x))$          // error above acceptable threshold
   6         if gradual drift handling effective then goto step 3
   7         foreach $x^{th}$ sample in dataset stream $D_{1,..,n}^t$ do
                 foreach $i^{th}$ cascade layer do
                     $\alpha_{j++} = \sum_{j=1}^{2^J} ec_{ij}(x)$        // insert unique layer alpha value
                     update($thresh_{ij}^{stats}$)        // update stats for layer alpha value
             foreach $i^{th}$ cascade layer do
                 foreach $k^{th}$ distinct $thresh_i$ do
                     $Calculate\_Performance(thresh_{ik}^{stats}, thresh^{tp}, thresh^{fp})$
   8                 $\epsilon_{ik}^{thresh} = \frac{1}{2}((thresh^{fp} * \psi) - 1 - (thresh^{tp} * \phi))$
   9                 if $\epsilon_{ik}^{thresh} < \epsilon^{current\_min\_thresh}$ then make $\epsilon_{ik}^{thresh}$ current min.
```

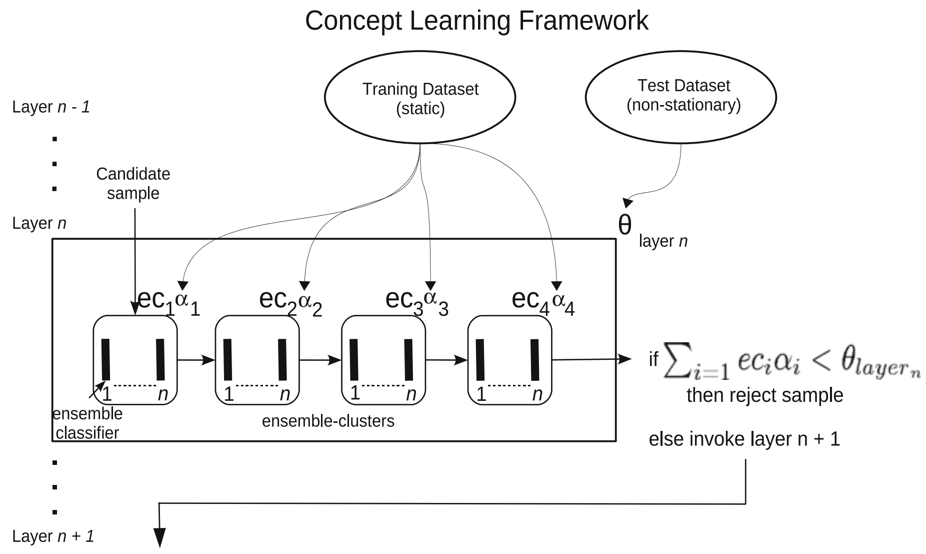

**Fig. 3** Diagram of the concept-drift learning algorithm

$$H_i(x) = \begin{cases} 1 & \text{if } \sum_{j=1}^{n} \alpha_{ij} ec_{ij}(x) > thresh_i \\ 0 & \text{otherwise} \end{cases}$$

in which a sample instance $x$ is positively predicted and passed to succeeding layers for more rigorous testing only if the sum of confidence values for the current layer surpass it thus producing more robust collective decisions.

In contrast to ensemble-cluster confidence weights, the layer confidence thresholds are computed on the incoming data streams from the application domain in which drift is present. In Step 7, an optimal layer threshold $thresh\_conf_i$ for layer $i$ is computed by first calculating all possible sums of ensemble-clusters. By treating each distinct sum as a threshold, a generalization error can then be calculated for each one. Additionally, the confidence threshold value can be set to either favor higher hit rates or lower false positive rates by varying the weights of $\psi$ and $\phi$ values in (Step 8). Once the algorithm has completed classifying all instances from a current data stream and all layer confidence sums with their respective errors have been computed, then the sum with the lowest error rate for each layer is selected as the optimal threshold (Step 9).

Once threshold learning is finished, the classifier is ready to be redeployed and begin handling drift (Step 3). Unlike most ensemble-based methods, our algorithm makes use of a trigger mechanism to inform it that drift is occurring in the environment. In its current form, our algorithm uses the classification error rate as a trigger for drift handling to begin if the generalization ability falls beyond a predefined level (Step 4).

