Int. J. Mach. Learn. & Cyber. (2018) 9:37–61
https://doi.org/10.1007/s13042-015-0333-x

ORIGINAL ARTICLE

A novel online ensemble approach to handle concept drifting data streams: diversified dynamic weighted majority

Parneeta Sidhu · M. P. S. Bhatia

Received: 15 July 2014 / Accepted: 16 January 2015 / Published online: 31 January 2015

Abstract We present an online ensemble approach, diversified dynamic weighted majority (DDWM) to classify new data instances which have varying conceptual distributions. Our approach maintains two sets of weighted ensembles that differentiate in their level of diversity. An expert in either of the ensembles is updated or removed as per its classification accuracy and a new expert is added based on the final global prediction of the algorithm and the global prediction of the ensemble for any data instance. Experimental evaluation using various artificial and real-world datasets proves that DDWM provides very high accuracy in classifying new data instances, irrespective of size of dataset, type of drift or presence of noise. We compare DDWM with the other learners in terms of new performance metrics such as kappa statistic, model cost, and the evaluation time and memory requirements. Our approach proved to be highly resource effective achieving very high accuracies even in a resource constrained environment.

Keywords Concept drift · Ensemble · Diversity · Data stream · Online learning

1 Introduction

Data stream mining is a very important research area in machine learning community. It is the process of studying the concept underlying the data and the variations in that concept to classify new data instances with higher accuracy. Data streams differ from the static databases as they may have varying concepts underlying the data, unlimited size, high speed and high dimensionality [52]. We can access a data instance in a data stream only "once" when it arrives, after that the given instance is replaced by a new instance which may have a different conceptual distribution.

'Concept' for a data instance refers to the underlying data distribution, illustrated by the joint distribution [1], $p(\mathrm{x}, \mathrm{y})$ where x represents the n-dimensional feature vector and y represents its class label. The term 'concept drift' refers to change in the underlying conceptual distribution [6, 7, 15] as new instances arrive for example in various applications like Market-Basket analysis [10], computer security, internet data, credit fraud detection, bioinformatics etc. In Market-Basket analysis, similar concept is seen in the customer buying behavior each year during Christmas festivity. This pattern re-occurs every year (i.e. recurrent drift), resulting in a drift from the customer's last month buying pattern. A drift present in a dataset is measured by its severity and speed. Severity represents the amount of changes caused by a new concept. Speed is the inverse of the time taken for a new concept to completely replace the old concept.

The approaches to handle drifts could be classified as either incremental approaches [7, 22, 25, 40–45] or online approaches [1, 4, 6, 7, 10, 11, 18, 19, 24, 54]. An incremental algorithm [46] processes data several times in batches of considerably large size. Whereas, the online approach process each training example only once when it arrives, without any need for storage or reprocessing. A current hypothesis is maintained that reflects all the training instances so far. The online approaches can further be

P. Sidhu (✉) · M. P. S. Bhatia
Division of CoE, Netaji Subhas Institute of Technology, Sec-3 Dwarka, New Delhi 110078, India
e-mail: parneeta07@gmail.com

M. P. S. Bhatia
e-mail: bhatia.mps@gmail.com

categorized as: approaches that use a mechanism to handle concept drift [1, 6, 7, 18, 19] and those that do not explicitly use a mechanism to detect drifts [4, 10, 11, 24]. The former approaches use some measure related to the classification accuracy in handling drifts. They rebuild the system once a drift is detected. The latter type of online approaches assigns weights to each base learner as per their accuracy in classifying the new data instances. These approaches update the experts, adding newly learnt experts upon detection of a drift.

There are two main models for handling classification in data streams: single classifier incremental model and the ensemble model. In single classifier model, the classifier is incrementally updated to cope with drifting data distributions requiring complex operations. The ensemble model uses a set of classifiers and trains each classifier with the intention to create an improved model. The result of prediction is the combined result achieved in a way (weighted or un-weighted voting or by maximum value) in classifying the new instances. However, for better performance the ensembles need to be pruned such that they maintain a set of diverse and accurate classifiers [56]. Analysis of the experimental results done in the past, prove that an ensemble model has higher generalization accuracy [58, 59] and the success of an ensemble is based on the level of diversity between the classifiers. Diversity [29] in an ensemble of experts (classifiers) is the measure of variation in the classification accuracy of ensemble members for a given training example. Diversity [57] can be created among the ensemble members by one of the following ways: data manipulation method, homogeneous method or heterogeneous (hybrid) method. However, the diversity in the ensemble cannot be treated as a replacement for the accuracy of the system.

This paper provides an empirical evaluation of our approach diversified dynamic weighted majority (DDWM) that maintains two ensembles with different levels of diversity, so as to accurately handle various types of concept drifts. In all the experimental comparisons, our approach performed better or at least similar to the other approaches under variations in speed of drift, severity of drift, presence of noise and varying size of datasets. Section 2 explains the various research questions that have been answered by our research paper.

2 Research questions and paper organization

Our work aims at answering the following research questions related to the work on concept drift:

1. How can the concept of diversity be applied on an online approach that does not explicitly use a mechanism to detect drifts [10, 12]? Earlier work on concept drift [16] used the concept of diversity with the online approaches that use a mechanism to handle drifts.
2. Does the high diversity ensemble always provide better classification accuracy than the low diversity ensemble when handling drifting concepts? The study proves that the classification accuracy of each ensemble in handling any drifting dataset depends on the speed and severity of drift present in the dataset.
3. How the level of diversity among the experts in an ensemble impacts its performance in terms of various performance metrics: prequential accuracy, kappa statistic, model cost, time and memory?
4. How does diversity helps in improving the performance of an ensemble system in handling various types of drifting concepts? The work presented in [3, 10, 12, 13] shows the performance of a system maintaining an ensemble of experts without using the concept of diversity.
5. What is the impact of presence of noise in the dataset, on the performance of a concept drifting system in terms of various performance metrics? The work presented in [10, 12] showed the impact of noise on the system performance in terms of accuracy and expert count only.

In order to answer the first research question, we propose a new online ensemble learning approach to handle drifting concepts called diversified dynamic weighted majority (DDWM). It is an online ensemble approach that does not explicitly use a mechanism to detect drifts. It maintains weighted ensembles with different diversity levels among its experts. The experts would be updated as per their classification accuracy in handling new data instances, as in dynamic weighted majority (DWM) [10, 12] approach. The two ensembles with different levels of diversity are: an ensemble with low diversity and an ensemble with high diversity. The final prediction is the class having the maximum support, involving the weighted majority vote of the experts' predictions from each of the ensembles. We use two different ensembles varying in their diversity levels as the two ensembles would perform with different classification accuracies, depending upon the severity and speed of drift present in the dataset. Empirical analysis would prove that our diversified approach provides very good accuracy in handling drifts in data streams, using various artificial and real-world datasets with different types of drifts, irrespective of noise present in the dataset.

In order to answer the second and the third research questions, we perform a study of the various performance metrics using high diversity ensemble and low diversity ensemble and analyze their comparative performance while handling datasets containing abrupt concept drifts. The analysis identifies that for high speed drifts, the low diversity ensemble provides better prequential accuracy than the high diversity ensemble. The low diversity ensemble helps DDWM achieve better accuracy than early drift detection method (EDDM) [1] and DWM. However, with the progress in learning the experts in the high diversity ensemble also achieve an improved adaptation rate to the new concepts. The average evaluation time and the memory requirements of both the ensembles are almost the same, independent of the level of diversity among its experts. Hence, the model cost is also the same for both the ensembles. The low diversity ensemble in DDWM has very high homogeneity/low diversity (i.e. high value of kappa statistic) among its experts (as its name suggests) whereas the high diversity ensemble has very high level of diversity (i.e. low value of kappa statistic) among its experts. Hence, the value of kappa statistic is dependent on the λ value used in modified version [15] of online bagging [21] to explicitly encourage more or less diversity in an ensemble.

In order to perfectly answer the fourth research question, empirical evaluation of our approach was done using various artificial and real-time datasets containing different levels of severity and speed of drift and random noise in some cases. The analysis identifies that while handling abrupt concept drifts irrespective of presence of noise our approach converged very quickly to the target concepts achieving accuracies better than dynamic weighted majority (DWM) and Blum's weighted majority (WM) [3], independent of any base classifier. When datasets contain gradual drifts and noise, our approach responds quickly to changes in concepts, achieving very high accuracies same as WM and better than DWM approach. Evaluation of DDWM using real time datasets identifies that our approach provided a highly stable system, achieving almost similar or better accuracy than DWM.

In order to answer the last question, we compared the performance of our approach in a noisy and a non-noisy domain. For datasets containing abrupt drifts and noise, DDWM shows high sensitivity to noise even when the base algorithm does not support noise. The results led us to state that the presence of noise creates a negative impact on the performance of the system. The system suffers from an increase in the average evaluation time and increase in the model cost, with decrease in prequential accuracy and decrease in kappa statistic value when in a noisy domain, as compared to its performance in a non-noisy domain. However, there is no change in the average memory requirements of our concept drifting system. The comparative evaluation of DDWM to the other approaches in a noisy domain states that our approach performs better/similar as the other approaches in terms of average prequential accuracy. For datasets containing gradual drifts and noise, our approach provides a highly stable system with very good average prequential accuracy irrespective of presence of noise in the dataset.

This paper is further organized as follows. Section 3 presents the related work on online concept drifting approaches and provides us with the background knowledge of the concept of diversity. In this section, we also describe the various performance metrics used to evaluate the approaches. In Sect. 4, we will give a brief description of the various artificial and real-world datasets used for empirical evaluation of our approach. Section 5, presents the DDWM algorithm and its detailed description. In Sect. 6, we present the analysis, evaluation and a detailed comparison of our approach with the existing approaches. In the end, we summarize our paper and discuss directions for future research.

3 Related work

3.1 Online concept drifting approaches to handle drifting concepts

The first drift handling ensemble learning system was STAGGER [26] that applies a set of heuristics guided by the Bayesian evaluation measures to prune an established characterization that proves in-effective and/or add a new generated characterization element whose weight surpasses the threshold. The FLORA [28] system induces current target concept from a single window of recent examples by incrementally learning a new instance and forgetting the least recent one within its window [27]. Weighted majority (WM) [3, 13] is an online ensemble approach that maintains set of weighted experts. It believes in the principle that all not features of a given data instance are necessary to make a prediction. Drift detection method (DDM) [6], controls the online error-rate (number of errors) produced during prediction whereas Early Drift detection method (EDDM) [1] was based on the estimated distribution of the distances between classification errors.

Adaptive classifier ensemble (ACE) [20] is an adaptive classifier ensemble approach that uses an online classifier, a set of batch classifiers, and a drift detection mechanism to handle mainly recurrent drifts. To improve the classification accuracy further, an enhanced version of ACE

[18] that used a pruning strategy for classifiers. Detection with statistical test of equal proportions (STEPD) [19] uses a single online classifier that compares the classification accuracies: the overall accuracy from the beginning of the learning, and the accuracy of recent examples after concept drift, by using statistical test of equal proportions. Addictive Expert Ensembles (AddExp) [11] adds a new classifier whenever the system output is incorrect, and used two pruning methods: the oldest first and the weakest first. Two online classifiers for learning and detecting concept drift (Todi) [17] reduced the impact of false alarms on the accuracy of the system in classifying the new instances. Dynamic weighted majority (DWM) [10, 12] is a modified version of WM [13]. DWM creates new experts dynamically in response to changes in its performance, and removes an expert if its weight reaches a threshold value.

CDR-Tree [37] mined concept drifting rules in data streams by integrating new and old instances from different time scales into pairs. It extracted a classification model of each data block for decision making. STREAM-DETECT [38] detected changes in data streams by measuring on-line clustering result deviation over time. RePro [39] was another concept drifting approach that incorporated reactive and pro-active predictions. AC–DC algorithm [34] was based on association rules to find relationships between item-sets and extract the set of frequently occurring patterns from the input dataset. Learn ++. NSE [55] is an incremental ensemble approach for incremental learning of concept drift, where the classifiers were combined using a dynamically weighted majority voting technique based on each classifier's time-adjusted accuracy rate on current and past instances. Diversity for dealing with drifts (DDD) [16] used the concept of varying diversity levels between ensembles. After the detection of drift, new high diversity and low diversity ensembles were created in addition to already existing high diversity and low diversity ensembles.

3.2 Concept of diversity in an ensemble of experts

The literature [15, 29, 30, 53] study on diversity states that diversity in an ensemble reduces the initial increase in error caused by a drift. Secondly an ensemble delivers more reliable results if it includes diversity between its experts. Based on an analysis of various diversity measures, Yule's Q statistic [29, 31] was recommended to be the best measure for diversity. It minimizes the error of ensembles and is very easy to interpret. Different levels of diversity in an ensemble was explicitly introduced by varying the λ value in a modified version of online bagging [15, 21], where Poisson (1) distribution was replaced by Poisson (λ) distribution. Higher/lower values of λ are associated with lower/higher diversity (higher/lower Q-average) in an ensemble of experts.

The performance of an ensemble with high diversity differs from the performance of the low diversity ensemble in classifying drifting data instances. The accuracy with which each ensemble classifies the instances depends on the level of severity and speed of drift present in the dataset. From earlier research work on concept drift in DDD [16], it has been found that for drifts with low severity and high speed, the old high diversity ensemble provides the best generalization accuracy. For high severity and high speed drifts, it is a good strategy to use the new low diversity ensemble. For low speed drifts (independent of severity), old high diversity ensemble provides the best accuracy during the whole period since the beginning of the drift.

If the dataset contains drifts with low severity and high speed, even though the new concept after the drift would not be same but it would be similar to the old concept. In this case the high diversity ensemble learns the old concept only partly, and is able to converge to the new concept by learning it with low diversity. A low diversity ensemble would provide information from the old concept but would have problems to converge to the new concept [15].

If the dataset contains drift with high severity and high speed [16], it causes big changes very suddenly. The new concept is completely different from the old concept. The new ensemble learning from scratch is thus the best option. For low speed drifts (independent of severity), the new concept gradually replaces the old concept. So only shortly after the beginning of the drift, the best strategy is to use the low diversity ensemble as the conceptual distribution underlying the data instances is almost the same; but longer after the drift independent of severity, the high diversity ensemble learning the new concept with low diversity provides better classification accuracy.

