Heterogeneous Ensemble for Feature Drifts in Data Streams

Hai-Long Nguyen[1], Yew-Kwong Woon[2], Wee-Keong Ng[1], and Li Wan[3]

[1] Nanyang Technological University, Singapore
nguy0105@ntu.edu.sg, wkn@acm.org
[2] EADS Innovation Works Singapore
david.woon@eads.net
[3] New York University
wanli@cs.nyu.edu

Abstract. The nature of data streams requires classification algorithms to be real-time, efficient, and able to cope with high-dimensional data that are continuously arriving. It is a known fact that in high-dimensional datasets, not all features are critical for training a classifier. To improve the performance of data stream classification, we propose an algorithm called HEFT-Stream (Heterogeneous Ensemble with Feature drifT for Data Streams) that incorporates feature selection into a heterogeneous ensemble to adapt to different types of concept drifts. As an example of the proposed framework, we first modify the *FCBF* [13] algorithm so that it dynamically update the relevant feature subsets for data streams. Next, a heterogeneous ensemble is constructed based on different online classifiers, including *Online Naive Bayes* and *CVFDT* [5]. Empirical results show that our ensemble classifier outperforms state-of-the-art ensemble classifiers (AWE [21] and OnlineBagging [15]) in terms of accuracy, speed, and scalability. The success of HEFT-Stream opens new research directions in understanding the relationship between feature selection techniques and ensemble learning to achieve better classification performance.

1 Introduction

With rapid technological advancement, many real-life applications, such as stock markets, online stores and sensor networks can produce massive datasets, or *data streams*. To discover knowledge from data streams, scientists have to confront the following challenges: (1) tremendous volumes of data; (2) dynamic changes of the discovered patterns, which is commonly referred to as *concept drifts*; and (3) real-time response. Concept drifts are categorized into two types: gradual drifts with moderate changes and sudden drifts with severe changes. Motivated by the above challenges, there are two common approaches of existing classification models for data streams: online incremental learning and ensemble learning.

Incremental learning trains a single classifier and updates it with newly arrived data. For example: Domingos and Hulten [5] proposed a very fast Hoeffding tree

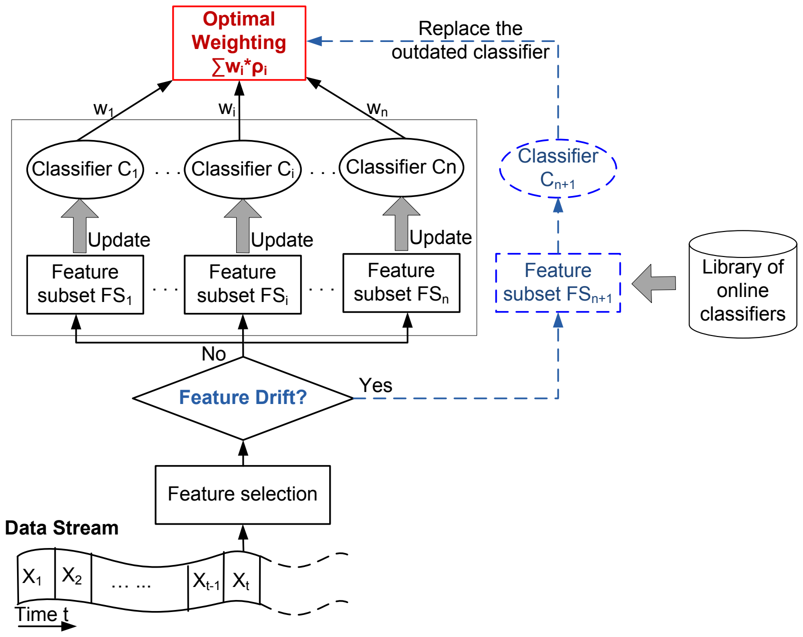

Fig. 1. Ensemble Learning of Feature Drifts

learner (VFDT) for data streams. The VFDT was later extended to CVFDT [12], which can handle the concept drifting streams by constructing alternative nodes and replacing them to the outdated nodes when concept drifts occur. Incremental learning is quite efficient but it cannot adapt to sudden drifts. Ensemble learning, which aims to combine multiple classifiers for boosting classification accuracy, has attracted a lot of research due to its simplicity and good performance. It can manage concept drifts with the following adaptive approaches: (1) using dynamic combiner like a majority vote or weighting combination [18], (2) continuously updating the individual classifiers online [2,15], and (3) changing the ensemble structure by replacing outdated classifiers [16,21]. However, it has high computational complexity as there are many time-consuming processes, eg. generating new classifiers, and updating classifiers.