Initial experiments have shown that employing layer confidence thresholds achieves an aggressive strategy for eliminating false positive detections that can sometimes also reduce positive hit rates if not applied optimally. We can combine this observation with the flexibility of cascaded classifiers, which allows us to use a strategy that varies the number of layers that utilize confidence thresholds depending on its current generalization. During runtime, the drift learning algorithm can progressively increase the number of layers to which layer thresholds are applied (Step 5) until a sufficient number of false positives have been eliminated while preceding layers are calculated in original form. The ability of the framework to progressively increase the number of layers that use confidence thresholds becomes the algorithm's facility for handling gradual drifts.

As data streams begin to drift more acutely, the algorithm activates proportionally larger numbers of layer confidences in respect to increasing false positive rates. This proceeds until the drift stabilizes or until all available layers with confidence thresholds are exhausted. If all layer thresholds have been deployed and the error rate is still above an acceptable level, then the layer thresholds have been rendered irrelevant to current conditions, and subsequently, optimal layer threshold learning is reinitiated (Step 6).

## 3 Experiment design

We tested our algorithm on a collection of images containing faces. The comparisons were conducted between our algorithm and the same underlying classifier that was trained on stationary data without concept learning capabilities. The classifier in question was first trained offline using the [29] method with Haar-like features. Five thousand positive faces from a combination of FERET[1] and Yale *Face Database B* [31] data sets were used, against a

[1] http://www.frvt.org/FERET/default.htm

negative data set of 2,000 samples, bootstrapped from a pool of 2,500 negative images. The training and test images used for concept learning were completely independent. The resulting classifier comprised of 24 layers with 3–6 ensemble-clusters per layer. Table 1 summarizes the details of the static classifier and its training procedures.

The test data set used for concept learning consisted of 100 images comprising of 640 $\times$ 480 pixel dimensions, collected from a webcam from different and varying environments. Each frame provided a stream of 336,980 instances in the form of image subwindows that had to be evaluated. The total number of evaluated samples for the data set exceeded 33 million, thus simulating a large volume and high-speed data stream. Initial subwindow size was 24 $\times$ 24 pixels and increased by the factor of 1.2 after exhaustive raster scanning of an image at an increment of 2 pixels per scan. Table 2 lists properties of the test set and the properties of the adaptive learning.

In compiling this data set, we sought to include instances of gradual and abrupt shifts as well as recurring contexts and data distribution changes over time. Gradual shifts were modeled through illumination changes and by panning the camera in a given environment. Sudden environment changes in the form of outdoor and indoor settings were used to model abrupt shifts as were rapid illumination variations. Recurring contexts were modeled by repeating similar images from previously seen settings, while distribution shifts were created by exposing the algorithm to images with and without faces. Example images from our test data set can be seen in Fig. 4.

A maximum of one face per image was present in order to simplify the extraction of the ground truth data concerning face coordinates. The extraction of facial coordinates was assisted by using markers that were used for distinguishing positives from negatives during learning and verification phases. All learning was conducted automatically by the algorithm without manual intervention. The initial frame was programmed to trigger threshold learning irrespective of the offline trained classifier's performance.

In order to ascertain optimal learning durations for our classifier on this domain, five experiments were conducted in which a different number of frames were utilized for each training phase. Initial tests relied on learning from a single frame and extended to the maximum of five consecutive frames. Since accuracy analysis was performed using all images, including the frames that were utilized for learning, experiments that learned multiple frames applied their intermediate learning to their current classifier after each subsequent frame for reasons of fairness. Concept-drift learning occurred on frames succeeding the initial frame whose error rate triggered the learning facility.

Due to the rare-event operating environment of face detectors, higher weights were assigned to positive samples than to the negatives, when layer thresholds were learned. The purpose of this was to assist in maintaining high hit rates; yet, in images where faces were absent, the algorithm was equally capable of learning. The algorithm initiated gradual drift adaptability each time false positives were encountered. Concept learning for abrupt shifts was initiated if a false negative detection occurred or if false positives continued to occur on the lowest cascade layer set for calculating thresholds. In our experiments, using a classifier with 24 layers, the eighth layer was set as the lowest for which layer threshold learning would be performed.