3.3 Performance evaluation metrics for various concept drifting approaches

– Prequential Accuracy (%): prequential accuracy [1, 4, 32] is the average online accuracy obtained by prediction of each example to be learned, prior to its learning.
– Kappa Statistic (%): the Kappa statistic value is a performance measure that gives a score of homogeneity among the experts.
– Model cost (RAM-Hours): the resource efficiency of any stream mining algorithm is judged by another

measure i.e. model-cost (RAM-Hours). One RAM-Hour is equivalent to one GB of RAM being deployed for one hour.

- Time (CPU-seconds): it is the total CPU runtime involved when testing of new instances and further training of experts.
- Memory (bytes): it is the total amount of memory used by any online algorithm. It consists of memory used to store the running statistics, and that used to store the online model.

4 Concept drifting data streams

4.1 Artificial datasets

4.1.1 *Stagger concepts: Abrupt concept drift, without noise*

A concept in a Stagger [26, 54] dataset consists of three attribute values: *shape* ∈ {circle, triangle, rectangle}, *color* ∈ {green, blue, red}, and *size* ∈ {small, medium, large}. The dataset contains 240 training examples, with one example at each time step. Here, we are evaluating a classifier based on maximum of a pair of features (i.e. two features) and in each context at least one of the features is irrelevant.

4.1.2 *SEA Concepts: Very large dataset, abrupt concept drift with noise*

The SEA [25] dataset provides a very large sized dataset with each training example consisting of three real-valued attributes, $x_i \in \mathbb{R}$ such that $0.0 \le x_i \le 10.0$.The target concept is as in Eq. (1).

$$Y = [x_0 + x_1 \le \theta] \tag{1}$$

where $\theta \in \{7, 8, 9, 9.5\}$ for the four data blocks. A data instance belongs to class 1 if condition in Eq. (1) is correct and class 0 otherwise. We can observe that only the first two attributes (x_0, x_1) are relevant. There are 50,000 training examples with one example at each time step. Noise was introduced to evaluate our approach in a noisy domain.

4.1.3 *Moving hyperplane: gradual drift with noise*

The training examples in this dataset [9] are uniformly distributed in multidimensional space $[0, 1]^{10}$ and the examples are labeled as positive if they satisfy the condition as in Eq. (2)

$$a_0 \le \Sigma_{i=1}^{10} a_i x_i \tag{2}$$

The weights of the hyperplane, $\{a_i\}$, are initialized to $[-1, 1]$ randomly, and are updated as $a_i \leftarrow a_i + 0.001 s_i$ at each time step, where $s_i \in \{-1, 1\}$ represents direction of change for each weight. The threshold a_0 is calculated as in Eq. (3), at each time step.

$$a_0 = {}^1\!/_2\, \Sigma_{i=1}^{10} a_i \tag{3}$$

$\{s_i\}$ is reset randomly after every 1000 training examples. Random noise was introduced to evaluate our approach in noisy domain. The total number of examples was 3,000.

4.2 Real-world datasets

4.2.1 *Electricity pricing domain*

To evaluate our approach on a real-world problem we selected, “the electricity pricing domain [8]”. This dataset was obtained from TransGrid, the electricity supplier in New South Wales, Australia. The dataset has a total of 45,312 instances collected at 30-min intervals between 7 May 1996 and 5 December, 1998. Each instance has a feature vector consisting of five features and a class label of either up or down. The prediction task is to predict whether the price of electricity will go up or down and is affected by demand and supply.

4.2.2 *Breast cancer dataset*

“The Breast cancer dataset [36] “classifies an instance as recurrence-events or no-recurrence-events based on the various attribute values. This dataset was provided by the Oncology Institute, and has repeatedly appeared in the machine learning literature. The data set includes total of 286 instances out of which 201 instances belong to one class and the rest 85 instances belong to another class. Each data instance consists of 9 attributes, having either linear or nominal values.

5 Diversified dynamic weighted majority (DDWM)

This section presents the algorithm for our approach, diversified dynamic weighted majority (DDWM) followed by its detailed discussion. Our approach develops two ensembles: one with low diversity and the other ensemble having high diversity between the experts. These ensembles of weighted experts are updated based on their classification accuracy for new instances as in DWM [10, 12]. Our approach allows creation of new experts and deletion of poor performing experts based on its generalization accuracy.

We use the two ensembles that vary in their levels of diversity so that DDWM always provides good classification accuracy in handling any type of drift present in the dataset, ranging from high speed to low speed drifts; or from high severity to medium severity or low severity of drift. If the low diversity ensemble does not provide good classification accuracy in handling any type of drift, the high diversity ensemble is likely to provide good accuracy while handling that drift and vice versa.

As shown in Algorithm 1, the learning system contains two ensembles: an ensemble with low diversity (**EL**) and an ensemble with high diversity (**EH**). Different levels of diversity in each of these ensembles are introduced by replacing *Poisson* (1) with *Poisson* (λ) distribution in online bagging [21], using λ_l and λ_h to generate low and high diversity ensembles, respectively. Each of these ensembles contains weighted pool of experts (each with an initial weight of one (lines 3-8). An expert is added to the ensemble based on the final global performance (*G*) of the algorithm and the global performance of the ensemble (*GL* for low and *GH* for high) in classifying the new instances (lines 36, 37 and 46). The global performance of an ensemble (*GL* or *GH*) is the weighted majority vote of its experts' predictions (line 24–25). The final global performance of the algorithm (*G*) is the class with the maximum support considering both the ensembles (**EL or EH)** (lines 26–30). If the overall global prediction is incorrect, each of the ensembles is checked for their individual global performance. If the global prediction (*GL* or *GH*) is incorrect, a new expert with an initial weight of one is introduced in the corresponding ensemble (lines 43 and 52).

If an expert in any ensemble classifies the given instance incorrectly, then the experts' weight is reduced by a multiplicative factor (β) (lines 14–15, 20–21) and if after further training also it still performs poorly, then that expert is removed from the ensemble, provided its weight reaches a given threshold value (θ) (lines 34–35). However, a parameter p controls the weight update, creation and removal of experts. For training of the experts in an ensemble at each time step, one could use any base learner considering the various parameters of the base learner and the diversity level of the concerned ensemble (lines 58 and 61).

Algorithm 1. **Diversified Dynamic Weighted Majority** ($\{\mathbf{X}, Y\}_1^n$, β, p, θ, λ_l, λ_h)

```
{X, Y}: a training data instance, with a feature vector (X)
    and class label (Y)
β: multiplicative factor for reducing weight of experts, 0≤β
    <1
C ∈ ℕ: number of classes
{el, wl }_1^m : a low diversity ensemble EL of m experts and
    their weights
{eh, wh }_1^m : a high diversity ensemble EH of m experts
    and their weights
GL, GH, LH, LL ∈ {1,…, C}: global and local predictions
    corresponding to low and diversity ensemble
G ∈ {1,…, C}: final global prediction for any instance
    considering both the ensembles
WL, WH ∈ ℝ^c: sum of weighted predictions for each class
    corresponding to low diversity and high diversity
    ensemble, respectively
θ: threshold for removing experts
p:period between expert creation, removal or weight update
n: total number of instances
λ_l, λ_h: λ parameter for generating low and high diversity
    ensemble

1: EL ←new ensemble; // new low diversity ensemble
    using λ_l
2: EH ←new ensemble;  // new high diversity ensemble
    using λ_h
3:  for jl =1, …., m
4:    wl_jl←1;
5:  end
6:  for jh=1, …., m
7:    wh_jh←1;
8:  end
9: for  i =1,… n
10:   WL ← 0;
11:   WH ← 0;
12:   for  kl = 1,… m
13:     LL= classify (EL_kl , X_i );
14:       if (LL ≠ Y_i and i mod p = 0)
15:           wl_kl = β wl_kl ;
16:       WL_LL ← WL_LL + wl_kl;`
17:   end
18:   for kh = 1,… m
19:     LH= classify (EH_kh , X_i );
20:       if (LH ≠ Y_i and i mod p = 0)
21:           wh_kh = β wh_kh ;
22:       WH_LH ← WH_LH + wh_kh ;
23:   end
24:   GL ← argmax_kl WL_kl ;
25:   GH ← argmax_kh WH_kh ;
26:   if (WL_GL > WH_GH)
27:       G ← GL;
28:   else
29:       G ← GH;
30:   end
```

```
31:  if ($i$ mod $p$ = 0)
32:      $wl \leftarrow normalize\_weights(wl)$;
33:      $wh \leftarrow normalize\_weights(wh)$;
34:      $\{el, wl\} \leftarrow delete(\{el, wl\}, \theta)$;
35:      $\{eh, wh\} \leftarrow delete(\{eh, wh\}, \theta)$;
36:    if ($G \neq Y_i$)
37:        if ($GL \neq Y_i$)
38:            if (num_experts($EL$) = $m$)
39:              $\{el, wl\} \leftarrow rem\_weak(\{el, wl\})$
40:              $m \leftarrow m-1$;
41:            end
42:            $m \leftarrow m+1$;
43:            $el_m \leftarrow new\_expert\_creation()$;
44:            $wl_m \leftarrow 1$;
45:        end
46:        if ($GH \neq Y_i$)
47:            if (num_experts($EH$) = $m$)
48:              $\{eh, wh\} \leftarrow rem\_weak(\{eh, wh\})$;
49:              $m \leftarrow m-1$;
50:            end
51:            $m \leftarrow m+1$;
52:            $eh_m \leftarrow new\_expert\_creation()$;
53:            $wh_m \leftarrow 1$;
54:        end
55:    end
56:  end
57:   for ($kl$ =1, …… , $m$)
58:      $el_{kl} \leftarrow learn\_experts(el_{kl}, \mathbf{X_i}, Y_i)$;
59:   end
60:   for ($kh$=1, …… , $m$)
61:      $eh_{kh} \leftarrow learn\_experts(eh_{kh}, \mathbf{X_i}, Y_i)$;
62:  end
63:  output $G$;
64: end
```

The formal step by step algorithm is presented in **Algorithm 1**. The algorithm maintains two ensembles **EL** and **EH,** each containing m experts each with a weight of one, $wl_{jl,}$ and wh_{jh} for jl, jh = 1,… m (lines 1–8). These two ensembles, **EL** and **EH** maintain a set of experts having low and high diversity between the experts, respectively. Input to the system is n training instances, each consisting of a feature vector and its corresponding class label (line 9).

Every expert in an ensemble (high diversity ensemble and simultaneously low diversity ensemble) is used for classification of training instances, one instance at each time step (lines 13 and 19). This gives us their local prediction results. If the result of local prediction by an expert does not match the exact label of the training instance, the weight of the corresponding expert is reduced by a multiplicative factor β (lines 15 and 21). Regardless of the experts' prediction correctness, DDWM uses its weight to calculate the weighted sum for each class corresponding to each ensemble (lines 16 and 22), at each time step. The global prediction results for low diversity ensemble (*GL)* and high diversity ensemble (*GH)*, is the class with the maximum support (weight) as per the prediction results from each of the ensembles (lines 24 and 25). However, the final global prediction result *G*, is the class with the maximum weight considering both the ensembles (line 26–30). We consider both the ensembles because it is not necessary that the high diversity ensemble would always provide better classification accuracy than the low diversity ensemble, rather the classification accuracy of each ensemble in handling any drifting dataset depends on the speed and severity of drift present in the dataset, as explained in Sect. 3. After each weight update, the weights of all the experts are normalized so that after transformation, the maximum value of weight is one (lines 32 and 33). The normalization also prevents the newly added experts having initial weight of one to dominate the classification results.

The approach also removes the poorly performing experts from both the ensembles having weight less than a predefined threshold value θ (lines 34 and 35). However, this does not remove the last expert in each of the ensembles. The weight reduction and deletion of poorly performing experts is controlled by a parameter period p, which helps the learning system to handle large datasets (lines 14, 20 and 31).

If the final global prediction (*G)* result for any training instance is incorrect, we will check the global prediction by each of the low and high diversity ensembles, *GL* and *GH* respectively. If *GL* is incorrect, and the number of experts existing in the low diversity ensemble reaches the maximum limit of permitted ensemble size (m), the weakest expert with the minimum weight is removed from the ensemble (lines 38–40) and a new expert (with an initial weight of one) in *GL* is created maintaining low diversity among the experts (lines 42-44). Similarly, if *GH* is incorrect, a new expert in *GH* is created maintaining high diversity among the experts (lines 46–54).

The creation of new experts is also controlled by the parameter p (lines 31–56). Finally, after using the new training instance to train every single expert in each of the ensembles, at each time step (lines 57–62), DDWM outputs the final global prediction *G*, which gives us the final classification result for any instance (line 63). DDWM is an online algorithm for handling concept drift. The parameter p, defines the period over which DDWM does not update the experts' weight and will not create new experts' or delete the poorly performing experts. However, learning of the experts continues during this period p, to make the ensembles adapt better to the new target concepts.

6 Experimental evaluation and results

6.1 Experimental objectives, design and measures analyzed

The objective of the various experiments with DDWM is to analyze the behavior of our approach in different situations and to validate it, showing that it provides an answer to our first and fourth research questions presented in Sect. 2. The empirical analysis also identifies for which types of drift our approach works better and the reason for its behavior.

Empirical analysis of our approach was done using massive online analysis (MOA) [2], a tool developed for analyzing online data streams. In our experiments, the various performance measures that have been analyzed are prequential accuracy, memory usage and evaluation time. In some cases, the model-cost and kappa statistic values have also been compared. Our paper also compares the two ensembles existing in DDWM: the low diversity ensemble and the high diversity ensemble at each time step, while handling abrupt drifts in dataset. The paper presents a detailed comparative analysis of the behavior of our approach in noisy and non-noisy domains. The impact of variations in the number of experts, the period value (that controls the update, deletion and creation of new experts), variation in multiplicative factor for weight reduction and the threshold value on the performance of DDWM, has also been analyzed in detail. DDWM has been empirically compared with EDDM [1] (an approach that explicitly uses a mechanism to deal with drifts), DWM [10, 12] (an approach that does not explicitly use a mechanism to handle drifts), with Blum's implementation of Weighted Majority [3, 13], the worst-case learner i.e. standard implementation of naïve bayes (NB) with no drift handling capabilities and with Hoeffding Tree (HT) that used methods that were based on VFDT [49] enriched with change detection mechanism.

The standard implementation of naïve bayes in MOA provided the worst-case learning system as the system was not designed to handle any drifting concepts and learns from all the examples in the dataset. Hoeffding Tree is a decision tree learner that requires each example to be learned at most once and requires a small constant time to process it. The resulting tree is nearly identical with the tree built by conventional batch learner, given enough examples to train and build the Hoeffding tree. It uses Hoeffding bound that gives certain level of confidence on the best attribute to split the tree, hence we can build the model based on certain number of instances that we have seen. It does not assume feature independence for a dataset as was the requirement for the naïve bayes classifier.