In this paper, we address the above problems by presenting a novel framework to integrate feature selection techniques and ensemble learning for data streams. To alleviate ensemble updating, we propose a new concept of *"feature drifts"* and use it to optimize the updating process. With a gradual drift, each classifier member is updated in a real-time manner. When a feature drift occurs, which represents a significant change in the underlying distribution of the dataset, we train a new classifier to replace an outdated classifier in the ensemble.

Moreover, feature selection helps to enhance ensemble learning. It not only improves the accuracy of classifier members by selecting the most relevant features and removing irrelevant and redundant features, but also reduces the complexity of the ensemble significantly as only a small subset of feature space is processed. Finally, we propose a heterogeneous ensemble where different types of classifier members are well selected to maximize the diversity of the ensemble [11,22]. Figure 1 gives an overview of our framework. The ensemble consists of many classifiers, each of which has its own feature subset. If there is a feature drift, the ensemble is updated with a new classifier together with a new feature subset; otherwise, each classifier is updated accordingly. To aggregate the classification results of classifier members, we assign each classier a weight w.r.t its performance. This weighting method is proven to minimize the expected added error of the ensemble.

In summary, the following are contributions of our framework which integrates feature selection and ensemble learning techniques for data streams:

- We enhance ensemble learning with feature selection which helps to lessen computational complexity and increase accuracy.
- We propose the definition of feature drifts and explore relationships between feature drifts and concept drifts.
- We significantly increase the accuracy of the ensemble by designing a heterogeneous ensemble with well-chosen member classifiers and an optimal weighting scheme.

2 Related Work

Ensemble learning, a process to construct accurate classifiers from an ensemble of weak classifiers, has attracted extensive research in the past decade. In general, these methods vary in the way of they construct various classifiers and combine their predictions. The *first* step of constructing a group of classifiers can be differentiated according to the dependencies among classifiers. The independent approach trains classifiers randomly and can be easily parallelized, for example, bagging [3], random subspace [19], and random forest [4]. The dependent approach constructs a new classifier while taking advantage of knowledge obtained during the construction of past classifiers, such as AdaBoost [8], AdaBoost.M2 [7], and Gradient boosting [9]. In the *second* step of combining the classifiers' predictions, majority voting is one intuitive method to choose the dominant decision [3,4,19]. As majority voting cannot guarantee that the voting result will be better than the best individual one, the weighting method is introduced which assigns competent classifiers higher weights, such as performance weighting [7,8,9,21], Naive Bayes weighting [6], and entropy weighting [17].

Ensemble learning can also work well with data streams by effectively tackling the challenges of continuous incoming data and concept drifts [2,15,16,18,21]. In [15], Oza *et al.* employed the Poisson distribution to adapt the traditional bagging and AdaBoost techniques for data streams. Bifet *et al.* [2] proposed an ensemble of Adaptive-Size Hoeffding Trees (ASHT) and used a statistical test to detect concept drifts. Street and Kim [18] proposed a streaming ensemble of decision trees using majority voting. Wang *et al.* [21] proposed a carefully weighted ensemble for concept-drifting data streams and proved that the ensemble is more accurate than a single classifier trained on the aggregated data of k sequential chunks. In [16], Sattar *et al.* adapted the traditional one-vs-all (OVA) classifiers for data streams where k individual *concept-adapting very fast decision trees* (CVFDT) [12] are learnt and each one is used to distinguish the instances of one class from the instances of all other classes. However, the above algorithms suffer from high complexity and the inability to adapt to different types of concept drifts.