**Table 1** Training and data set details together with the classifier properties for constructing the *static* classifier

| Property | Attribute |
|---|---|
| Positive images | 5,000 |
| Negative images | 2,500 |
| Boosting algorithm | Discrete AdaBoost |
| Cascade layers | 24 |
| Ensemble-clusters per layer | 3–6 |
| Ensemble-cluster size | 1–20 |
| Number of Haar-like feature types | 8 |
| Maximum Haar-like features per image subwindow | 200,000 |
| Minimum pixel area size per Haar-like feature | 16 |

**Table 2** Characteristics of the concept-drift learning data set and the method

| Property | Setting |
|---|---|
| Total images | 100 |
| Image size | 640 $\times$ 480 |
| Subwindow kernel size | 24 $\times$ 24 |
| Subwindow scale increase factor | 1.2 |
| Subwindow raster scan pixel increment | 2 |
| Total subwindows per image | 336,980 |
| Images used for learning per drift phase | 1–5 |
| Subwindow scale increase factor | 1.2 |
| Subwindow raster scan pixel increment | 2 |

## 4 Results

Our analysis of experimental results will first involve accuracy comparisons between the static classifier and the drift learning classifiers. Since face detection is a rare-event detection domain, focus will be on the reduction in the false positive rate. The adaptive classifiers are then compared with each other in order to yield some insight as

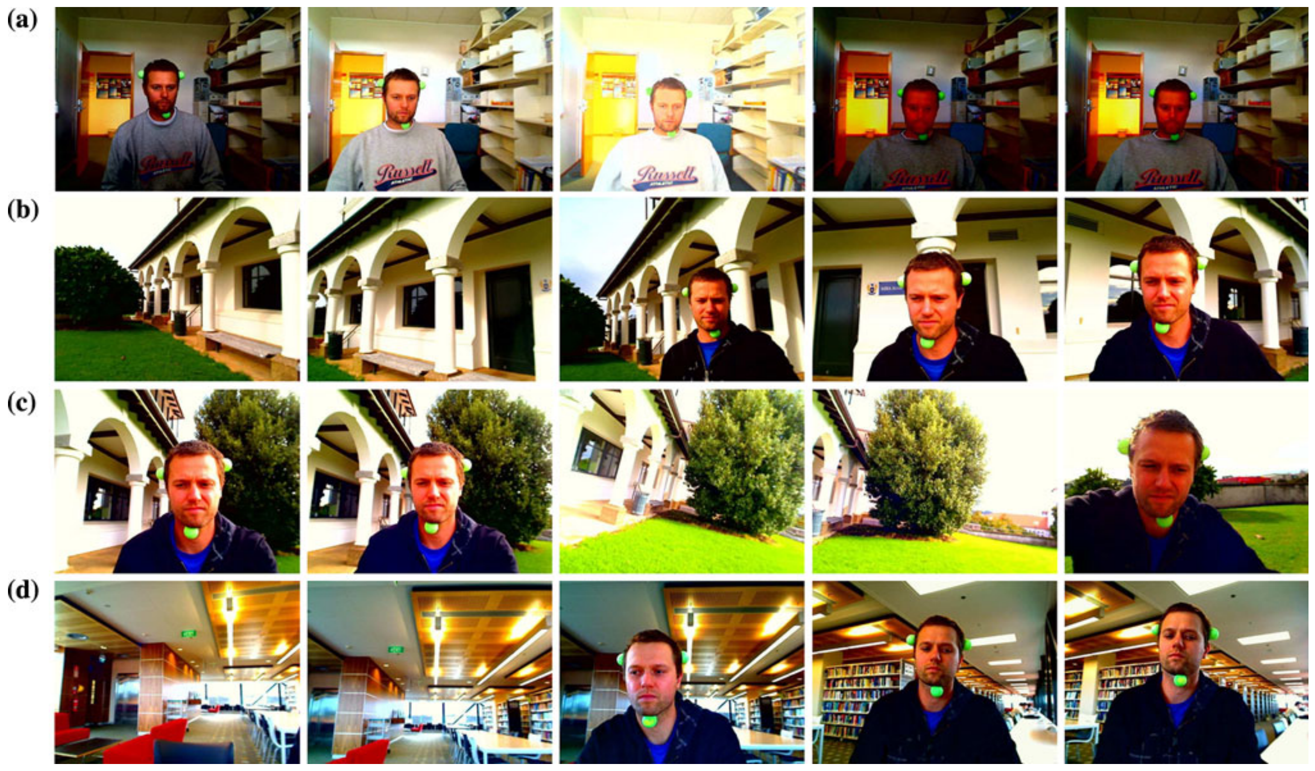

**Fig. 4** Sample test image sequence with various forms of drift. Starting from the *top*, *row* (**a**), example of gradual drifts becoming abrupt due to illumination changes in an indoor environment, *row* (**b**), succeeded by sudden drifts into an outdoor setting, containing both data distribution changes through absences of faces and gradual changes modeled by camera panning into *row* (**c**), concluding with a sudden change to indoors with a combination of the above drifts in row (**d**)

to what might be the optimal learning time-window for this algorithm in similar computer vision domains. The best performing adaptive classifier is selected for more detailed accuracy comparisons with the static classifier in addition to the analysis of runtimes and responsiveness to gradual and abrupt drifts.