The prequential accuracy is calculated based on the classification given to the current training instance before the instance is used for updating the system. The term update here refers to the learning of the experts as per the current training example; change the weights of the experts in each ensemble, deletion of poor performing experts from each ensemble and creation of new experts in each ensemble. Our approach has been compared with the other approaches in terms of the various performance metrics as discussed in Sect. 3.3, numerically as well as empirically. The numerical results provide us the average values corresponding to the various performance metrics whereas the graphs give us the actual information at each time step of learning and predictions. The illustrations provide us information about the behavior of our approach: at the time of drift, soon after the drift, fairly longer after the drift and in the presence of noise. It also helps us to identify for which types of drifts, our system is highly stable or highly sensitive. The slope of the graphs gives us information about the rate at which our concept drifting system converges to new concepts, achieving higher accuracy levels.

The method to update, create or remove the experts in DDWM is similar to the one used to maintain the experts in DWM. This is so as to provide a fair comparison between a concept drifting ensemble system with weighted experts and an ensemble system with weighted experts using the concept of diversity, thus answering the first and fourth research questions as validated by the experimental results. Comparisons were also done with EDDM, a concept drifting approach that uses a mechanism to handle concept drift. The results were completely in favor of our approach. Better performance of our approach as compared to Blum's implementation of weighted majority helps us to state that for handling drifting concepts, a pair of features is not sufficient for getting a good ensemble learning system, as was the belief of WM algorithm.

The ensembles were created using (modified) online bagging [21]. Further, diversity in the ensembles was introduced using the *Poisson* (λ) distribution in Online Bagging [21], and the parameter λ_l for getting the low diversity ensembles was set to 1, for all the experimental evaluations. The λ_h value for the high diversity ensembles were chosen after performing 10 preliminary executions using $\lambda_h = 0.0001$, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1 and 0.5, giving a total of 80 executions for each data set. The values of λ_h which gives the best average results are: 0.1 for Stagger dataset, 0.0005 for SEA dataset and 0.0005 for hyperplane. Further for real world datasets, the value of λ_h was 0.5 for electricity pricing domain and 0.05 for Breast cancer dataset. For the various experiments, the base learner mainly used by DDWM was naïve bayes [20] and to analyze the behavior of DDWM when there was feature dependence, Hoeffding tree (HT) was used in some cases. In our experiments, we repeated this procedure 50 times for each dataset and the results were the average results over these 50 runs of each dataset.

For all the experimental evaluations, the value of the multiplicative factor β for DDWM and DWM was set to 0.5. However, in case of WM the value of β (multiplicative factor) was taken to be 0.9. The value of the threshold for removing the experts in DDWM, WM and DWM, was set to 0.01 (i.e. $\theta = 0.01$ for DDWM and DWM; $\gamma = 0.01$ for WM). For WM approach each expert maintained a history of only its last prediction. The parameters warning level and drift level, for EDDM have been set to 0.95 and 0.90 respectively.

To perform empirical analysis of DDWM on Stagger concepts with 240 instances (i.e. $n = 240$), the maximum size of the ensembles was set to 10 experts (i.e. $m = 10$). We set to update the weights, create or remove experts every 10 time steps (i.e. $p = 10$). In the first context of 80 time steps, the examples are treated as positive if they have the following concept description, i.e. *size* = medium or *size* = large. In the next (80 time steps), the concept description is defined by two relevant attributes, *color* = green or *shape* = circle. In the third context, the training examples are classified as positive if the concept is *size* = small and *color* = red. To evaluate our algorithm, we randomly generate 80 training examples of the current target concept, and calculate the prequential accuracy with one example at each time step.

We perform empirical analysis on SEA concepts using 50,000 instances (i.e. $n = 50{,}000$), and setting the maximum size of the ensembles to 5. For DDWM and DWM we set to update the weights, create or remove experts every 50 time steps (i.e. $p = 50$). For the first 12, 500 time steps, the target concept is with $\theta = 7$. For the second data block, $\theta = 8$; the third data block has $\theta = 9$; and the last concept has $\theta = 9.5$. To evaluate our drift detection algorithm DDWM on SEA concepts, we randomly generate 12,500 examples of the current target concept, and compute the prequential accuracy with a new example at each time step. In one experimental evaluation, we added 10 % class noise.

To perform empirical analysis of DDWM on hyperplane concepts with 3,000 instances (i.e. $n = 3{,}000$), the size of the ensembles was set to 4 experts (i.e. $m = 4$). For DDWM and DWM, we set the value of period to be 50 time steps (i.e. $p = 50$). Random noise was introduced by switching the labels of 5 % of the training examples. We computed the prequential accuracy of the various approaches with a new example at each time step. Here we have also evaluated DDWM with another base learner, Hoeffding Tree (HT) that does not assume any feature independence as was the assumption in case of standard naïve bayes classifier.

To perform empirical analysis of DDWM on electricity pricing dataset, the maximum size of the ensembles was set to 15 experts (i.e. $m = 15$). For DDWM and DWM, we set the value of period to be 10 time steps (i.e. $p = 10$). Here we have evaluated, DDWM using two different base learners, one was naïve bayes (DDWM-NB) and another was HT (DDWM-HT). All the values for the various parameters of DWM and DDWM were the same so the comparison could be based solely on the approach used in each of the cases and not because of variation in any other feature. Since this is an online task, we first obtained predictions from the various learners and then trained each learner with one example at each time step, in the temporal order as each example appears in the dataset.

For empirical analysis of DDWM on a real time static dataset, we used the Breast cancer dataset from the UCI repository [48] containing 286 instances (i.e. $n = 286$). The maximum size of the ensembles was set to 5 and the value of period in DDWM and DWM, was set to 10 time steps. Here we have evaluated DDWM, EDDM and DWM, by using naïve bayes as the base classifier and the experimental results were the average results over 50 runs of the dataset. We processed the examples in the same order as they appear in the dataset with one example at each time step.

The empirical analysis of DDWM and the final conclusions are based on the following facts:

1. EDDM approach always uses new classifiers created from scratch which is not always the best choice to handle drifting concepts. DWM uses old experts updated as per their accuracy in classifying the new instances, create newly learnt experts and removes the experts which perform very poorly on the present concepts. Blum's implementation of WM changes the weights of the experts as per their accuracy on the training instances and trains these experts as per the new instances but does not have the provision to create new experts. Hence, EDDM, DWM and WM do not appear to be good systems in handling all types of drifts. However, our approach maintains two ensembles: a high diversity ensemble (experts classify the training instances very differently from each other) and a low diversity ensemble (experts classify the training instances almost similar to each other). In situations where low diversity ensemble does not provide good accuracy, the high diversity ensemble has a very high probability of achieving good accuracy levels and vice versa also holds.

Secondly, DDWM maintains almost double the number of experts as compared to DWM, hence giving DDWM an opportunity to maintain better learnt and better varied experts. Hence, DDWM appears to be the best system for handling any type of drifts, irrespective of the presence of noise in the dataset. The performance of our approach in terms of the various performance metrics is dependent on the selection of the base learner used in the various experimental evaluations. Our approach provided very good accuracy levels without any fear of non-accurate drift detections.

Another observation from the comparative analysis of the low diversity ensemble and the high diversity ensemble in DDWM helps us to state that for high speed drifts, the low diversity ensemble provides better prequential accuracy than the high diversity ensemble. However, with the progress in learning the experts in high diversity ensemble have an improved adaptation rate to the new concepts.

2. The memory requirements of DDWM are the highest among all the approaches. This is because DDWM maintains two sets of experts as compared to a single expert in DWM and WM, and a single classifier in EDDM and naïve bayes. Empirical analysis shows that WM requires consistent memory storage whereas DDWM storage varies with the progress in learning. This is because Blum's implementation of WM does not have the provision to delete the poor performing experts or create newly learnt experts. DWM also requires almost consistent storage as the rate of update to experts, creation or removal of experts in its single ensemble is very low as compared to DDWM which has a higher rate of update to experts, creation or removal of experts in its two ensembles in response to the drifts present in the dataset and maintains only good experts in the system. For similar reasons, the average evaluation time and the model-cost of DDWM is the maximum among all the approaches. However the exponentially low cost and higher accuracy levels of our system, makes it highly beneficial even in a resource constrained environment.

Another observation from the comparative analysis of the low diversity ensemble and the high diversity ensemble in DDWM helps us to state that the average evaluation time and the memory requirements of both the low diversity ensemble and the high diversity ensemble are almost the same and are independent of the level of diversity among the experts.

3. The kappa statistic value in DDWM varies as per the type of drift present in the dataset. DWM maintains lower value of homogeneity (lower kappa statistic value) as compared to DDWM irrespective of noise or type of drift present in the dataset. When a drift is detected, DWM maintains its ensemble the same way as DDWM but DDWM includes a low diversity ensemble which makes the system highly homogeneous (higher kappa statistic value) as compared to DWM. EDDM resets the system when the drift level is triggered hence maintaining a lower kappa statistic value (i.e. higher diversity) as compared to DDWM that only updates the system. However, in the presence of noise their kappa statistic values are almost similar because of large number of misclassifications and extensive updates to ensembles in DDWM. The kappa statistic value is dependent on the base learner used by DDWM.

In Sect. 6.2 to 6.6, using various artificial and real time datasets we have proved that the concept of diversity in DDWM has given us an ensemble system that performs with very good prequential accuracy, while handling any type of drift varying from gradual to abrupt drifts, irrespective of the presence of noise in the dataset. In Sect. 6.2, we have also empirically compared the two ensembles: the low diversity ensemble and the high diversity ensemble existing in DDWM, in terms of the various performance metrics.

The analysis has helped us to identify the performance metrics for which DDWM works better than the other approaches, on the various datasets and explained the reason for its performance. We have also analyzed the influence of the various parameters: m (size of each ensemble) (in Sect. 6.3) and p (value of period) (in Sect. 6.3 and 6.4), θ (value of threshold) (in Sect. 6.5) and β (the value of the multiplicative factor) (in Sect. 6.5) on the various performance metrics used to evaluate DDWM. The results have illustrated the impact of using naïve bayes (NB) and using Hoeffding tree (HT) as base classifiers, on the performance of DDWM while handling different types of drifts.

6.2 Experimental evaluation on stagger concepts

The results of analysis of DDWM-NB (DDWM with naïve bayes as base learner) on the Stagger concepts in terms of prequential accuracy have been illustrated as in Fig. 1a. The graphs show that DDWM performs similarly as DWM-NB (DWM with naïve bayes as base learner) and EDDM-NB (EDDM with naïve bayes as base learner) on the first target concept, and illustrates better accuracy on the second and third target concepts. However, EDDM detects false alarms between time steps 60 and 70. DDWM reached second and third target concepts earlier than EDDM with higher prequential accuracy. DDWM illustrates similar behavior as DWM on the second and third target concepts as their graphs have similar slope and asymptote and shows better accuracy than DWM. The better accuracy of DDWM is because of the diversified ensemble of experts present in DDWM and not only because of the basic ensemble of experts that also exists in DWM.

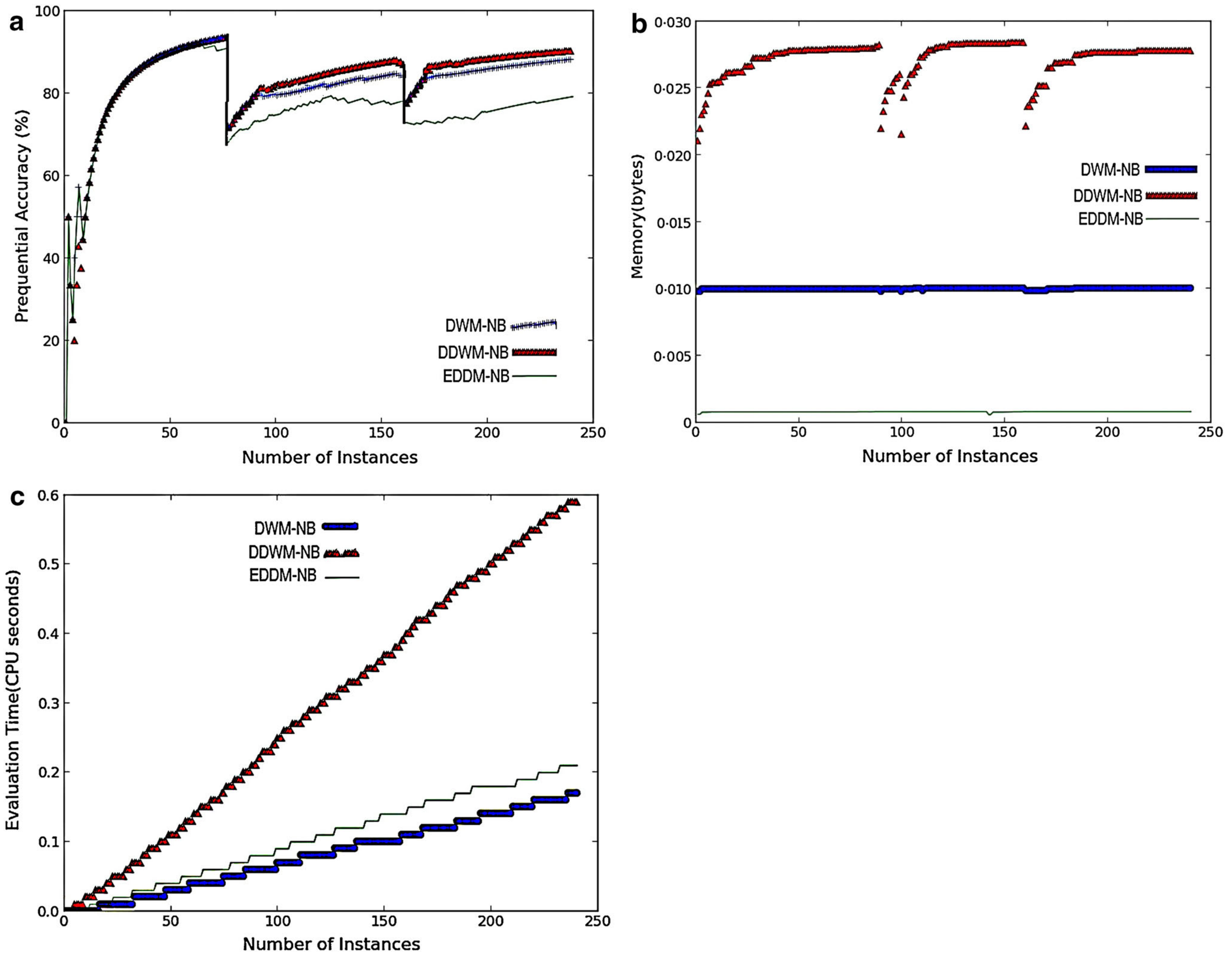

Fig. 1 Average results of evaluation of DDWM on Stagger concepts based on **a** Prequential accuracy **b** Memory **c** Evaluation Time

As expected the memory requirements of DDWM is more than double the requirements for DWM as DDWM maintains two ensembles (one low diversity and one high diversity) at any given time step in contrast to DWM which maintains only a single ensemble. The graphs for DDWM and DWM show almost consistent memory need apart from time steps, 80 and 160 where a sudden drift is observed, resulting in incorrect local predictions and large number of experts are removed from the ensembles as their weights reach the given threshold value, leading to a sudden fall in memory requirements. However, when the final prediction by DDWM was incorrect there was a sudden drop in prequential accuracy as illustrated in Fig. 1a, new experts were created in each of the ensembles resulting in gradual increase in storage requirements, and these experts after effective training result in an increase in accuracy as illustrated in Fig. 1a, b, respectively.