Feature selection is another important research issue as data streams are usually high-dimensional. Feature selection techniques can be classified into three categories: filter, wrapper and embedded models [14]. The filter model evaluates a feature subset by using some independent measure, which only relies on the general characteristics of data. The wrapper model is attached to a learning algorithm and uses its performance to evaluate a feature subset. A hybrid model takes advantage of the above two models. As data streams require real-time responses, we favor the filter approach due to its simplicity and independence to classification models. Moreover, in data streams, the definition of relevant features is dynamic and restricted to a certain period of time. Features that are previously informative may become irrelevant, and previously rejected features may become important features. Thus, dynamic feature selection techniques are required to monitor the evolution of features. Unfortunately, to the best of our knowledge, there is limited research about the relationship between feature selection and ensemble learning, especially for data streams.

In this paper, we will address this gap between feature selection and ensemble learning and propose a novel framework for integrating feature selection and heterogeneous ensembles. Our framework not only adapts to different kinds of concept drifts properly, but also has low complexity due to its dynamic updating scheme and the support of feature selection techniques.

3 Proposed Framework

In this section, we propose a general framework for ensemble learning for data streams. We assume infinite data streams $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}, \ldots]$ as input in the framework, where $\mathbf{x_t} = [\mathbf{f^1_t}, \mathbf{f^2_t}, \ldots, \mathbf{f^p_t}]^\mathbf{T}$ is a p-dimensional vector arriving at time t. We assume the data streams have c different class labels. For any data vector $\mathbf{x_t}$, it has a class label $\mathbf{y_t} \in \{\mathbf{y_1}, \mathbf{y_2}, \ldots, \mathbf{y_c}\}$. Generally when the dimension p is large, there is often only a small set of key features that is critical for building accurate models for classification.

Data streams tend to evolve over time and so do the key features correspondingly. For ease of discussion, we will use the following definitions:

Definition 1. *A **data source** or a concept is defined as set of prior probabilities of the classes and class-conditional probability density function (pdf):*

$$\mathbf{S} = \{(\mathbf{P}(\mathbf{y_1}), \mathbf{P}(\mathbf{X}|\mathbf{y_1})), \ldots, (\mathbf{P}(\mathbf{y_c}), \mathbf{P}(\mathbf{X}|\mathbf{y_c}))\}. \tag{1}$$

Definition 2. *Given data streams X, every instance x_t is generated by a data source or a concept S_t. If all the data is sampled from the same source, i.e. $S_1 = S_2 = \ldots = S_t = S$, we say that the concept is stable. If for any two time points i and j $S_i \neq S_j$, we say that there is a **concept drift**.*

Definition 3. *Given a feature space $\mathcal{F}$, at time point t, we can always select the most discriminative subset $\widehat{\mathcal{F}}_t \subseteq \mathcal{F}$. If for any two time points i and j $\widehat{F}_i \neq \widehat{F}_j$, we say that there is a **feature drift**.*

Next, we explore the relationship between concept drifts and feature drifts.

Lemma 1. *Concept drifts give rise to feature drifts.*

Proof. Assume that certain feature selection techniques evaluate the discrimination of a feature subset $\mathcal{F}$ at time t by a function:

$$\mathcal{D}(\mathcal{F}_t, t) = \mathcal{D}(P(f^i|y_j), t), \quad f^i \in \mathcal{F}_t \subseteq \{f^1, \ldots, f^p\}, y_j \in \{y_1, \ldots, y_c\} \tag{2}$$

And, there is a feature drift in $[t_i, t_{i+\delta t}]$,

$$\begin{cases} \hat{\mathcal{F}}_t = argmax_{\mathcal{F}_i \subseteq \mathcal{F}} \mathcal{D}(\mathcal{F}_i, t) \\ \mathcal{F}_{t+\delta t} = argmax_{\mathcal{F}_i \subseteq \mathcal{F}} \mathcal{D}(\mathcal{F}_i, t + \delta t) \\ \hat{\mathcal{F}}_t \neq \hat{\mathcal{F}}_{t+\delta t} \end{cases} \tag{3}$$

We can always find a feature $f^a \in \mathcal{F}$ and a class y_b so that $P(f^a|y_b, t) \neq P(f^a|y_b, t + \delta t)$. Else, $P(f^i|y_j, t) = P(f^i|y_j, t + \delta t), \forall(i, j)$, then $\hat{\mathcal{F}}_t = \hat{\mathcal{F}}_{t+\delta t}$. Hence, $P(X|y_b, t) \neq P(X|y_b, t + \delta t) \Rightarrow S_i \neq S_{i+\delta i}$. This denotes a concept drift in the time interval $[t_i, t_{i+\delta t}]$.