## 4.1 Analysis of multi-frame concept-drift learning

The results given in Fig. 5 depict the logarithmic rate of decrease in the total number of false positive detections as a function of each successive cascade layer. At the end of each individual accuracy curve, the total number of false positive detections from each classifier are displayed. From this figure, the static classifier demonstrates the slowest rate of decrease in the false positive detections per cascade layer and ultimately produced the highest number of total false positive detections. All adaptive classifiers attained a reduction in the total number of false positive detections that ranged from 94–99% over those of the static classifier. On this particular test data set, of the five adaptive classifiers, the best accuracy was achieved by drift-learning performed on a window containing a single image frame.

The following observations regarding the speed of responsiveness and adaptation of all classifiers are made

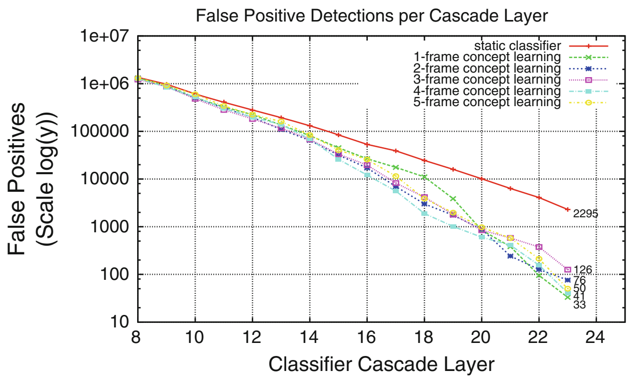

**Fig. 5** Total number of false positive detections per layer for each classifier and learning approach. The total number of false positive detections is shown at the end of each curve

from Fig. 6. This figure approximates the actual runtimes through curve smoothing for the purpose of readability and highlighting overall trends. As expected, the figure points to generally longer runtimes for adaptive learning that takes place on larger windows. Learning on multiple snapshots of data proved to be particularly more expensive between frames 50–75. These frames correspond with images that presented both patterns of increased difficulty to classify, as

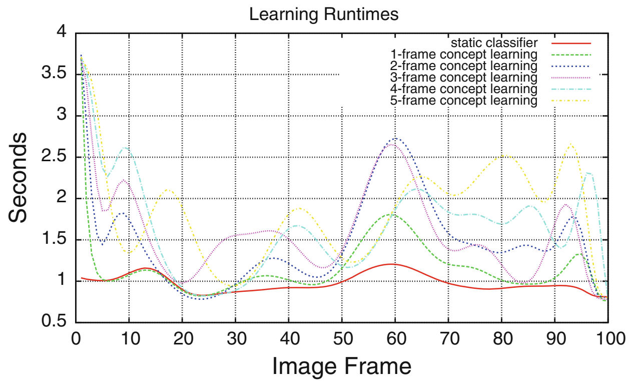

**Fig. 6** Execution runtimes for all classifiers displayed in approximated line curves that highlight general trends

well as a sequence of more frequent abrupt drifts that resulted in a high concentration of explicit threshold learning.

The total number of explicit learning phases did not vary significantly between the five adaptive classifiers, with all classifiers triggering 7–8 learning phases. From this, we can conclude that learning exposures on larger snapshots of incoming data did not result in an overall decrease in runtimes that might have been expected had increased accuracy occurred since fewer explicit learning phases would have been initiated. On the contrary, both the accuracy of classifiers deteriorated when multiple snapshots of data were employed, and a vulnerability in runtimes was observed when using multiple data snapshots in the presence of frequent sudden drifts. On this particular domain with this pattern and magnitude of environmental change, the algorithm has demonstrated an ability to rapidly adapt to drifting conditions using minimal data. The subsequent analysis examines in more detail the characteristics of the single-frame adaptive classifier and compares it with the static classifier.