It can be seen from the graphs in Fig. 1b, the memory requirements dropped by almost 0.007 bytes in DDWM at times of drift as compared to a drop of only 0.001 bytes by DWM. This means that DDWM is more sensitive to errors, detecting changes and improving its performance giving higher accuracies than DWM on the new concepts. EDDM requires hardly any storage for its model as it is a single classifier approach. DDWM takes more average evaluation time than that needed by EDDM and DWM as illustrated in Fig. 1c. This is because DDWM needs to update two sets of ensembles in contrast to DWM which updates a single ensemble when there is a drift. Further, the rate of removal, creation and update to experts in DDWM is quite higher than the updates in case of DWM as seen in Fig. 1c with DDWM showing the maximum slope and DWM illustrating the minimum slope.

When DDWM is evaluated using Stagger concepts (i.e. high speed drifts), we observe that the ensemble with low diversity gives better prequential accuracy than the high diversity ensemble on all the three target concepts as illustrated in Fig. 2a. The low diversity ensemble helps DDWM achieve better accuracy than EDDM and DWM. The slope of high diversity ensemble is higher than the low

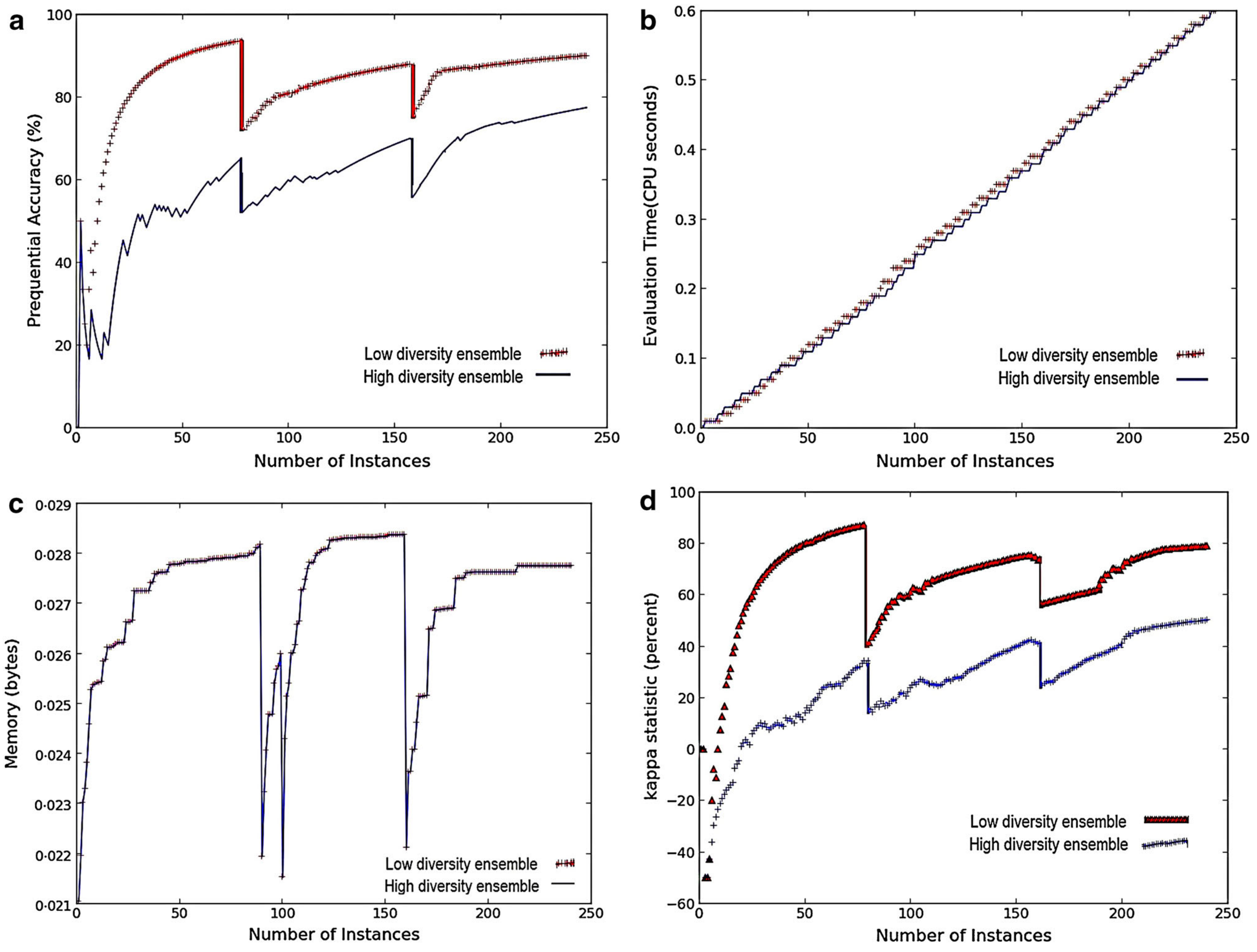

Fig. 2 Comparison of the high diversity and the low diversity ensemble in DDWM using Stagger concepts **a** Prequential Accuracy (%) **b** Evaluation Time (CPU seconds) **c** Memory (bytes) **d** kappa statistic (%)

diversity ensemble as seen in the Fig. 2a, around time steps 130 and 180, which means that with the progress in learning the experts in high diversity ensemble have a higher adaptation rate to the new concepts. This is because when a drift has high severity and high speed, it causes big changes very suddenly. The new concept has almost no similarities to the old concept. So in our approach of the two ensembles with high or low diversity levels, the ensemble learning with low diversity provides better accuracy on the new concept, after the drift.

The average evaluation time taken by both the ensembles is almost the same at each time step as seen in Fig. 2b. The low diversity and the high diversity ensemble of DDWM maintain same number of experts at each step as illustrated by overlapping of their graphs in Fig. 2c. Hence, we can state that the time and the memory requirements of any ensemble approach are independent of the level of diversity among its experts. It has been illustrated in Fig. 2d, that the low diversity ensemble in DDWM has almost 80 % homogeneity (or 20 % diversity) among its experts as its name suggests whereas the high diversity ensemble with λ_h value of 0.1 illustrates almost 30 % homogeneity (or 70 % diversity).

In the time period after 80 time steps, when the system tries to adapt to drift, the slope for the kappa statistic graph is higher for low diversity ensemble than that for high diversity ensemble. This means new experts created in low diversity ensemble increase the homogeneity level of the experts within a very short time span, maintaining the low diversity characteristic of the ensemble. Further with the progress in learning, the experts in both the ensembles regain their desired homogeneity levels for effective accuracy in classifying the new training instances. Hence, it is a good strategy to use a low diversity ensemble for handling abrupt drifts present in a dataset as it gives better accuracy than the high diversity ensemble within same time and memory requirements.

DDWM—NB performed similarly as naïve bayes and WM on the first target concept as can be seen in Fig. 3a.

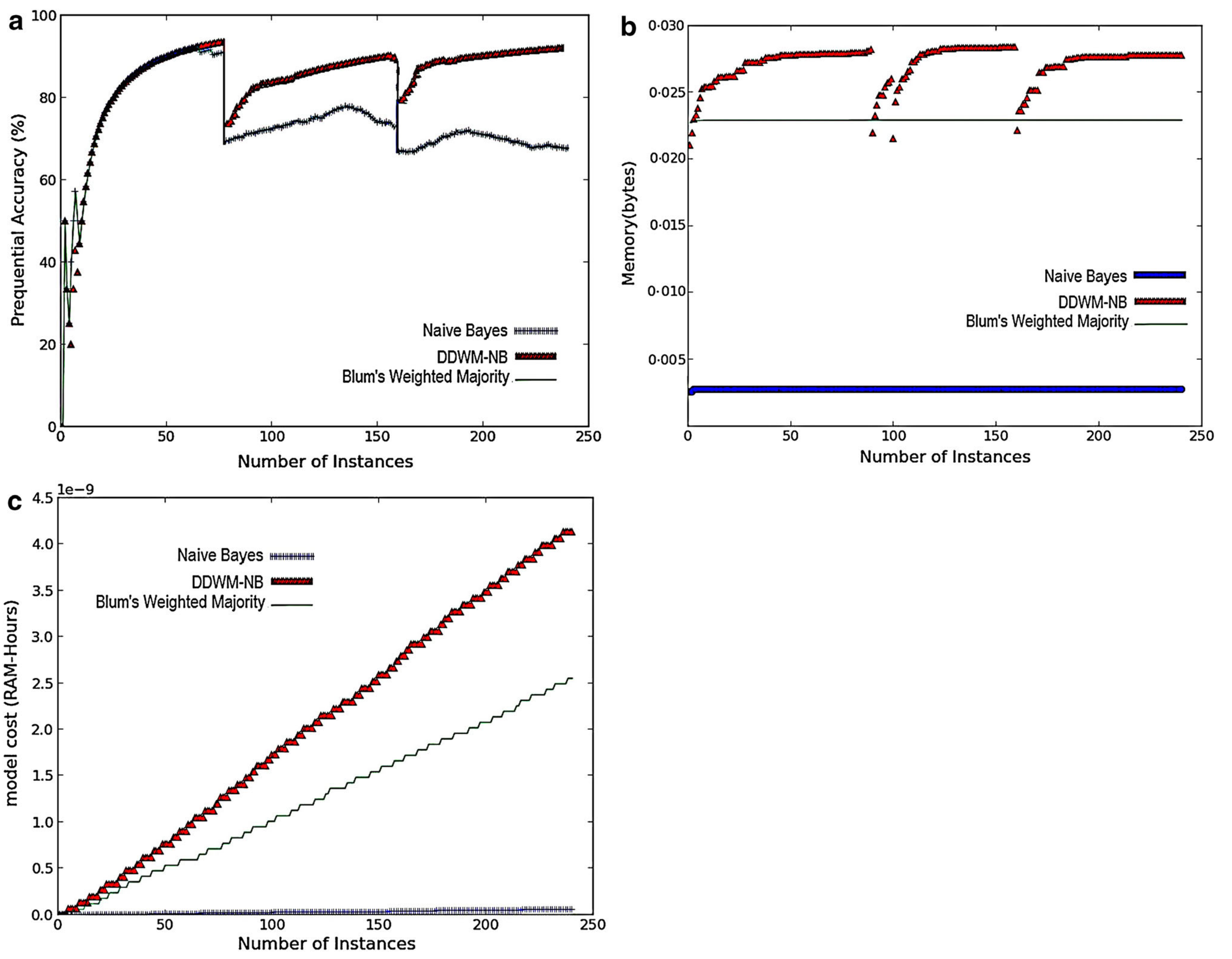

Fig. 3 Performance of DDWM -NB on the Stagger Concepts in comparison with Naïve Bayes and weighted majority. **a** Prequential accuracy (%). **b** Memory (bytes). **c** Model cost (RAM-Hours)

However, it showed better prequential accuracy than the other two learners on the second and third target concepts. Both the naïve bayes and WM gave similar accuracy on all the three target concepts, as seen by overlapping of their graphs. Our approach outperforms WM and naïve bayes in terms of slope and asymptote. WM detects false alarms drop at time steps 70 and 140, so we can say only a pair of features is not sufficient for getting a good predictor for this problem, as was believed by WM algorithm.

As expected, the memory utilization of DDWM is far more than naïve bayes and WM as illustrated in Fig. 3b, as explained earlier. However, it can be observed that WM requires consistent storage in contrast to DDWM whose storage varies as per detection of drifts in the dataset. This is because WM does not have the provision to create new experts when the predictions are incorrect and receives predictions from all the existing ($^{nu}{}_2$) experts (*nu* corresponds to number of features for any instance). On the other hand, DDWM dynamically updates, create or remove the experts in response to the drifts present in the dataset and maintains only good experts in its ensembles.

The model cost for DDWM is the maximum as it maintains its two ensembles as seen in Fig. 3c. On the other hand, WM is less costly than DDWM as it maintains only a single ensemble of experts. Naïve bayes is the least costly of all the three approaches since the model is not updated to handle drifts in concept. Further, the graph for DDWM has the highest slope indicating the highest rate of increase of model-cost as compared to that for WM and naïve bayes.

The average results of our experimental evaluation using the Stagger concepts over fifty runs of the dataset have been summarized as in Table 1. From the empirical analysis of the results, DDWM is highly sensitive to change and updates very frequently, resulting in best prequential accuracy among all the approaches. We have evaluated our

Table 1 Experimental results for DDWM-NB on the stagger concepts average over 50 runs

	EDDM	DWM	DDWM	NB	WM
Accuracy	79.16	83.31	88.43	76.00	76.00
kappa statistic	59.54	67.88	70.87	53.38	53.38
model cost (*exp. −9)	0.32	0.25	2.11	0.00	1.25
Time	0.11	0.09	0.29	0.03	0.21
Memory	0.00	0.01	0.03	0.00	0.02

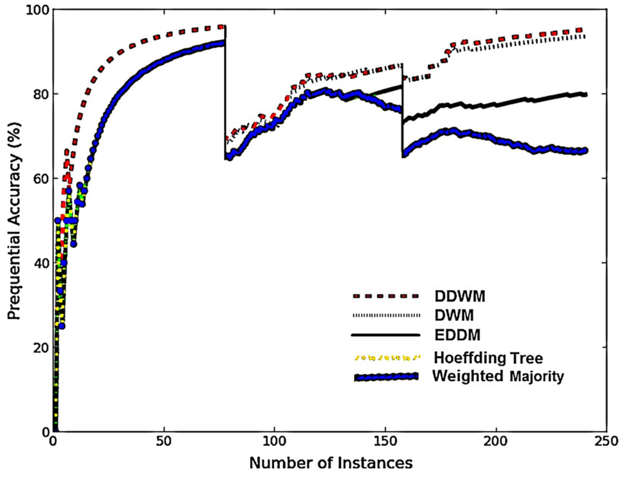

Fig. 4 Average prequential accuracy of DDWM on the stagger concepts using Hoeffding tree as base classifier

approach using naïve bayes as the base classifier that assumes independence of the features of the dataset.