Lemma 2. *Concept drifts may not lead to feature drifts.*

Proof. For example, given data streams $\mathbf{X}$ within a time period $[t_i, t_{i+\delta t}]$ we assume the prior probability of a class i and j, $P(y_i)$ and $P(y_j)$, are changed; but their sum $(P(y_i) + P(y_j))$ and other probabilities remain the same. In this scenario, there is a concept drift but no feature drift.

Combining Lemmas 1 and 2, we can conclude that:

Theorem 1. *Feature drifts occur at a slower rate than concept drifts.*

Feature drifts, which are observed in high dimensional data streams, occur no faster than concept drifts. As shown in the overview of our framework Figure 1, we need to modify feature selection techniques to detect feature drifts. The key idea is using feature selection to accelerate ensemble learning and steer its updating process. Moreover, feature selection techniques not only remove irrelevant and redundant features, but also accelerate the learning process by reducing the dimensionality of the data. Thus, the overall performance of the ensemble is improved in terms of accuracy, time and space complexities.

First, we select online classifiers as classifier members as they can be incrementally updated with new data. Then, we construct a heterogeneous ensemble with a capability to adapt to both concept and feature drifts. With gradual drifts, we only need to update the classifier members. With feature drifts, we adjust the ensemble by replacing the outdated classifier with a new one. We further deploy a weighting technique to minimize the cumulative error of the heterogeneous ensemble.

3.1 Feature Selection Block

Feature selection selects a subset of q features from the original p features ($q \leq p$) so that the feature space is optimally reduced. Generally, we expect the feature selection process to remove irrelevant and redundant features. We decide to use FCBF [13] as it is simple, fast and effective. FCBF is a multivariate feature selection method where the class relevance and the dependency between each feature pair are taken into account. Based on information theory, FCBF uses symmetrical uncertainty to calculate dependencies of features and the class relevance. Starting with the full feature set, FCBF heuristically applies a backward selection technique with a sequential search strategy to remove irrelevant and redundant features. The algorithm stops when there are no features left to eliminate.

Symmetrical Uncertainty (SU) uses entropy and conditional entropy values to calculate dependencies of features. If X, Y are random variables, X receives value x_i with probability $P(x_i)$, Y receives value y_j with probability $P(y_j)$; the symmetrical uncertainty between X and Y is:

$$SU(X,Y) = 2\left[\frac{H(X) - H(X|Y)}{H(X) + H(Y)}\right] = 2\left[\frac{I(X,Y)}{H(X) + H(Y)}\right], \quad (4)$$

where $H(X)$ and $H(Y)$ are the entropies of X and Y respectively; $I(X,Y)$ is the mutual information between X and Y; the higher the $SU(X,Y)$ value, the more dependent X and Y are.

We choose to the sliding window version of FCBF so that it has low time and space complexities. Incoming data is stored in a buffer (window) with a predefined size. Next, the matrix of symmetrical uncertainty values is computed to select the most relevant feature subset. The process is performed in a sliding window fashion, and the selected feature subsets are monitored to detect feature drifts. When two consecutive subsets are different, we postulate that a feature drift has occurred.

3.2 Ensemble Block

Heterogeneous Ensemble. When constructing an ensemble learner, the diversity among member classifiers is expected as the key contributor to the accuracy of the ensemble. Furthermore, a heterogeneous ensemble that consists of different classifier types usually attains high diversity [11,23]. Motivated by this