## 4.2 Results of single-frame concept-drift learning

Given in Fig. 7a, are comparison results of the total number of false positive detections per frame between the static and the adaptive classifiers. The figure shows that the false positive detections for the static classifier seldom stayed at zero, while the upper range exceeded 50 false detections per frame. By applying the drift-learning algorithm, the false positive detections were removed from all but 18% of the total images, while their total per image never exceeded two false positive detections. In total, 33 false positive detections belonging to the single-frame adaptive classifier were confined to only 18 images, while the distribution of 2,295 false positive detections of the static classifier for all practical purposes rendered most frames unusable. In regard to the overall hit rates, the reduction in false positive detections came at a small price. The static classifier

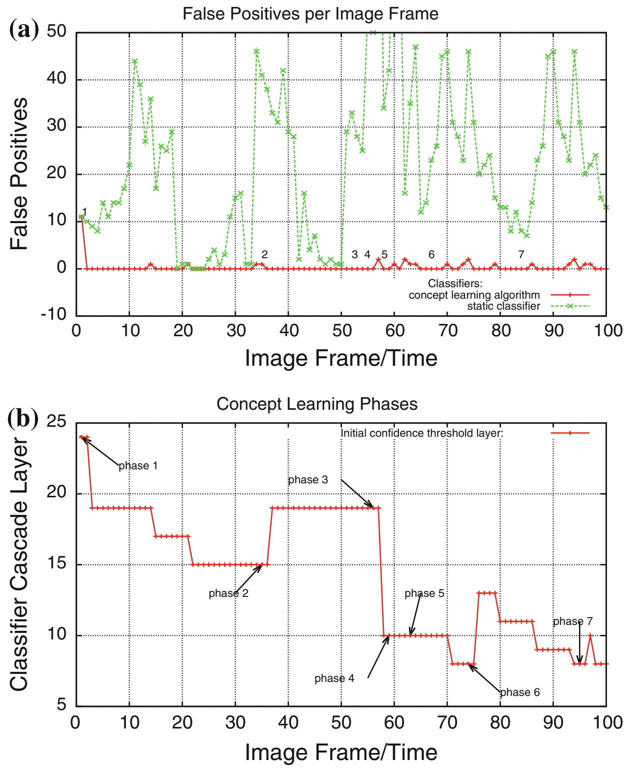

**Fig. 7** **a** Total number of false positive detections per frame for a single-frame learning classifier. Positions of the learning phases from 1–7. **b** The pattern of the learning algorithm's adaptation to gradual drifts for each successive frame. As detection difficulty increases, the algorithm increases the number of layers for which threshold confidences are used by selecting lower layers as initial starting points

correctly classified all positive samples while our drift-handling algorithm produced three false negative detections.

Recurring contexts were primarily simulated in the first 50 frames and made more challenging by inserting in the middle an abrupt drift in order to ascertain if concept forgetting would ensue. From the same figure, we can observe very low false positive detections occurring for the adaptive classifier throughout this phase and affirm that there is robustness in this algorithm to cyclical environmental shifts.

## 4.3 Response patterns to gradual and abrupt drifts

Figure 7b demonstrates the adaptation pattern of our algorithm to gradual concept drifts per frame. More specifically, it shows how out of the total 24 layers of the cascaded classifier, an increasing number of layer thresholds were employed for classification as the accuracy deteriorated. This is seen in the figure in the manner which the initial cascade layer for calculating layer thresholds decreases in response to increasing false positive detections. This figure is complemented by Fig. 7a referred to

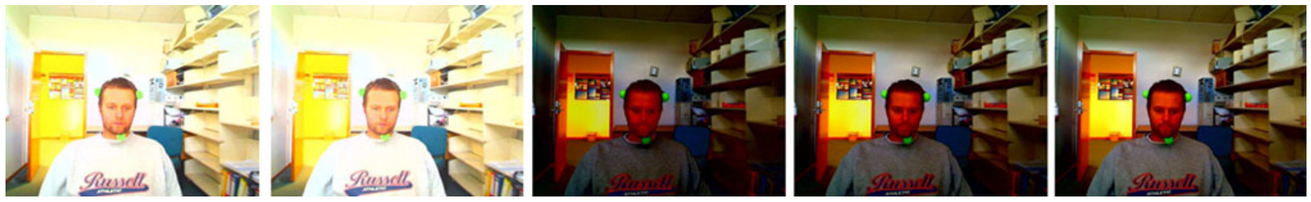

**Fig. 8** An example of an image sequence containing an abrupt drift due to a significant illumination change, that subsequently triggered a concept-drift learning phase

earlier, by showing the magnitude of false positive detections that initiate the gradual concept-drift-handling facility. By noting the removal of subsequent false positive detections in Fig. 7a, we can see the effectiveness of the algorithm to handle gradual drifts.