However, to prove that the high performance of our approach is not dependent on any such requirement, we evaluate our approach using Hoeffding tree as base classifier that does not have any such requirement of feature independence.

The graph of Fig. 4 illustrates the results for DDWM-HT (DDWM with Hoeffding Tree as base learner) on the Stagger concepts in terms of prequential accuracy. DDWM performs with better accuracy than EDDM-HT (EDDM with Hoeffding tree as base learner) andWM-HT (WM with Hoeffding tree as base learner) on the all the three target concepts. It has also been observed that Hoeffding tree performs similarly as WM on all the target concepts as illustrated by overlapping of their graphs. DDWM-HT illustrates better performance than DWM-HT (DWM with Hoeffding tree as base classifier) on the first and third target concepts and almost similar performance on the second target concept. DDWM performs with the highest average accuracy of 85.02 %, whereas DWM-HT performs with 82.19 % accuracy, EDDM performs with 77.69 % average accuracy, Hoeffding tree and WM-HT performs with similar accuracy of 74.35 %, average over 50 runs of the dataset. Hence, from analysis of the graphs in Fig. 1a and Fig. 4, we can easily state that our approach performs the best while handling abrupt drifts in dataset, independent of any base classifier. These results prove that the high performance of DDWM is applicable for any dataset, having a variation in its feature set from being independent such as using naïve bayes classifier to dependent features such as time stamped variables.

6.3 Experimental evaluation on a very large dataset with concept drift: SEA Concepts

DDWM shows high sensitivity to noise even when the base algorithm does not support noise and improves its performance very quickly ultimately resulting in very good average prequential accuracy on the fourth target concept as illustrated in Fig. 5a. The high sensitivity to noise could be observed as we see large number of fluctuations in graph for DDWM as compared to EDDM. On the second target concept, DDWM converged more quickly to the target concept that too with higher accuracy better than EDDM and DWM approach. With the progress in learning, DDWM reached high accuracy levels almost similar to EDDM as seen in the Fig. 5(a) on the second, third and fourth target concepts.

EDDM detected false alarms as can be seen in the period surrounding 37,500 time steps by a drop in accuracy of EDDM before actual drift occurs at 37,500 time steps. On the third and fourth target concepts, DDWM and DWM learned the new concepts at almost the same time step but EDDM took more time to adjust to drifts, to achieve its best accuracy levels. On the fourth target concept after 43,000 time steps, DDWM achieved accuracy similar to EDDM approach while DWM gave the worst accuracy. Our approach was better than EDDM in terms of slope on all the four target concepts and provided very good accuracy levels without any fear of non-accurate drift detections.

Comparative analysis of DDWM-NB and DWM-NB on Stagger and SEA concepts led to the conclusions that in case of dataset with abrupt concept drift both DDWM and DWM showed similar accuracy on the first target concept as in Fig. 1a, however when noise was present our approach gave better accuracy than DWM as illustrated in Fig. 5a. This is because of extensive updates to the ensembles in DDWM as a result of noise and large number of misclassifications. Hence, DDWM handles drifts in a noisy domain better than the DWM approach and this is only because of the underlying mechanism in DDWM and not because of the base learner as both DDWM and DWM used the naïve bayes as the base learner.

As expected the evaluation time taken by DDWM is always higher than the time taken by EDDM and DWM as

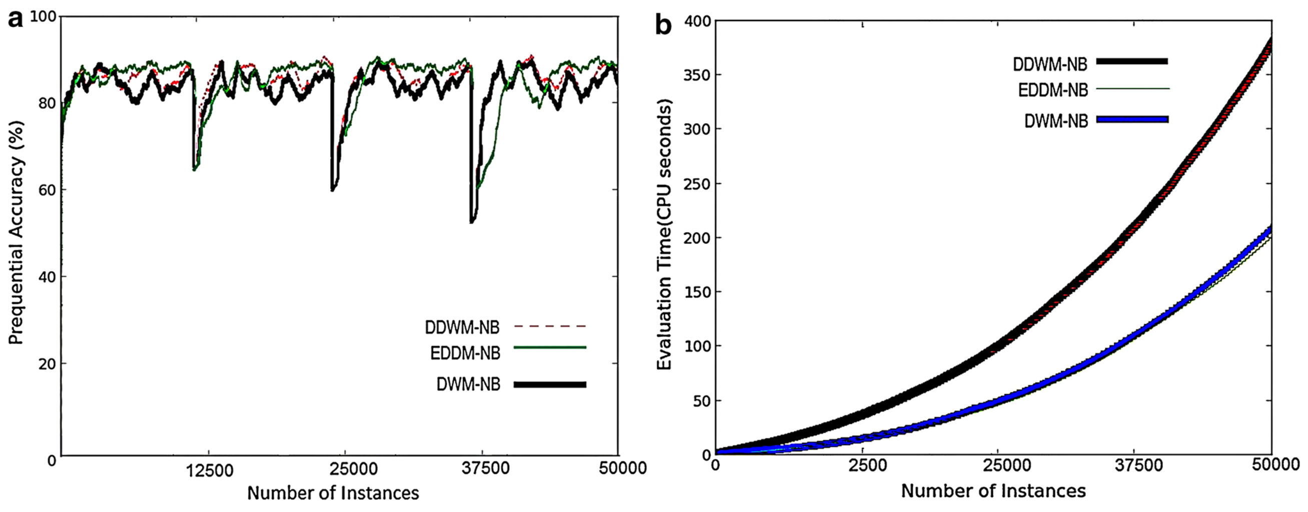

Fig. 5 Performance evaluation of DDWM-NB as compared to EDDM and DWM, using SEA concepts with 10 % class noise. **a** Prequential accuracy (%). **b** Evaluation time (CPU- seconds)

illustrated in Fig. 5b. All the learning systems have exponential time graphs in contrast to linear time graphs in-case of Stagger concepts (abrupt drift without noise). This means in noisy condition, the approaches made more misclassifications and therefore the rate of creation of experts and updates was higher, involving more CPU time than the condition without noise.

WM and naïve bayes achieve better prequential accuracy than DDWM on the first target concept as shown in Fig. 6. However, with the progress in learning on the second, third and the fourth target concept, DDWM achieved very high accuracies outperforming naïve bayes and WM approach. DDWM is highly sensitive to noise than WM and naïve bayes, detecting changes and

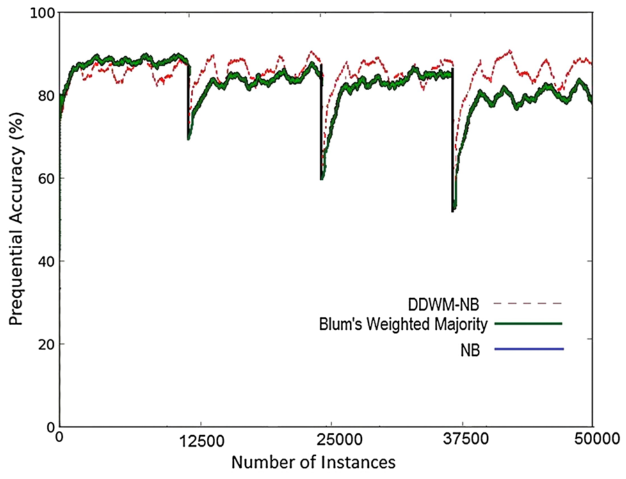

Fig. 6 Performance evaluation of DDWM-NB as compared to WM and naïve bayes on SEA concepts with 10 % class noise. The graphs for NB and WM overlap showing similar accuracy on all the four target concepts

improving its performance as seen by 5–7 % accuracy variation by DDWM as compared to only 2–3 % in case of WM and naïve bayes approach. The standard implementation of naïve bayes provided the worst case learner as it has no direct method of removing the outdated concept descriptions. When DDWM detects the presence of noise, because of incorrect local predictions the experts weights are updated and they are removed when their weights reach the defined threshold value. If during further learning, the global prediction by an ensemble is incorrect, a new better learned expert is created in the concerned ensemble. Hence for very large datasets containing noise, DDWM provides the best learnt system among all the approaches, achieving very high average accuracies.

DDWM illustrates better accuracy on SEA concepts without noise than it observed in a noisy domain as seen in Fig. 7a. In the noisy condition, since 10 % of the examples are relabeled, DDWM-NB made more mistakes and created more experts than in the condition without noise. DDWM also shows large variations in accuracy (almost 5 % to 7 %) in the noisy domain as the accuracy drops because of large number of incorrect global predictions and needed extensive training of its ensembles, whereas in the condition without noise, the graph illustrates an almost consistent and stable behavior (with maximum of 1 % variation in accuracy).

As expected, the memory graph for DDWM-NB in noisy condition shows large number of misclassifications, resulting in increased frequency of removal and creation of experts, as illustrated in Fig. 7b by large fluctuations in memory requirements in a noisy domain as compared to a condition without noise as seen in the period between time steps 16,000 and 25,000. However, after 28,000 time steps

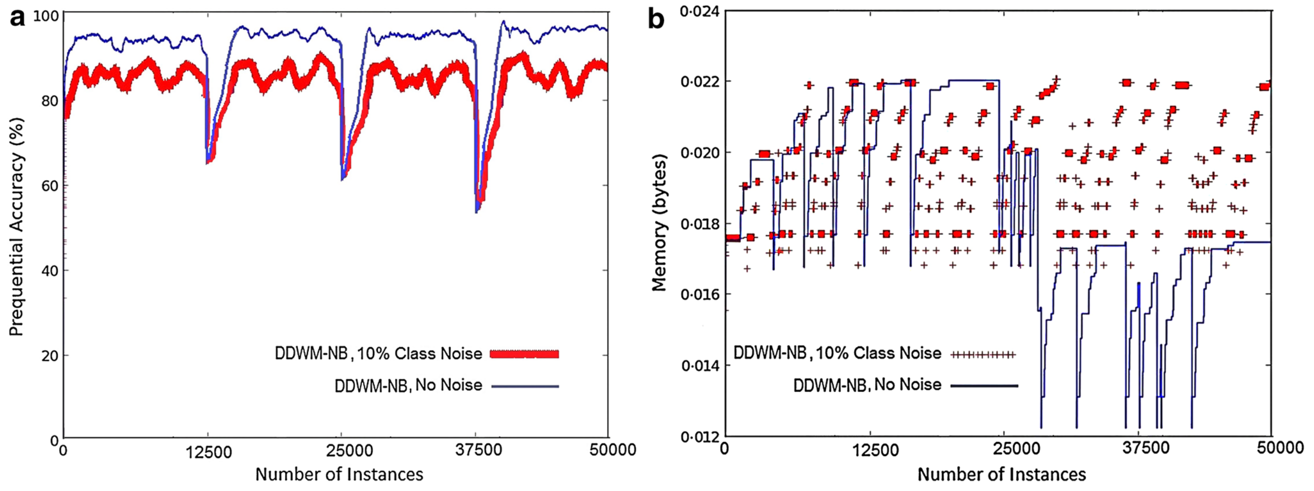

Fig. 7 Performance of DDWM-NB using the SEA concepts with 10 % class noise compared to SEA concepts with no noise. **a** Prequential accuracy (%). **b** Memory (bytes)

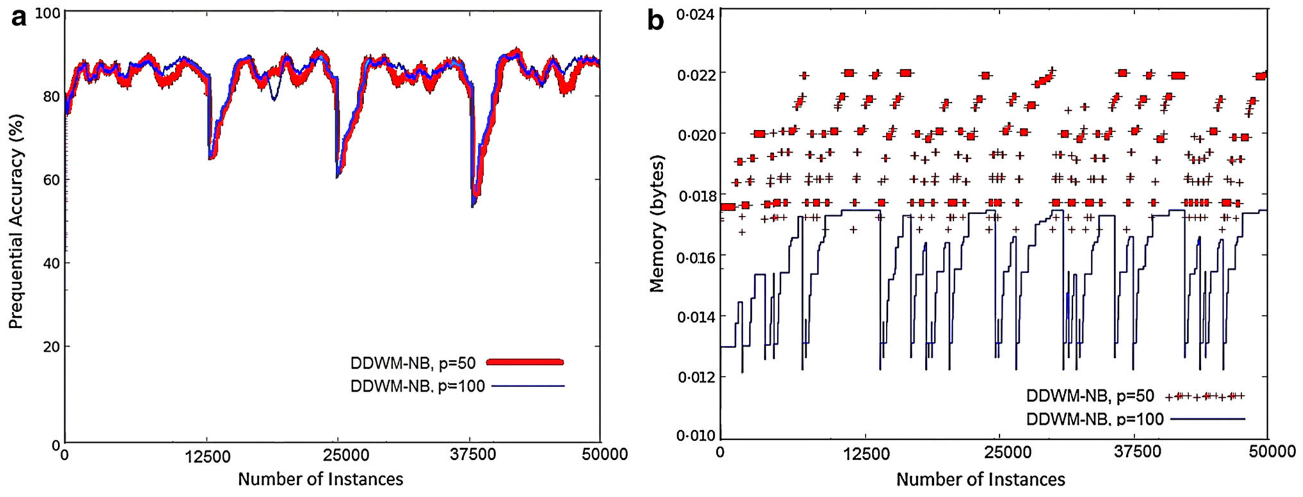

Fig. 8 Performance evaluation of DDWM-NB on SEA concepts with 10 % class noise. The value of period was increased from 50 to 100 with number of experts capped at five. **a** Prequential Accuracy (%). **b** Memory (bytes)

the two graphs show a similar trend but the memory needs of DDWM-NB in a non-noisy domain is far lower than its requirements in a noisy condition.