Algorithm 1. Ensemble Learning

Input: A series of infinite streaming data $\mathbf{X} = [\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_t}, \ldots]$, where $\mathbf{x_t}$ is a p dimensional vector $[f_t^1, f_t^2, \ldots, f_t^p]^T$ with a class label y_t arriving at time t, $y_i \in \{y_1, y_2, \ldots, y_c\}$.
A set of l different classifier types, $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_l\}$.
Output: An heterogenous ensemble $\mathcal{E}$.

```
1: Initialize the ensemble $\mathcal{E}$ with $k$ classifiers of each model in $\mathcal{M}$, denoted as
   $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_{k*l}$.
2: while X has more instance do
3:   if $chunk$ is not full then
4:     Add $x_i$ to $chunk$.
5:   else
6:     Perform FCBF to get the relevant and non-redundant feature subset $\varphi_i$.
7:     if $\varphi_i \neq \varphi_{i-1}$ then
8:       Find the best accurate classifier $\mathcal{C}_{best}$ having the smallest aggregated error
         in the ensemble and get its type.
9:       Build a new classifier $\mathcal{C}_{new}$ having the same type with $\mathcal{C}_{best}$, and associated
         with the feature subset $\varphi_i$.
10:      Remove the classifier with the worst accuracy from $\mathcal{E}$.
11:      Add the new created classifier $\mathcal{C}_{new}$ into $\mathcal{E}$.
12:    end if
13:    for each classifier in the ensemble $\mathcal{C}$ do
14:      for each instance $x$ in $chunk$ do
15:        Set $m$ according to Poisson(1)
16:        Update $m$ times each classifier member with $x$.
17:      end for
18:    end for
19:  end if
20: end while
```

observation, we construct a small heterogeneous ensemble rather than a big homogeneous ensemble with a large number of classifiers of the same type, which will compromise speed.

As mentioned, we aim to select online classifiers so that the ensemble can properly adapt to different types of concept drifts. Here, CVFDT [12] and Online Naive Bayes (OnlineNB) are chosen as the basic classifier types, but the framework can work with any classification algorithm. The OnlineNB is an online version of the Naive Bayes classier. When a training instance (x_t, y_t) comes, OnlineNB updates the corresponding prior and likelihood probabilities, $P(y_t)$ & $P(F_i = f_i^t | y_t)$. To classify a testing instance, it applies Bayes' theorem to select the class having the maximum posterior probability as follows:

$$OnlineNB(x_t) = argmax_{y_j} P(y_j) \prod_{i=1}^{n} P(F_i = f_i^t | y_j) \tag{5}$$

Details of the heterogeneous ensemble's learning process are given in Algorithm 1. Given a data stream X and a predefined set $\mathcal{M}$ of different classifier

Algorithm 2. Real Time Classification

```
Input: A new testing unlabeled instance $x_t$.
An ensemble $\mathcal{E}$ of $N$ classifiers, denoted as $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_N$. Every classifier $\mathcal{C}_i$ is associated
with a feature subset $\varphi_{\mathbf{i}}$.
Given a testing instance, every classifier $\mathcal{C}_i$ outputs a probability distribution vector
$\rho_i = [\rho_{i1}, \rho_{i2}, \ldots, \rho_{ic}], (\sum_{j=1}^{c} \rho_{ij} = 1,\ i \in \{1, 2, \ldots, N\})$.
Output: The ensemble 's probability distribution vector for $x_t$.

1: for every classifier $\mathcal{C}_i$ in the ensemble $\mathcal{E}$ do
2:     Project the arriving testing instance$\mathbf{x}_t$ onto a low dimensional feature space $\varphi_{\mathbf{i}}$
       and get $\widehat{\mathbf{x}}_t$.
3:     Compute the probability distribution vector $\rho_i = \mathcal{C}_i(\widehat{\mathbf{x}}_t)$.
4:     Get the aggregated error of $\mathcal{C}_i$, $err_i$, and calculate the weight following the
       Equation 8, $w_i = 1/(err_i + \alpha)$.
5: end for
6: Aggregate all probability distribution vectors as follows:
       $\mathcal{E}(\rho) = \sum_{i=1}^{z} w_i * \rho_i$
7: Normalize and return vector $\mathcal{E}(\rho)$.
```

types, we initialize the ensemble with k classifiers of each type in $\mathcal{M}$. Next, data streams are processed in a sliding window mode and data instances are grouped into predefined-size chunks. When a new chunk arrives, we apply a feature selection technique to find the most discriminative feature subset. If the subset is different from the previous one, there is a feature drift. We would then need to construct a new classifier with the selected feature subset. We add it to the ensemble, and remove the worst classifier if the ensemble is full; the new classifier will be of the same type as the best classifier member which has the smallest aggregated error (lines 7-12). Finally, we employ online bagging [15] for updating classifier members to reduce the variance of the ensemble (lines 13-18).

