Published in final edited form as:
Proc SIAM Int Conf Data Min. 2014 April ; 2014: 722–730. doi:10.1137/1.9781611973440.83.

Classifying Imbalanced Data Streams via Dynamic Feature Group Weighting with Importance Sampling

Ke Wu[*], **Andrea Edwards**[*], **Wei Fan**[†], **Jing Gao**[‡], and **Kun Zhang**[*]

[*]Department of Computer Science, Xavier University of Louisiana

[†]Huawei Noah Ark's Lab, david.fanwei@huawei.com

[‡]Department of Computer Science and Engineering, University at Buffalo, jing@buffalo.edu

Abstract

Data stream classification and imbalanced data learning are two important areas of data mining research. Each has been well studied to date with many interesting algorithms developed. However, only a few approaches reported in literature address the intersection of these two fields due to their complex interplay. In this work, we proposed an importance sampling driven, dynamic feature group weighting framework (DFGW-IS) for classifying data streams of imbalanced distribution. Two components are tightly incorporated into the proposed approach to address the intrinsic characteristics of concept-drifting, imbalanced streaming data. Specifically, the ever-evolving concepts are tackled by a weighted ensemble trained on a set of feature groups with each sub-classifier (i.e. a single classifier or an ensemble) weighed by its discriminative power and stable level. The un-even class distribution, on the other hand, is typically battled by the sub-classifier built in a specific feature group with the underlying distribution rebalanced by the importance sampling technique. We derived the theoretical upper bound for the generalization error of the proposed algorithm. We also studied the empirical performance of our method on a set of benchmark synthetic and real world data, and significant improvement has been achieved over the competing algorithms in terms of standard evaluation metrics and parallel running time. Algorithm implementations and datasets are available upon request.

Keywords

Data stream classification; Class imbalance; Ensemble weighting; Feature group ensemble; Importance sampling

1 Introduction

Recent years have witnessed a dramatic increase in our ability to collect data continuously. Most of these data are characterized by fast arrival, high volume and infinite length, and thus are referred to as data streams. Applications involving streaming data are ubiquitous. Typical examples include stock market trend analysis, surveillance monitoring and so on. In

kwu@xula.edu, aedwards@xula.edu, kzhang@xula.edu

[3]$\mathscr{F} = \{F_1, F_2, \ldots, F_r\}$

traditional classification tasks, data are assumed to be static. That is, the underlying concept that projects the attributes to the class labels is unchanging. However, in data streams, such a concept is not stable but drift over time due to changes in the environment. For example, the stock market fluctuates daily as a result of economy, corporate earnings, government monetary policy, etc. Often the incessant changes will outdate the classifier learned from old data on a temporal basis, and updating or retraining the model is indispensable. This is generally known as concept drift. Based on Bayes's theorem, three kinds of concept drifts, including feature change, conditional change and dual change, are formally defined and analyzed in [7]. So far, numerous classification algorithms have been proposed with most of them focusing on the drifting concepts inherent in ever-evolving data streams.

Class imbalance is a practical problem and usually occurs when there are fewer instances in the target class (positive or minority class) compared to other classes (negative or majority class). Class imbalance can be introduced either due to the nature of an application, or the limitations in collecting a representative data set as a result of the cost or privacy issues. Class imbalance presents several challenges in learning tasks, including skewed class distribution, data insu ciency and more complicated concepts. As those challenges interfere with concept drifts in the context of data streams, it becomes an even more severe and compound problem. For example, over a data stream, the time interval of receiving a positive instance can be unpredictably long. Thus, it is always hard to collect su cient positive examples at a timestamp to unbiasedly infer the true function describing this class. In addition, successive positive examples may be drawn from arbitrarily distinct distributions. Therefore, distributional discrepancy can exist between any two positive examples not received at the same time. Algorithms designed for classifying skewed data streams must take those imbalance-posed challenges into account, while equipped with effective mechanisms for handling drifting concepts.

To date, data stream classification on skewed class distribution is a relatively unexplored area, and not much work has been reported. Existing methods [7, 8, 2, 4] do not well solve the class imbalance problem in the concept-drifting data steam scenario due to the following observations. First, to augment the minority set in the training data chunk, current approaches either aggregate all minority instances over time, or select part of the instances by some similarity or distance measures with rigorous thresholds. The former strategy implicitly assumes that there is no drift in the underling concept of the minority class, while the latter may fail to identify and proliferate adequate minority examples when the distribution is extremely skewed and(or) the underlying concepts drift rather heavily. Second, most of the methods [7, 2, 4] mainly count on the most recent data to handle the concept drifts, which actually disregards feature change that could be assessed by the just-arrived, to-be-predicted chunk. [8] addressed this issue. However, by focusing only on the treatment of feature change via the unlabeled data, this approach is somewhat heuristic and lacks a theoretical basis for its design.