The high concentration of abrupt concept drifts that took place between frames 50–75, as mentioned earlier, is also pointed out in Fig. 7b by phases 3–6. These transitions caused higher error rates to occur, resulting in the initiation of four layer threshold learning phases in short succession. Despite this, the algorithm demonstrated robustness in its responsiveness to frequent sudden drifts in data. This can be observed in the decrease and rapid stabilization of the false positive detections after abrupt drifts have taken place. Figure 8 shows an example of a sequence of images containing an abrupt drift that was simulated through an extreme illumination change, which activated a layer threshold learning phase.

### 4.4 Learning and detection runtimes

The runtime cost of adapting to drift amounts to 3–4 times the classification time of the static classifier (Fig. 9a). In spite of this, there is sufficient scope for optimizing the learning phase through parallelization that would result in an overall marginal performance penalty for time critical applications. We believe that for most applications, the modest detection time increase is likely to be acceptable in exchange for a considerable improvement in accuracy.

Contrary to our expectations, we also note that the detection runtimes of our adaptive classifiers using layer thresholds are consistently faster on images not being learned, than that of the static classifier (Fig. 9b). The detection runtime acceleration of up to 10% is attributed to the fact that the learned layer thresholds correctly classify more negative samples in earlier cascade layers, thus preventing their further propagation to subsequent layers and in the process decreasing computation costs. The faster detection runtime of our adaptive classifiers mitigates, to some degree, the increase in runtimes brought on by the learning phases of layer thresholds.

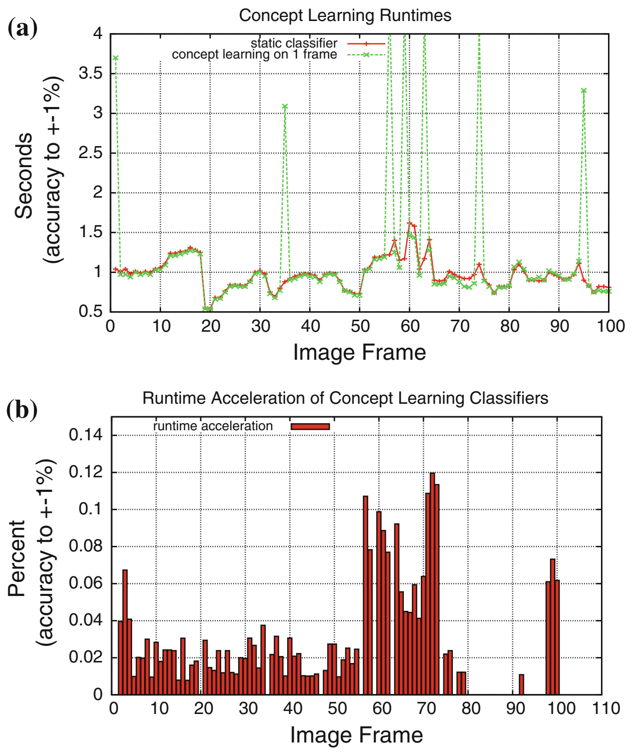

**Fig. 9** **a** An accurate comparison of runtimes between the single-frame learning classifier and the static classifier. **b** The percentage of increased execution runtime of the single-frame learning classifier over the static classifier

## 5 Discussion

Unlike most ensemble-based solutions, the strength of the proposed approach is that individual ensemble-clusters are *semantically aware*. According to the definition of Wang et al. [2], systems without this characteristic are unable to correspond individual weak classifiers with a hidden concept. These results in costly global-updates of ensembles, which makes the methods vulnerable to high-speed data streams. Our approach possesses this attribute, since each ensemble-cluster is responsible for independently predicting a subset of samples. Additionally, our method is semantically aware since the ensemble-clusters are entirely revisable without affecting other components. They can also be removed, while new ones may be appended during runtime to allow them to be explicitly associated with emerging hidden concepts.

The ability of the approach to rapidly adapt in large data streams is a noteworthy property. Since only layer thresholds are learned and updated, the algorithm achieves a unique balance between stability and plasticity in learning. It is strongly robust to recurring contexts, while still being sufficiently responsive to changes in its operating environment. The method has proved itself particularly effective in rare-event operating domains where extremely low false positive detections are paramount. Its capacity to improve accuracies of poorly performing classifiers has implications that may simplify and accelerate offline training processes.

Lastly, the algorithm is also capable of adapting without deterioration when only exposed to negative samples. This makes it suitable for background modeling. In all cases, effective adaptation is achieved with minimal data and the detection runtime has been shown to accelerate when the learned thresholds are utilized.