In order to control the large number of experts existing in a noisy domain we can use the parameter p effectively. Increasing the value of period p from 50 to 100 reduced the memory needs of DDWM-NB considerably as illustrated in Fig. 8b. This is because as we increased the value of period, the rate of updates to experts and rate of removal of experts is reduced, the system got more time to train these existing experts making them highly adaptable to the new concepts. This reduced the number of incorrect final predictions, reducing the need for creation of new experts in each of the ensembles and still achieving almost similar accuracy levels as the system with earlier period value of 50 as illustrated in Fig. 8a.

The average results of our experimental evaluations on the SEA concepts for all the concept drifting systems over the 50 runs of the dataset have been summarized as in Table 2. Analysis of the results, led to the conclusion that DDWM gave very high accuracies on dataset containing abrupt drift and noise. DDWM gave almost similar accuracy as EDDM maintaining a low diversity and a high diversity ensemble based on earlier concept descriptions and that are trained to adapt themselves on the new concept distributions. The extensive updates of experts' weights and creation of new better learnt experts, makes it highly accurate than the DWM approach, WM and the naïve bayes classifier, in terms of prequential accuracy adapting quickly to the new concepts. Our approach proves to be very resource effective as depicted by an exponentially low value of RAM-Hours.

Table 2 Average experimental results for evaluation of DDWM-NB on the SEA concepts, over 50 runs of the dataset with number of experts capped at 5 and the period value set to 50

	EDDM	DWM	DDWM (10 % class noise)	DDWM (no noise)	WM	NB
Accuracy	86.30	84.45	85.81	94.83	83.7	83.7
Kappa statistic	69.86	65.67	68.99	87.81	64.7	64.7
Model cost(*exp.−6)	0.14	0.12	0.66	0.56	0.43	0.02
Time	64.06	65.99	129.01	120.01	96.50	31.38
Memory	0.00	0.00	0.02	0.02	0.02	0.00

Table 3 Experimental results for DDWM-NB on the SEA concepts, average over 50 runs varying the parameters: number of experts *m* and period *p*

	$m = 5$, $p = 50$	$m = 5$, $p = 100$	$m = 10$, $p = 50$
Accuracy	85.81	86.36	85.54
Kappa statistic	68.99	70.29	68.38
Model cost (*exp. −6)	0.66	0.53	1.33
Time	129.01	127.62	204.43
Memory	0.02	0.02	0.03

The average results of DDWM-NB on the SEA concepts with 10 % class noise with variations in the parameters: period p and number of experts m averaged over the 50 runs have been summarized as in Table 3. The change in the value of period from 50 to 100 did not considerably affect accuracy. However, it increased the homogeneity among the experts as depicted by the value of kappa statistic. This is because if the period value is 100, it will update the experts' weights or will remove or create new experts only after 100 time steps have passed since the last update. When the period value is increased, it will take longer to update the ensembles and the old experts would be maintained for a longer time increasing the homogeneity level of the ensemble. The system achieved reduced model cost and slight reduction in the average evaluation time.

The increase in the value of parameter *m* (i.e. number of experts) from 5 to 10 experts increased the average evaluation time and the memory requirements of a system without showing any improvement in accuracy, in handling drifts. The cost of the system was almost doubled with *m* set to 10 experts. This is because our system required more time to update, store and train the 10 experts than it needed for a 5 expert system with no change in accuracy of the system. Empirical analysis of DDWM on SEA concepts with 10 % class noise led to the conclusion that the appropriate values of parameters: period p and number of experts m can make our system more real time and very effective in handling drifts in a resource constrained environment with little effect on accuracy.

6.4 Experimental evaluation on moving hyperplane dataset

From earlier research work on concept drift, EDDM has been found to be the best approach for handling gradual and very slow gradual changes. As illustrated in Fig. 9a, DDWM provides better accuracy than EDDM in the first 1,200 examples and almost similar performance as DWM. However, in the next set of 1,200 examples, EDDM achieves very high accuracy as compared to DDWM and DWM. DDWM responds quickly to changes than EDDM, updating its ensembles and improving its performance achieving very high accuracy levels same as EDDM after 2,400 time steps. On the other hand, DWM shows reduction in prequential accuracy while handling gradual drifts and noise. The better performance of DDWM as compared to DWM is because of concept of diversity introduced in its two ensemble approach and not only because of the methodology of weight updates, creation or removal of experts that is also inherent in DWM.

EDDM however detects non- accurate drifts as seen in Fig. 9(a) between time steps 300 and 400, where DDWM and DWM shows gradual increase in accuracy whereas EDDM's accuracy drops.

As illustrated in Fig. 9 (b), WM and naïve bayes perform similarly in terms of prequential accuracy as seen by overlapping of their graphs. DDWM also performed almost as well as naïve bayes and WM achieving very high accuracy on the first set of 1,200 examples. However, on the next set of examples its performance dropped considerably and the differential between the performance of DDWM and naïve bayes was very large. As seen in Fig. 9(b) around 1,200 time steps DDWM responds quickly to changes than WM approach. It updates its ensembles in response to gradual changes and noise, improving its performance thus achieving very high accuracy levels as similar as WM after 2,400 time steps. Hence our approach, DDWM provides a very good concept drifting system good enough to handle noisy gradual drifts.

In the case of gradual drifts, variation in value of period p, drastically affects the performance of DDWM as illustrated in Fig. 10a. When the period value was reduced to

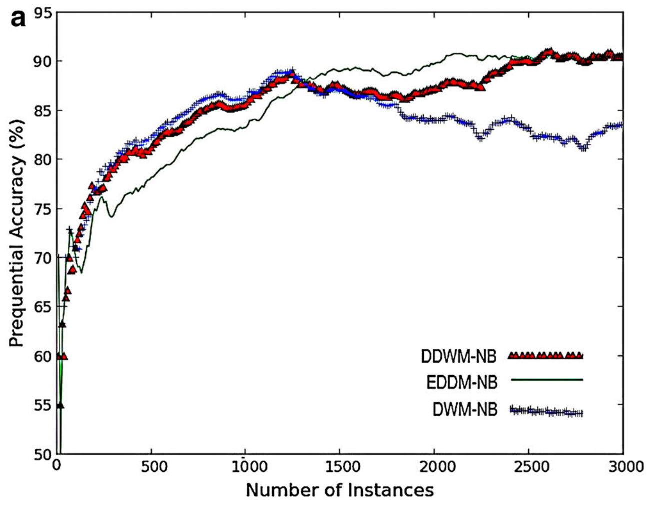

Fig. 9 Prequential accuracy (%) of DDWM-NB on moving hyperplane dataset with 5 % noise. **a** DDWM-NB as compared to EDDM and DWM. **b** DDWM-NB as compared to Naïve bayes and weighted majority

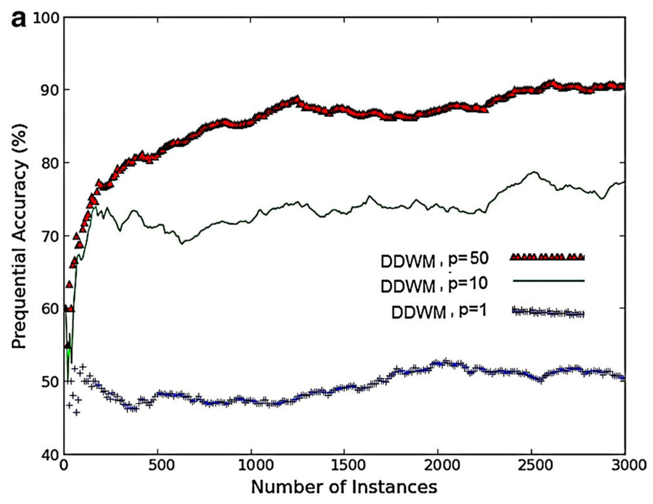

Fig. 10 Performance evaluation of DDWM on moving hyperplane dataset containing gradual drifts and noise, varying the value of period *p* based on following performance metrics. **a** Prequential accuracy (%). **b** Evaluation Time (CPU seconds)

10 time steps, i.e. updates, creation and removal of experts could happen only after 10 time steps since the last update, the prequential accuracy dropped considerably. These accuracy levels dropped further when the updates were allowed after each time step (i.e. $p = 1$). Hence, we can easily conclude that increase in the rate of updates to experts (with $p = 1$), resulted in increased rate of removal and creation of experts effecting the accuracy levels as experts did not get enough time to adapt to gradual drifts. As expected the increase in updates, creation or removal of experts increased the CPU involvement time as illustrated in Fig. 10b. All the time graphs show linear rise despite the presence of noise in dataset. So, the best value of period p for hyperplane problem was 50 time steps resulting in the best accuracy levels with the least CPU involvement.

As expected when DDWM is implemented on hyperplane dataset without noise, it achieves better accuracy as illustrated in Fig. 11a. However, at around 2,300 time steps DDWM when implemented on hyperplane dataset with noise illustrates a considerable improvement in accuracy reaching levels, higher than that achieved in the condition without noise. This means incase of datasets with gradual drifts our approach gives a more stable system and provides very good prequential accuracy irrespective of noise present in the dataset. DDWM provides a very good ensemble learning system that detects gradual changes very quickly and improves its performance achieving very high accuracies even in a noisy domain. On the contrary, in datasets with sudden drift DDWM gives better accuracy results in the condition

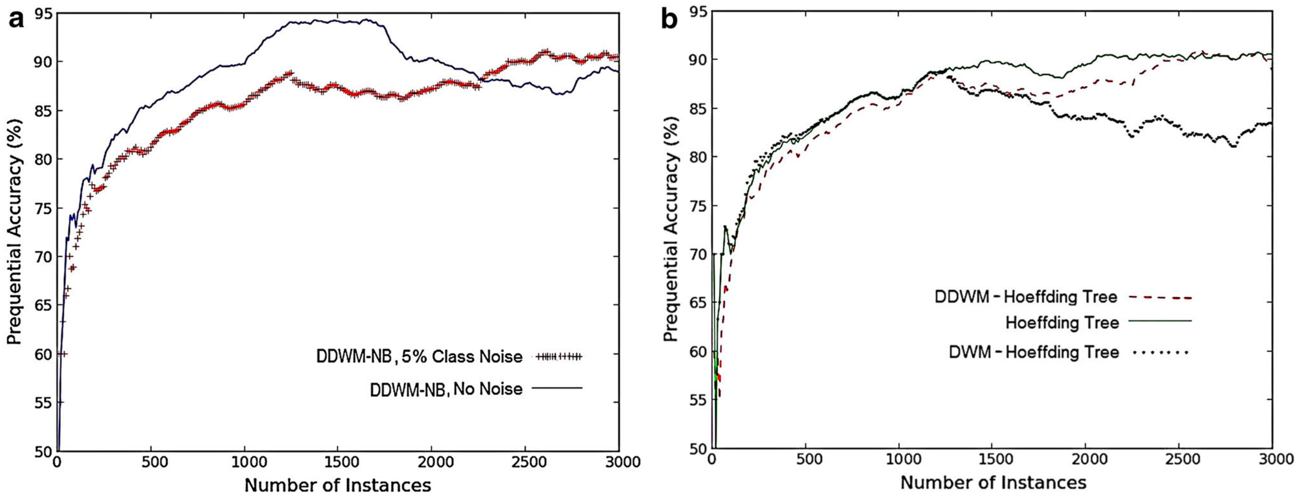

Fig. 11 Accuracy of DDWM on hyperplane dataset. **a** Dataset with noise as compared to the condition without noise. **b** DDWM with Hoeffding Tree (HT) as the base classifier as compared to DWM-HT and simply Hoeffding Tree as the concept drifting system

without noise than in the noisy domain as discussed earlier in SEA concepts.

To compare our approach using another base learning algorithm which does not have any assumption of feature independence, we have used Hoeffding Tree (HT) as the base classifier. DWM-HT and the standard implementation of HT illustrate very good prequential accuracy in the first 1,200 time steps as illustrated in Fig. 11b. DDWM-HT performed almost as well as DWM-HT and HT on the first 1,200 time steps. All the three systems have curves which are similar in terms of slope and asymptote. However, after 1,200 time steps DDWM-HT achieves higher accuracy than DWM but considerably lower accuracy than HT. Around 2,400 time steps, with the progress in learning our approach DDWM-HT illustrates very good performance as similar as HT and far better than DWM-HT which shows decrease in accuracy levels. Also, we can state that change in the base classifier does not affect the accuracy of DDWM as can be seen from graphs in Figs. 9a and 11b. This is because both naïve bayes and HT perform similarly while handling gradual drifts and noise. The average results of DDWM-NB on the hyperplane concepts averaged over 50 runs have been summarized as in Table 4. DDWM-NB provided very high accuracies while handling datasets containing gradual drifts irrespective of noise present in the dataset. It responds very quickly to change in concepts and updates its ensembles to achieve the best accuracy levels.

Table 4 Experimental results for DDWM on the hyperplane dataset with 5 % noise, average over 50 runs of the dataset using naïve bayes as the base learner

	DDWM	EDDM	DWM	WM	NB
Accuracy	85.51	85.30	83.44	86.08	86.08
kappa statistic	70.90	70.47	66.76	73.26	73.26
Model cost (*exp. −9)	2.76	0.71	0.72	1.53	0.12
Time	0.55	0.23	0.17	0.33	0.10
Memory	0.03	0.00	0.01	0.03	0.01

6.5 Experimental evaluation on electricity pricing domain

Overall, hoeffding tree averaged 79.23 % and DWM-HT averaged 87.43 % prequential accuracy whereas DDWM-HT depicted the best accuracy of 88.69 % as illustrated in Fig. 12a. DDWM converged more quickly to the target concepts than DWM and Hoeffding Tree learner as can be seen in the graphs, surrounding time step 21,000.

As seen in Fig. 12a, DDWM-HT showed the maximum accuracy whereas HT showed the minimum accuracy while handling drifts in the real-world electricity pricing dataset. The curves for DDWM and DWM were similar in terms of slope however DDWM provided a more stable system than DWM and HT. For reference, we can see the work by Harries [8] who used an online version of C4.5 [50] and builds a decision tree from examples in a sliding window reporting accuracy between 66 and 67.7 %. Hence the concept of diversity which is exclusive to our approach helps DDWM-HT achieve the best accuracy among all the approaches.