Ensemble Classification. Based on the research work of Tumer *et. al* [20] and Fumera *et. al* [10], the estimate error of the ensemble are:

$$E_{add}^{ens} = \sum_{k=1}^{N} w_k^2 E_{add}^k + \sum_{k=1}^{N} \sum_{l \neq k} w_l w_k \left[\beta^k \beta^l + \rho^{kl} (\sigma_i^k \sigma_i^l + \sigma_j^k \sigma_j^l)/s^2 \right], \qquad (6)$$

where β^k and σ^k are the bias and the standard deviation of the estimate error ε^k of classifier C_k , ρ^{kl} is the correlation coefficient of the errors ε^k and ε^l.

We assume that classifier members are unbiased and uncorrelated (i.e. $\beta^k = 0, \rho^{kl} = 0, k \neq l$). Then, we have $E_{add}^{ens} = \sum_{k=1}^{N} w_k^2 E_{add}^k, \sum_{k=1}^{N} w_k = 1$. To minimize the added error of the ensemble, the weights of classifier members are set as follows:

$$w_k = (E_{add}^k)^{-1} \left[\sum_{m=1}^{N} (E_{add}^m)^{-1} \right] \qquad (7)$$

When constructing the ensemble classifier, we estimate the added error of every classifier member, E^k_{add}, which is its accumulated error from its creation time to the current time. Moreover, to alleviate the extreme case of $E^m_{add} \approx 0$, we modify the Equation 7 as follows:

$$w_k = (E^k_{add} + \alpha)^{-1} \left[\sum_{m=1}^{N} (E^m_{add} + \alpha)^{-1} \right], \tag{8}$$

where α is a padding value which is empirically set to 0.001.

Details of the ensemble classification process are shown in Algorithm 2. Given the ensemble $\mathcal{E}$ where each member can classify a testing instance and output the result as a probability distribution vector, we use a weighting combination scheme to get the result. First, for each classifier $\mathcal{C}_i$ we project the testing instance x_t onto the subspace $\varphi_\mathbf{i}$. Then, the classifier $\mathcal{C}_i$ processes the projected instance and outputs a probability distribution vector. We attain the aggregated accuracy of $\mathcal{C}_i$ from its creation time, and calculate its optimal weight according to Equation 8 to minimize the expected added error of the ensemble. Finally, the ensemble's result is set as the normalized sum of the weighted classification results of the members (lines 6-7).

4 Experiments and Analysis

4.1 Experimental Setup

For our experiments, we use three synthetic datasets and three real life datasets. The three synthetic datasets, SEA generator (SEA), Rotating Hyperplane (HYP), and LED dataset (LED), are generated from the MOA framework [1]. Concept drifts are generated by moving 10 centroids at a speed of 0.001 per instance in the RBF dataset, and changing 10 attributes at a speed of 0.001 per instance in the HYP dataset. The three real life datasets are: network intrusion (KDD'99[1]), hand-written digit recognition (MNIST[2]), and protein crystallography diffraction (CRYST[3]) datasets. Table 1 shows the characteristics of the six datasets.

We compare our algorithm HEFT-Stream with other prominent ensemble methods: AWE [21] and OnlineBagging [15]. The experiments were conducted on a Windows PC with a Pentium D 3GHz Intel processor and 2GB memory. To enable more meaningful comparisons, we try to use the same parameter values for all the algorithms. The number of classifier members is set to 10 for all the ensemble algorithms, and the chunk size is set to 1000. To simulate the data stream environment, we process all experiments in a practical approach, called *Interleaved-Chunk*. In this approach, data instances are read to form a data chunk. Each new data chunk is first used to test the existing model. Then it is used to update the model and it is finally discarded to save memory.