The shortcomings of the algorithm lie in its current inability to explicitly learn novel positive samples or positives whose class description has been radically modified. The modularity of the framework does, however, enable it to address this by augmenting each cascade layer with new ensemble-clusters. Consequently, although our approach is restricted to detecting target concepts that do not alter substantially, the effect is that the algorithm becomes resistant to incorrectly learning and modeling noise.

In its present form, our algorithm is also confined to binary-class problems. Nevertheless, since all multi-class problems can be reformulated as series of two-class classification tasks [32, 33], our approach can be applied to each resulting classifier. An additional challenge in our system is the difficulty in determining the optimal balance of responsiveness to concept drift between gradual adaptation and explicit relearning of layer-thresholds. A poor configuration will either result in increased classification errors prior to adaptive learning being initiated or a performance degradation due to constant layer-threshold updates.

Finally, as is the case in all supervised learning, our approach is reliant on explicit input of ground truth describing class labels of new instances. However, this disadvantage can be alleviated by our algorithm to some degree in rare-event domains, by periodically exposing the learning algorithm to negative samples and thereby performing regular background modeling as discussed earlier.

## 6 Conclusion

The main contribution of this paper was the introduction of an adaptive learning algorithm for nonstationary environments that is designed for cascades of boosted ensembles. Its novelty lies in learning optimal cascade layer thresholds in order to achieve adaptability, without resorting to the altering of confidence weights of individual classifiers. Within each layer, individual classifiers are amalgamated into ensemble-clusters whose collective decisions are assigned competence weights based on their performance on training data. During runtime, a classification is attained by combining the decisions of all ensemble-clusters within each layer in order to produce a collective value. The comparison of this value with the learned layer thresholds formulates the final classification.

A foremost aim of the research was to achieve real-time learning adaptability in problem domains where explicit training of new ensembles is computationally infeasible due to large volumes of streaming data as well as data that are represented by immense dimensionalities. The goal was to replicate such an operating environment and to demonstrate the ability of this algorithm to adapt to diverse forms of drift without inducing concept forgetting.

In our experiments, we showed the ability of our system to achieve real-time drift learning for a binary-class problem. We simulated an operating environment with a large datastream by applying our algorithm to face detection and found that it was timely and effective in removing false positive detections. The method was responsive to both gradual and abrupt concept drifts, and we found that classifiers whose accuracy in time-evolving environments degraded beyond usability, became practical by applying our algorithm. Additionally, our approach did not suffer from concept forgetting as found in recurring contexts, thus creating a balance between both plasticity and stability of learning that is sustainable over time.

Subsequent research will focus on applying this algorithm to other problem domains as well as extending its learning capabilities that will enable it to also become incremental in nature by integrating novel target concept information into its ensemble.

## References

1. Muhlbaier M, Polikar R (2007) Multiple classifiers based incremental learning algorithm for learning in nonstationary environments. In: 2007 international conference on machine learning and cybernetics, vol 6. pp 3618–3623
2. Wang P, Wang H, Wu X, Wang W, Shi B (2007) A low-granularity classifier for data streams with concept drifts and biased class distribution. IEEE Trans Knowl Data Eng 19:1202–1213
3. May M, Berendt B, Cornuejols A, Gama J, Giannotti F, Hotho A, Malerba D, Menasalvas E, Morik K, Pedersen R et al (2008) Research challenges in ubiquitous knowledge discovery. Chapman & Hall/CRC Press, London

4. May M, Saitta L (2010) Introduction: the challenge of ubiquitous knowledge discovery. In: May M, Saitta L (eds) Ubiquitous knowledge discovery. Volume 6202 of lecture notes in computer science. Springer, Berlin, pp 3–18
5. Widmer G, Kubat M (1996) Learning in the presence of concept drift and hidden contexts. Mach Learn 23:69–101
6. Schlimmer J, Granger R (1986) Incremental learning from noisy data. Mach Learn 1(3):317–354