As a dataset derived from a real-world phenomenon, we cannot predict when and how many times concept drifts has occurred. Nonetheless, DDWM-HT has appeared to be more robust to change present in the samples than DWM-HT and Hoeffding tree. One example is the period around 20,000 time step, where prequential accuracy of DDWM remains nearly constant (maximum 3 % variation) whereas Hoeffding tree shows a large drop in their accuracies

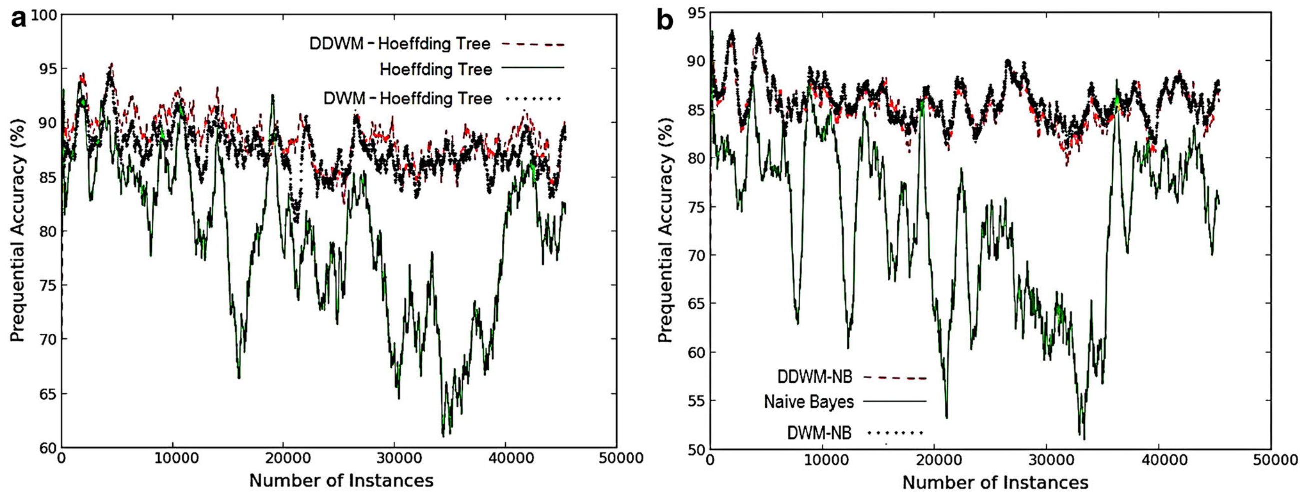

Fig. 12 Accuracy of DDWM on electricity pricing domain. **a** Using Hoeffding Tree as the base classifier. **b** Using naïve bayes as the base classifier

around 15 % drop. In case of DDWM if one ensemble performs poorly another ensemble provides good classification results so that the system provides good overall accuracy. This is clearly illustrative in the graphs around time step 20,000, as the accuracies of DWM and Hoeffding tree drop, whereas DDWM shows an improvement in its accuracy levels. DDWM-NB achieves accuracies higher than naïve bayes and almost similar accuracy as DWM-NB as seen in Fig. 12b.

DDWM-HT achieved higher accuracies than DDWM-NB as can be seen from the graphs of Fig. 12. If we compare the plots of Hoeffding tree and naïve bayes, we see that hoeffding tree achieved higher accuracies as compared to naïve bayes. Thus, the higher accuracy of DDWM-HT is due to differences in the base learners rather than to something inherent in DDWM.

As illustrated in Fig. 13a, variation in the value of β in DDWM does not affect accuracy of our approach. However, it does impact the memory requirements and the cost of the system. DDWM with β value of 0.7 requires the maximum average memory to store its experts whereas the systems with β value of 0.3 and 0.5 require lower and the same amount of memory for its effective evaluation as seen in Fig. 13b. This is because higher the value of β would reduce the experts' weight earlier to the threshold value and result in increased frequency of removal and creation of new experts in the ensembles, resulting in increased model cost of the system as illustrated in Fig. 13c.

The model cost of DDWM was maximum when the β value was 0.7 whereas DDWM with β value of 0.5 and 0.3, involved similar cost as illustrated in Fig. 13c. This means the best value for β was 0.5 to effectively update weight of poor performing experts, without any increase in memory requirements and cost (as further lower values of β did not reduce the memory requirements or cost) of the system. The variation in the value of threshold θ did not influence the performance of DDWM in terms of any of the metrics, apart from evaluation time which varies very little as observed from Table 5.

The average results of DDWM on electricity pricing dataset have been summarized as in Table 6. So we can easily conclude from the results, that DDWM gives the best prequential accuracy among various approaches on a real time electricity pricing domain when HT was used as the base classifier. When DDWM and DWM used HT as base classifier, they required almost double the CPU evaluation time than when naïve bayes was used as their base classifier. So the increase in the time is because of change in base classifier and not because of something inherent in DDWM. The use of HT as base classifier increased the homogeneity among the experts of DDWM as compared to that using NB as base learner, as validated by the value of kappa statistic in Table 6. All the systems proved to be very resource effective as validated by almost exponentially low value of RAM-Hours.

6.6 Experimental evaluation on breast cancer dataset

DDWM-NB achieved better prequential accuracy than DWM at every single time step as illustrated in Fig. 14a. The graph for DDWM is better than DWM in terms of slope as seen around 40 time steps. DDWM achieves higher accuracies as compared to DWM that too at a very fast rate as seen in the period between time steps 150 and 160. DDWM-NB provides a more stable system than DWM-NB. This can be clearly seen from the examples in the period between time steps 175 and 200, where the prequential accuracy of DDWM remains nearly constant while DWM illustrates a drop during this period. Overall,

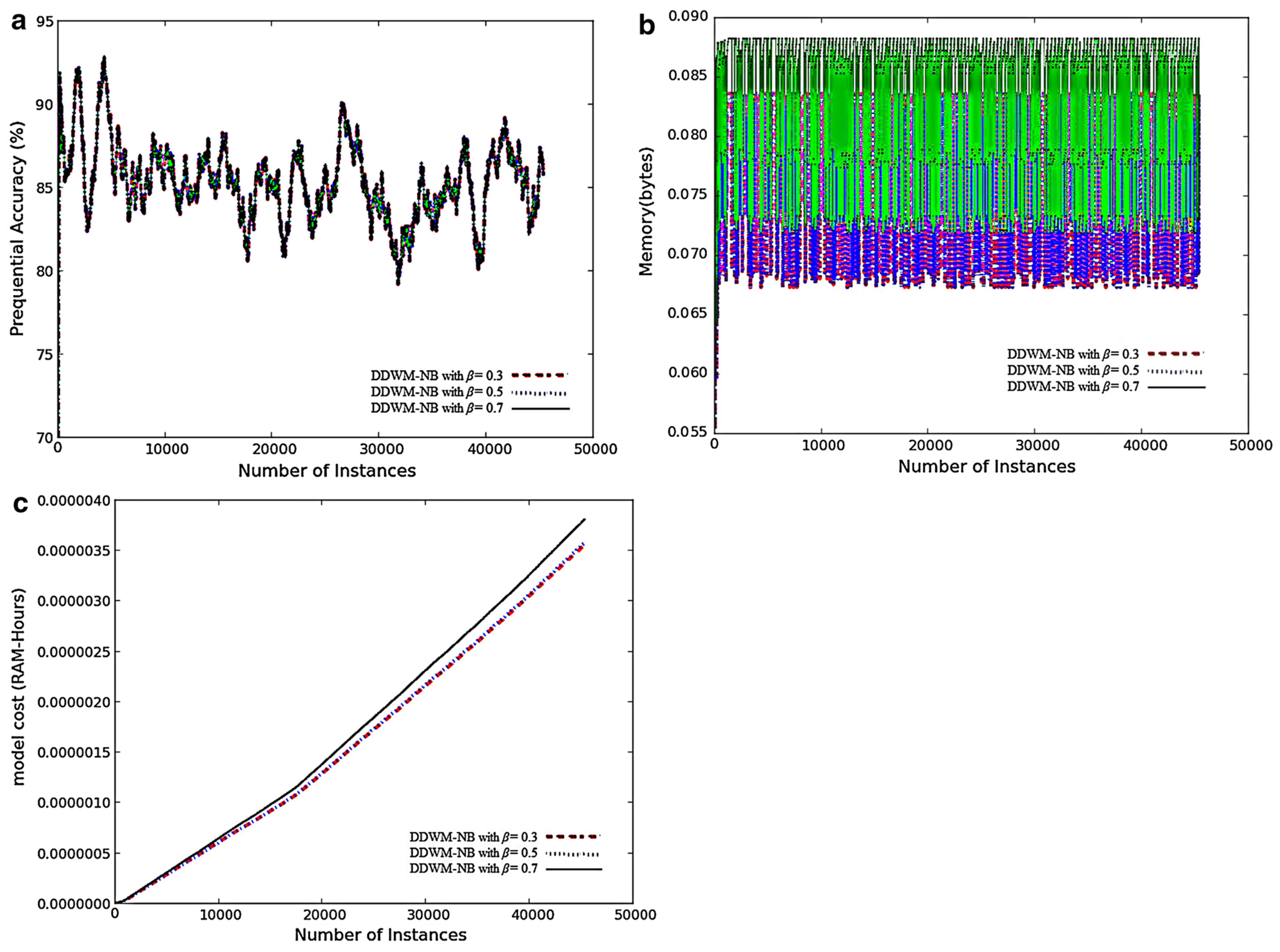

Fig. 13 Performance evaluation of DDWM on electricity pricing domain, varying the value of multiplicative factor β keeping threshold value at 0.01. **a** Prequential accuracy (%). **b** Memory (bytes). **c** Model cost (RAM-Hours)

Table 5 Experimental results for DDWM –NB on electricity pricing domain, varying the value of threshold θ, all the other parameters are same (i.e. $\beta = 0.5$, $m = 15$, $p = 10$)

	$\Theta = 0.005$	$\Theta = 0.01$	$\Theta = 0.03$
Accuracy	85.37	85.37	85.37
kappa statistic	69.72	69.72	69.72
Model cost (*exp. −6)	1.23	1.22	1.22
Time	77.77	77.67	77.64
Memory	0.08	0.08	0.08

DDWM averaged 68.66 % accuracy and DWM-NB averaged 65.06 % accuracy over 50 runs of the dataset.

The memory needs for DDWM is almost double the needs for DWM as it maintains two ensembles rather than one in-case of DWM as illustrated in Fig. 14b. However, DDWM provides better learnt ensembles as it maintains only the minimum required number of experts and removes the poor performing experts. This was shown in the graph by gradual drop of 0.01 bytes in memory needs of the two ensembles of DDWM whereas DWM shows a drop of almost 0.001 bytes for a single ensemble as seen in Fig. 14b around 70 time steps. This means there is an average drop of 0.005 bytes in memory requirements per ensemble in DDWM which is huge as compared to DWM. Reduction in memory needs means, large number of poor performing experts were removed in DDWM as compared to DWM. The average results of DDWM on breast cancer dataset have been summarized as in Table 7.

After analysis of the results as summarized in Table 7, we can conclude that DDWM performs better than DWM and EDDM in terms of prequential accuracy. However, DDWM's performance is almost similar as the performance of standard implementation of naïve bayes classifier that has not been designed to handle any drifts and learns from all the examples in the stream. The approach provides us highly homogeneous experts as observed from its kappa statistic value, which is highly beneficial for classification in static datasets. DDWM proves to be highly resource efficient achieving high accuracy even in static concepts.

Table 6 Average experimental results for DDWM on electricity pricing domain, with naïve bayes and hoeffding tree as the base learners

	DDWM-HT	DWM-HT	HT	NB	DDWM-NB	DWM-NB
Accuracy (%)	88.69	87.43	79.23	73.40	85.87	85.86
Kappa statistic(%)	76.52	73.98	56.27	39.95	69.72	70.69
Model cost (*exp.−6)	12.8	0.03	0.03	0.002	1.22	0.002
Time (CPU-seconds)	160.66	7.96	12.04	1.66	77.67	4.03
Memory (bytes)	0.39	0.02	0.07	0.01	0.08	0.01

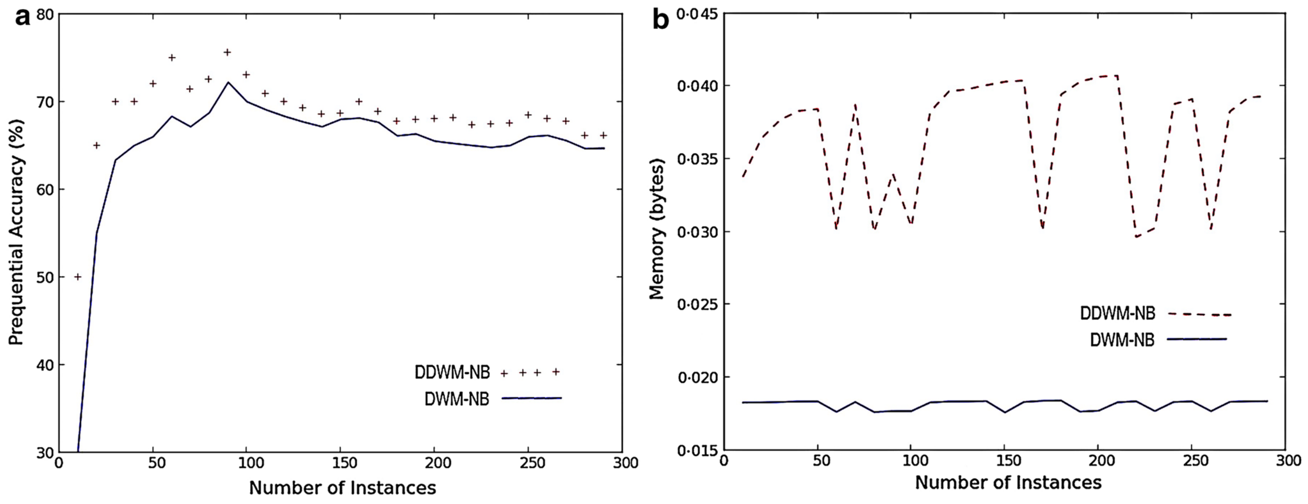

Fig. 14 Performance of DDWM on Breast Cancer dataset containing 286 instances, averaged over 50 runs of the dataset based on following metrics. **a** Prequential accuracy (%). **b** Memory (bytes)

Table 7 Average experimental results for DDWM using breast cancer dataset, with naïve bayes as base classifier

	DDWM	DWM	NB	EDDM
Accuracy (%)	68.61	65.06	68.62	66.18
Kappa statistic (%)	24.97	20.75	26.11	20.67
Time(CPU-seconds)	0.07	0.02	0.01	0.02
Memory(bytes)	0.04	0.02	0.01	0.00

Hence DDWM could be used for classification of any dataset varying from static to highly dynamic datasets.