[1] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[2] http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets
[3] http://ajbcentral.com/CrySis/dataset.html

Table 1. Characteristics of datasets used for evaluation

Name	#Instances	#Attributes	#Classes	Noise	#Selected Features	**Ratio of FS**
SEA	100,000	3	2	10%	2.25	**75%**
HYP	100,000	10	2	5%	6.71	**67.13%**
LED	100,000	24	3	10%	15.84	**65.98%**
KDD'99	494,022	34	5	N/A	2.33	**6.87%**
MNIST	60,000	780	10	N/A	30.77	**3.95%**
CRYST	5,500	1341	2	N/A	7.49	**0.56%**

4.2 Experimental Results

As AWE and OnlineBagging are homogeneous ensembles and can only work with one classifier type, we set different classifier types for these ensembles accordingly. For AWE, we set classifier members as Naive Bayes and C4.5, which are recommended by the authors [21], and denoted as *AWE(NB)* and *AWE(C4.5)*. For OnlineBagging, we set its classifier members as OnlineNB and CVFDT, and denoted as *Bagging(OnlineNB)* and *Bagging(CVFDT)* respectively. We conduct the experiments ten times for each dataset and summarize their average accuracy and running times in Table 2. Readers may visit our *website*[4] for algorithms' implementation, more experimental results, and detailed theoretical proofs.

We observe that the AWE ensemble has the worst accuracy and the longest running time. This is because the AWE ensemble uses traditional classifiers as its members and trains them once; when there are concept drifts, these members become outdated and accuracy is degraded. Moreover, the AWE ensemble always trains a new classifier for every upcoming chunk, and this increases processing time. The OnlineBagging has better performance but it largely depends on the classifier type. For example, Bagging(OnlineNB) is more accurate and faster than Bagging(CVFDT) for the LED dataset but Bagging(OnlineNB) become less precise and slower than Bagging(CVFDT) for the KDD'99 dataset. It is also noteworthy that both AWE and OnlineBagging do not work well with high dimensional datasets, such as MNIST and CRYST.

Our approach, HEFT-Stream, addresses the above problems and achieves better performance than WCE and OnlineBagging. It achieves the best accuracy values and the lowest running time for most datasets. HEFT-Stream continuously updates classifier members with gradual drifts, and only trains new classifiers whenever there are feature drifts or sudden drifts. This property not only enables HEFT-Stream to adapt to different types of concept drifts but also conserve computational resources. Furthermore, HEFT-Stream can dynamically change the ratio among classifier types to adapt to different types of datasets; when a particular classifier type works well with a certain dataset, its ratio in the ensemble will be increased. Finally, with integrated feature selection capability, HEFT-Stream only works with the most informative feature subsets which improves accuracy and reduces processing time. The last column of the Table 1

[4] http://www3.ntu.edu.sg/home2008/nguy0105/heftstream.html

Table 2. Comparisons of AWE(NB), AWE(C4.5), Bagging(OnlineNB), Bagging(CVFDT), and HEFT-Stream. Time is measured in seconds. For each dataset, the highest accuracy value is **boldfaced**, and the lowest running time is <u>underlined</u>.

Dataset	AWE(NB)		AWE(C4.5)		Bagging(OnlineNB)		Bagging(CVFDT)		HEFT-Stream	
	Acc	Time	Acc	Time	Acc	Time	Acc	Time	Acc	Time
SEA	88.08	6.61	88.12	26.08	87.91	<u>2.9</u>	89.12	21.47	**89.28**	22.65
HYP	87.94	19.42	72.72	27.40	86.92	<u>8.07</u>	88.90	40.41	**89.18**	12.42
LED	73.91	74.33	72.13	40.34	73.93	<u>26.21</u>	73.79	83.00	**74.07**	28.02
KDD'99	95.09	280.33	94.68	281.33	92.95	230.3	**97.75**	209.6	96.37	<u>142.0</u>
MNIST	9.87	2054.0	78.49	1246.7	9.87	1286.00	21.13	1456.00	**79.36**	<u>439.00</u>
CRYST	53.70	40.38	83.30	147.00	54.28	57.63	76.18	101.33	**83.52**	<u>37.10</u>
Average	68.10	412.51	81.57	294.80	67.64	268.53	74.48	318.65	**85.30**	<u>113.53</u>

shows the ratios of the selected features to the full feature sets for all datasets. We realize that feature selection techniques are very useful for high dimensional datasets. For example, the percentages of the selected features are 3.95% for the MNIST dataset, and only 0.56% for the CRYST dataset.