7. Kuncheva LI (2004) Classifier ensembles for changing environments. In: Roli F, Kittler J, Windeatt T (eds) Multiple classifier systems. Volume 3077 of lecture notes in computer science. Springer, Berlin, pp 1–15
8. Tsymbal A (2004) The problem of concept drift: definitions and related work. TCD-CS-2004-15, vol 4. Department of Computer Science, Trinity College, Dublin
9. Nishida K, Yamauchi K, Omori T (2005) Ace: adaptive classifiers-ensemble system for concept-drifting environments. Mult Classif Syst 3541:176–185
10. Japkowicz N (2000) The class imbalance problem: significance and strategies. In: Proceedings of the 2000 international conference on artificial intelligence (ICAI 2000), vol 1. pp 111–117
11. Zhu X, Zhang P, Lin X, Shi Y (2010) Active learning from stream data using optimal weight classifier ensemble. IEEE Trans Syst Man Cybern B Cybern 40:1607–1621
12. Procopio M, Mulligan J, Grudic G (2009) Learning terrain segmentation with classifier ensembles for autonomous robot navigation in unstructured environments. J Field Robot 26(2):145–175
13. Street W, Kim Y (2001) A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the seventh ACM SIGKDD international conference on knowledge discovery and data mining. KDD '01. ACM, New York, pp 377–382
14. Wang H, Fan W, Yu P, Han J (2003) Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the ninth ACM SIGKDD international conference on knowledge discovery and data mining. KDD '03. ACM, New York, pp 226–235
15. Scholz M, Klinkenberg R (2007) Boosting classifiers for drifting concepts. Intell Data Anal 11(1):3–28
16. Elwell R, Polikar R (2009) Incremental learning of variable rate concept drift. Mult Classif Syst 5519:142–151
17. Rodriguez J, Kuncheva L (2010) Combining online classification approaches for changing environments. Struct Syntactic Stat Pattern Recognit 5342:520–529
18. Karnick M, Muhlbaier M, Polikar R (2008) Incremental learning in non-stationary environments with concept drift using a multiple classifier based approach. In: 19th international conference on pattern recognition, 2008 (ICPR 2008). pp 1–4
19. Kolter J, Maloof M (2003) Dynamic weighted majority: a new ensemble method for tracking concept drift. In: Third IEEE international conference on data mining, 2003 (ICDM 2003). pp 123–130
20. Kolter J, Maloof M (2007) Dynamic weighted majority: an ensemble method for drifting concepts. J Mach Learn Res 8:2755–2790
21. Pocock A, Yiapanis P, Singer J, Luján M, Brown G (2010) Online non-stationary boosting. Mult Classif Syst 5997:205–214
22. Oza N (2001) Online ensemble learning. PhD thesis, University of California, Berkeley
23. Huang C, Ai H, Yamashita T, Lao S, Kawade M (2007) Incremental learning of boosted face detector. In: IEEE 11th international conference on computer vision, 2007 (ICCV 2007). pp 1–8
24. Pelossof R, Jones M, Vovsha I, Rudin C (2009) Online coordinate boosting. In: IEEE 12th international conference on computer vision workshops (ICCV workshops), 2009. pp 1354–1361
25. Grabner H, Leistner C, Bischof H (2008) Semi-supervised on-line boosting for robust tracking. In: Forsyth D, Torr P, Zisserman A (eds) Computer vision—ECCV 2008. Volume 5302 of lecture notes in computer science. Springer, Berlin, pp 234–247
26. Bifet A, Holmes G, Pfahringer B, Kirkby R, Gavaldà R (2009) New ensemble methods for evolving data streams. In: Proceedings of the 15th ACM SIGKDD international conference on knowledge discovery and data mining. KDD '09. ACM, New York, pp 139–148
27. Barczak ALC, Johnson MJ, Messom CH (2008) Empirical evaluation of a new structure for adaboost. In: SAC '08: proceedings of the 2008 ACM symposium on applied computing. ACM, Fortaleza, pp 1764–1765
28. Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In: CVPR01, Kauai, HI, IEEE (December), vol I. pp 511–518
29. Susnjak T, Barczak A, Hawick K (2010) A modular approach to training cascades of boosted ensembles. In: Structural, syntactic, and statistical pattern recognition. Volume 6218 of lecture notes in computer science. Springer, Berlin, pp 640–649
30. Freund Y, Schapire RE (1995) A decision-theoretic generalization of on-line learning and an application to boosting. In: EuroCOLT '95: proceedings of the second European conference on computational learning theory. Springer, London, pp 23–37
31. Georghiades A, Belhumeur P, Kriegman D (2001) From few to many: illumination cone models for face recognition under variable lighting and pose. IEEE Trans Pattern Anal Mach Intell 23(6):643–660
32. Allwein EL, Schapire RE, Singer Y (2001) Reducing multiclass to binary: a unifying approach for margin classifiers. J Mach Learn Res 1:113–141
33. Lorena AC, Carvalho AC, Gama JAM (2008) A review on the combination of binary classifiers in multiclass problems. Artif Intell Rev 30:19–37