7 Conclusions

In this paper, we presented an online ensemble method, Diversified dynamic weighted majority which maintains two sets of weighted ensembles: one with high diversity and the other one with low diversity. We described two implementations of DDWM, one with naïve bayes as the base learner and the other using hoeffding tree as the base classifier. On the problem domains we considered, the diversity concept in the two sets of weighted ensembles helped DDWM provide a better response to concept drift than the other learners such as those that considered only single ensemble of weighted experts (i.e. DWM and WM), and approaches based on the distance between classification errors (i.e. EDDM).

Analysis of the experimental results of DDWM using Stagger concepts state that DDWM shows the best average prequential accuracy among DWM and EDDM. Our approach is highly sensitive to errors, detects changes and improves its performance delivering the best accuracy in handling sudden drifts in dataset. However the average evaluation time was the maximum for DDWM as it maintains two sets of experts and involves highest rate of updates, creation and deletion of experts. Comparison between the high diversity and low diversity ensemble of DDWM clearly proves that for sudden drifting datasets, the low diversity ensemble provides us better accuracy than the high diversity ensemble. Secondly, the evaluation time taken by any ensemble is independent of the level of diversity among its experts. Empirical evaluation of our approach using Stagger concepts states that DDWM also provides better prequential accuracy than the Blum's implementation of weighted majority and the standard implementation of naïve bayes. We also evaluated our approach using Hoeffding tree as base classifier, achieving very high average prequential accuracy. The results led us to state that our approach performs with high accuracy for any dataset, varying from dependent feature set to an independent feature set as in naïve bayes classifier.

Our approach proves itself to be the best approach in handling abrupt drifts in very large datasets as seen by analysis of evaluation results using SEA concepts. It provided very high accuracies and converged very quickly to the target concepts as compared to the other approaches. DDWM shows very high sensitivity to noise but detects changes and improves its performance achieving the best accuracy levels as similar as EDDM. On the other hand, EDDM suffered from false alarms and large number of non-accurate drift detections. Results on SEA concepts without noise prove that DDWM provided very high accuracies on very large dataset without noise, with highly reduced memory requirements and lower evaluation time than in a noisy domain.

The change in the value of period did not significantly affect accuracy but considerably reduced the memory requirements of DDWM as seen in the experimental results using SEA concepts. The increase in the period value led to reduction in model cost and decreased the overall CPU involvement. Empirical analysis of DDWM using SEA concepts clearly led to the conclusion that the increase in the value of parameter *m* (i.e. number of experts*)*, increased the average evaluation time, model cost and the memory requirements of a system without showing any improvement in accuracy, in handling sudden noisy drifts. Hence we can easily state that the value of period and the maximum number of experts existing in any given system for handling abrupt drifts should be chosen carefully so as to get the best results within least possible memory and time requirements.

Our approach responds quickly to gradual changes, updating its ensembles and improves its performance achieving very high accuracy levels as seen in experiments using moving hyperplane problem. It was observed that the change in the value of period greatly affects the performance of our system while handling gradual drifts in dataset. The accuracy of the system was directly proportional to the value of period and the evaluation time was inversely proportional to the period value while handling gradual drifts containing noise.

To evaluate our approach on a real-world problem we used Electricity pricing domain. The results show that the use of a better base learning algorithm such as HT improved the accuracy of DDWM as compared to its accuracy when NB was used as the base learning algorithm. Secondly, our approach provided a very stable system with almost consistent accuracy levels. It was empirically proved that the variation in the multiplicative factor in DDWM does not affect accuracy of DDWM but it does impact the memory requirements and the cost of the system. However, the variation in the threshold value does not affect the performance of DDWM in terms of any of the performance metrics. We also evaluated DDWM on another real-world static dataset, breast cancer dataset. Our approach converged quickly to target concepts with higher accuracy levels, giving a better learnt system than DWM.

For future work, we plan to improvise our approach to handle datasets with weighted instances, in which based on the weights some training instances will be explicitly discarded from the training dataset. We would also like to investigate mechanisms for handling novel class detection such as those present in an adaptive ensemble classifier approach [51]. We would also try to enhance our approach to handle recurrent drifts, investigating mechanisms as those present in ACE [18, 20]. We also aim to handle concept drifting under dynamic feature sets where the distinctive features used for describing the instances, change over time. New better mechanisms can be introduced to handle weights of experts, to make our approach more resource effective by reducing its overall time and memory usage and maintaining only the best experts in each of the ensembles that are sufficient enough to provide good accuracy in a resource constrained environment.

References

1. Baena-Garcıa M, Del Campo-Avila J, Fidalgo R, Bifet A (2006) Early drift detection method. In: Proceedings Fourth ECML PKDD Int'l Workshop Knowledge Discovery from Data Streams (IWKDDS'06), pp 77–86
2. Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) MOA: massive online analysis, a framework for stream classification and clustering. In: workshop on applications of pattern analysis, JMLR: Workshop and Conference Proceedings, vol 11. p 44
3. Blum A (1997) Empirical support for winnow and weighted majority algorithms: results on a calendar scheduling domain, machine learning. Kluwer Academic Publisher, Boston
4. Dawid A, Vovk V (1999) Prequential probability : principles and proper ties. Bernoulli 5(1):125–162
5. Dietterich TG (1997) Machine learning research: four current directions. Artif Intell 18(4):97–136
6. Gama J, Medas P, Castillo G, Rodrigues P (2004) Learning with drift detection, In: Proceeding Seventh Brazilian Symp. Artificial Intelligence (SBIA'04), pp. 286–295
7. Gao J, Fan W, Han J (2007) On appropriate assumptions to mine data streams: analysis and practice. In: Proceedings IEEE Int'l Conf. Data Mining (ICDM,'07), pp 143–152
8. Harries M (1999) Splice-2 comparative evaluation: electricity pricing, Technical report. University of New South Wales, Australia
9. Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams, In: Proceedings KDD'01, ACM Press. San Francisco, 2001, pp 97–106
10. Kolter JZ, Maloof MA (2003) Dynamic weighted majority: a new ensemble method for tracking concept drift. In: Proceedings of the 3rd ICDM, USA, pp 123–130
11. Kolter JZ, Maloof MA (2005) Using additive expert ensembles to cope with concept drift. In: Proceedings Int'l Conf. Machine Learning (ICML'05), pp 449–456

12. Kolter JZ, Maloof MA (2007) Dynamic weighted majority: an ensemble method for drifting concepts. J Mach Learn Res 8:2755–2790
13. Littlestone N, Warmuth M (1994) The weighted majority algorithm. Inf Comput 108:212–261
14. Mansoori M, Zakaria O, Gani A (2012) Improving exposure of intrusion deception system through implementation of hybrid honeypot. IAJIT 9 (5): 436–444
15. Minku FL, White A, Yao X (2010) The Impact of Diversity on On-Line Ensemble Learning in the Presence of Concept Drift. IEEE Trans Knowl Data Eng 22(5):730–742
16. Minku LL, Yao X (2012) DDD: a new ensemble approach for dealing with concept drift. IEEE Trans Knowl Data Eng 24(4):619
17. Nishida K (2008) Learning and Detecting Concept Drift, PhD dissertation, Hokkaido Univ. [Online]. http://lis2.huie.hokudai.ac.jp/%20%20knishida/paper/nishida2008-dissertation%20.pdf
18. Nishida K, Yamauchi K (2007) Adaptive classifiers-ensemble system for tracking concept drift. In: Proceedings Sixth Int'l Conf. Machine Learning and Cybernetics (ICMLC'07), pp 3607–3612
19. Nishida K, Yamauchi K (2007) Detecting concept drift using statistical testing. In: Proceedings 10th Int'l Conf. Discovery Science (DS'07), pp 264–269
20. Nishida K, Yamauchi K, Omori T (2005) ACE: adaptive classifiers-ensemble system for concept-drifting environments. In: Proceedings of the 6th International Workshop on Multiple Classifier Systems, ser. Lect Notes Comput Sci 3541:176–185
21. Oza NC, Russell S (2001) Experimental comparisons of online and batch versions of bagging and boosting. In: Proceedings of the Seventh ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD'01), ACM Press, New York, pp 359–364
22. Scholz M, Klinkenberg R (2005) An ensemble classifier for drifting concepts. In: Proceedings of the Second International Workshop on Knowledge Discovery from Data Streams (IWKDDS'05), Porto, pp 53–64
23. Sidhu P, Bhatia MPS (2014) Extended dynamic weighted majority using diversity to handle drifts. New Trends Databases Inf Sys Adv Intell Sys Comput 241:389–395
24. Stanley KO (2003) Learning concept drift with a Commitee of decision trees, Technical Report AI-TR-03-302, Dept. of Computer Sciences, Univ. of Texas, Austin
25. Street W, Kim Y (2001) A streaming ensemble algorithm (SEA) for large-scale classification, In: Proceedings of the 7th ACM International Conference on Knowledge Discovery and Data Mining, ACM Press, New York, pp 377–382
26. Schlimmer JC, Granger RH (1986) Incremental learning from noisy data. Mach Learn 1(3):317–354
27. Tsymbal A (2004) The problem of concept drift: definitions and related work, Technical Report TCD-CS-2004-15. Department of Computer Science, Trinity College Dublin, Ireland
28. Kubat M, Widmer G (1996) Learning in the presence of concept drift and hidden contexts, Machine Learning, 23 (1): 69–101.16.Klinkenberg R., Learning drifting
29. Kuncheva LI, Whitaker CJ (2003) Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach Learn 51:181–207
30. Tang EK, Sunganthan PN, Yao X (2006) An analysis of diversity measures. Mach Learn 65:247–271
31. Yule G (1900) On the association of attributes in statistics, Philosophical Trans. Royal Soc. of London, Series A, vol 194, pp 257–319
32. Gama J, Sebastião R, Rodrigues PP (2009) Issues in evaluation of stream learning algorithms, In KDD'09, pp 329–338
33. Minku FL, Yao X (2009) Using diversity to handle concept drift in on-line learning, In: Proceedings Int'l Joint Conf. Neural Networks (IJCNN, 2009b), pp 2125–2132
34. Su L, Liu HY, Song ZH (2011) A new classification algorithm for data stream. International Journal of Modern Education and Computer Science 4:32–39
35. Murphy PM (1998) UCI Repository of machine learning databases. Department of Information and Computer Sciences, University of California, Irvine, available at http://www.ics.uci.edu/~mlearn/
36. Blake C, Merz C (1998) UCI repository of machine learning databases. Department of Information and Computer Sciences, University of California, Irvine, Web site (Online). http://www.ics.uci.edu/~mlearn/MLRepository.html.
37. Tsai CJ, Lee CI, Yang WP (2009) Mining decision rules on data streams in the presence of concept drifts. Expert Syst Appl 36:1164–1178
38. Gaber MM, Yu PS (2006) Detection and classification of changes in evolving data streams. Int J Inf Technol Decis Mak 5:659–670
39. Yang Y, Wu X, Zhu X (2005) Combining proactive and reactive predictions for data streams, In Proceedings of ACM SIGKDD, pp 710–715
40. Wang H, Fan W, Yu PS, Han J (2001) Mining concept-drifting data streams using ensemble classifiers. In: Proceedings ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp 226–235
41. Chu F, Zaniolo C (2004) Fast and light boosting for adaptive mining of data streams. In: Proceedings Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD'04), pp 282–292
42. Scholz M, Klinkenberg R (2007) Boosting classifiers for drifting concepts. Intell Data Anal Spec Issue Knowl Discov Data Streams 11(1):3–28
43. S. Ramamurthy, R. Bhatnagar, Tracking Recurrent Concept Drift in Streaming Data Using Ensemble Classifiers, In Proc. Int'l Conf. Machine Learning and Applications (ICMLA'07), pp. 404-409, 2007
44. Gao J, Fan W, Han J, Yu P (2007) A general framework for mining concept-drifting data streams with skewed distributions. In: Proceedings SIAM Int'l Conf. Data Mining (ICDM)
45. He H, Chen S (2008) IMORL: incremental Multiple-Object Recognition and Localization. IEEE Trans Neural Networks 19(10):1727–1738
46. Polikar R, Udpa L, Udpa SS, Honavar V (2001) Learn ++: an incremental learning algorithm for supervised neural networks. IEEE Trans Sys Man Cybernet Part C 31(4):497–508
47. Kasabov N (2003) Evolving connectionist systems. Springer, London
48. Asuncion A, Newman DJ (2007) UCI machine learning repository. Web site, Department of Information and Computer Sciences, University of California, Irvine, http://www.ics.uci.edu/~mlearn/MLRepository.html
49. Domingos P, Hulten G (2000) Mining high-speed data streams. In: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press, New York pp 71–80
50. Quinlan JR (1993) C4.5: Programs for machine learning. Morgan Kaufmann, San Francisco
51. Dewan MF, Zhang L, Hossain A, Chowdhury MR, Rebecca S, Graham S, Keshav D (2013) An adaptive ensemble classifier for mining concept drifting data streams. Expert Sys Appl 40(15):5895–5906. doi:10.1016/j.eswa.05.001
52. Zliobaite I (2009) Learning under concept drift: an overview, Technical report faculty of mathematics and informatics. Vilnius UniversityLithuania, Vilnius

53. Tumer K, Ghosh J (1996) Error correlation and error reduction in ensemble classifiers. Connect Sci 8(3):385–404
54. Schlimmer J, Granger R (1986) Beyond incremental processing: tracking concept drift. In: Proceedings of the 5th National Conference on Artificial Intelligence, AAAI Press, Menlo Park, CA, pp 502–507
55. Elwell R, Polikar R (2011) Incremental learning of concept drift in nonstationary environments. IEEE Trans Neural Netw 22:1517–1531
56. Bhardwaj M, Bhatnagar V (2014) Towards an optimally pruned classifier ensemble. Int J Mach Learn Cybernet. doi:10.1007/s13042-014-0303-8
57. Baumgartner D, Serpen G (2013) Performance of global–local hybrid ensemble versus boosting and bagging ensembles. Int J Mach Learn Cybernet 4(4):301–317
58. Christou IT, Gekas G, Kyrikou A (2012) A classifier ensemble approach to the TV-viewer profile adaptation problem. Int J Mach Learn Cybernet 3(4):313–326
59. Wang XZ, Wang R, Feng HM, Wang H (2014) A new approach to classifier fusion based on upper integral. IEEE Transactions on Cybernetics 44(5):620–635