5 Conclusions

In this paper, we proposed a general framework to integrate feature selection and heterogeneous ensemble learning for stream data classification. Feature selection helps to extract the most informative feature subset which accelerates the learning process and increases accuracy. We first apply feature selection techniques on the data streams in a sliding window manner and monitor the feature subset sequence to detect feature drifts which represent sudden concept drifts. The heterogeneous ensemble is constructed from well-chosen online classifiers. The ratios of classifier types are dynamically adjusted to increase the ensemble 's diversity, and allows the ensemble to work well with many kinds of datasets. Moreover, the ensemble adapts to the severity of concept drifts; we update the online classifier members for gradual drifts, and replace an outdated member by a new one for sudden drifts. We have conducted extensive experiments to show that our ensemble outperforms state-of-the-art ensemble learning algorithms for data streams.

In our future work, we will investigate more intelligent methods to adjust the ratios of classifier types as well as the ensemble size. We will continue to examine the relationship between concept and feature drifts and develop a metric to quantify concept drifts and use it to further adapt ensembles to achieve better accuracy.

References

1. Bifet, A., Holmes, G., Kirkby, R.: Moa: Massive online analysis. The Journal of Machine Learning Research 11, 1601–1604 (2010)
2. Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., Gavald, R.: New ensemble methods for evolving data streams. In: 15th ACM SIGKDD, pp. 139–148. ACM (2009)

3. Breiman, L.: Bagging predictors. The Journal of Machine Learning Research 24(2), 123–140 (1996)
4. Breiman, L.: Random forests. The Journal of Machine Learning Research 45(1), 5–32 (2001)
5. Domingos, P., Hulten, G.: Mining high-speed data streams. In: The Sixth ACM SIGKDD, pp. 71–80. ACM (2000)
6. Domingos, P., Pazzani, M.: On the optimality of the simple bayesian classifier under zero-one loss. The Journal of Machine Learning Research 29(2-3), 103–130 (1997)
7. Eibl, G., Pfeiffer, K.-P.: Multiclass boosting for weak classifiers. The Journal of Machine Learning Research 6, 189–210 (2005)
8. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: The 13th ICML, pp. 148–156 (1996)
9. Friedman, J.H.: Stochastic gradient boosting. Computational Statistics & Data Analysis 38(4), 367–378 (2002)
10. Fumera, G., Roli, F.: A theoretical and experimental analysis of linear combiners for multiple classifier systems. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(6), 942–956 (2005)
11. Hsu, K.-W., Srivastava, J.: Diversity in Combinations of Heterogeneous Classifiers. In: Theeramunkong, T., Kijsirikul, B., Cercone, N., Ho, T.-B. (eds.) PAKDD 2009. LNCS, vol. 5476, pp. 923–932. Springer, Heidelberg (2009)
12. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: ACM SIGKDD, pp. 97–106. ACM (2001)
13. Lei, Y., Huan, L.: Feature selection for high-dimensional data: A fast correlation-based filter solution. In: The 20th ICML, pp. 856–863 (2003)
14. Liu, H., Yu, L.: Toward integrating feature selection algorithms for classification and clustering. IEEE Transactions on Knowledge and Data Engineering 17(4), 491–502 (2005)
15. Oza, N.C.: Online bagging and boosting. In: 2005 IEEE International Conference on Systems, Man and Cybernetics, vol. 3, pp. 2340–2345. IEEE (2005)
16. Sattar, H., Ying, Y., Zahra, M., Mohammadreza, K.: Adapted one-vs-all decision trees for data stream classification. IEEE Transactions on Knowledge and Data Engineering 21, 624–637 (2009)
17. Shen, C., Li, H.: On the dual formulation of boosting algorithms. IEEE Transactions on Pattern Analysis and Machine Intelligence 32(12), 2216–2231 (2010)
18. Street, W.N., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: The 7th ACM SIGKDD, pp. 377–382. ACM (2001)
19. Tin Kam, H.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
20. Tumer, K., Ghosh, J.: Linear and order statistics combiners for pattern classification. Springer (1999)
21. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: ACM SIGKDD, pp. 226–235. ACM (2003)
22. Woods, K., Philip Kegelmeyer, J.W., Bowyer, K.: Combination of multiple classifiers using local accuracy estimates. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(4), 405–410 (1997)
23. Zhenyu, L., Xindong, W., Bongard, J.: Active learning with adaptive heterogeneous ensembles. In: The 9th IEEE ICDM, pp. 327–336 (2009)