In view of these limitations, we introduce a novel framework for classifying imbalanced data streams arriving in batches. Specifically, to augment the positive examples in the training data chunk arrives at any timestamp t, a sliding window of limited size temporally pushes into the training chunk the positive instances received in previous batches. The

amplified training set is then projected onto a set of pre-defined feature groups for potential feature drift detection. In each feature group, the underlying skewed distribution is rebalanced by importance sampling of the positive examples. A light-weight hypothesis (i.e., a single classifier or an ensemble) is then built on each feature group with the balanced distribution. All feature-group hypotheses are finally combined as an ensemble for prediction purposes, with each hypothesis dynamically weighed by two factors. One is the hypothesis's discriminative power tuned on a small amount of the labeled data in the most recent chunk. The other is its stable level estimated by the distributional similarity of the feature group from which the hypothesis is built with respect to the corresponding feature set in the just arrived, to-be-predicted chunk. The former factor reflects a hypothesis's ability to adapt to the conditional change, while the latter indicates a hypothesis's ability in response to the feature change, whereby the drifting concepts are neatly addressed in a timely manner.

The main contributions of this paper are summarized as follows.

- We propose an importance sampling driven, dynamic feature group weighting framework (DFGW-IS) for data stream classification with skewed distribution.
  - the underlying ever-evolving concepts are tackled by a weighted ensemble trained on a set of feature groups with each sub-classifier weighed by its discriminative power and stable level.
  - The un-even class distribution, on the other hand, is typically battled by the sub-classifier built in a specific feature group with the underlying distribution rebalanced by the importance sampling technique.
- We conduct time complexity analysis and derive the theoretical upper bound for the generalization error of the proposed algorithm.
- Extensive empirical results on multiple synthetic and real-world benchmark datasets demonstrate that the proposed framework statistically significantly outperforms the competing methods on multiple evaluation metrics.

2 Problem Setting

Let $\mathscr{X}=\mathbb{R}^d$ be the feature space and $\mathscr{Y}=\{+1,-1\}$ be the class label. Consider a data stream $\mathscr{S}$ comes in batches or chunks $\left\{\mathscr{S}^{(1)},\ldots,\mathscr{S}^{(t)},\ldots\right\}$, where $\mathscr{S}^{(t)}$ denotes the data chunk at timestamp *t*. Let $\mathscr{S}^{(t)}=\left\{\mathbf{x}_i^{(t)},y_i^{(t)}\right\}_{i=1}^{N_t}$ and $\left\{\mathbf{x}_i^{(t)},y_i^{(t)}\right\}_{i=1}^{N_t}\sim D_t$, where $\mathbf{x}_i^{(t)}\in\mathscr{X}$, $y_i^{(t)}\in\mathscr{Y}$, N_t is the number of instances in batch *t*, and D_t is the underlying distribution of batch *t*. D_t varies over time due to the type and/or the degree of the drifting concepts. Assume that there are two classes in batch *t*, i.e., positive class $\mathscr{P}^{(t)}$ and negative class $\mathscr{N}^{(t)}$, and the size of positive class is much smaller than that of negative class, that is, $|\mathscr{P}^{(t)}|<<|\mathscr{N}^{(t)}|$. In such a case, $\mathscr{S}^{(t)}$ is said to be imbalanced. Let $\mathscr{S}^{(t+1)}$ be the newly arrived chunk with unknown labels. Our learning task is to leverage the data chunks received so far (including the feature information carried by $\mathscr{S}^{(t+1)}$ to classify $x_i^{(t+1)}$ in $\mathscr{S}^{(t+1)}$, with high prediction accuracy

achieved over the positive instances while maintaining reasonable accuracy for the negative class. Table 1 summarizes the major notations used in the paper.

3 Method

In this section, we present the proposed framework DFGW-IS. The overall learning flow is outlined in Figure 1. Two tightly integrated components, the dynamically weighed feature group ensemble for fast adaptation to changes and the importance sampling driven sub-classifier to combat class imbalance, are discussed separately.

3.1 Drifting Concept Adaptation: Dynamically Weighed Feature Group Ensemble

In DFGW-IS, adaptation to changes is achieved through dynamically weighing the sub-classifiers of an ensemble trained on a set of feature groups.

Defined as $F_1, F_2, \ldots, F_r \hat{} F$ with $F_0 = F$, where $F_i \neq F_j$ when $i \neq j$, a set of possibly overlapping feature groups can be decided in advance according to some prior or domain knowledge regarding any possible feature change in a data stream. If such information is not available, they can be generated randomly. These randomly generated feature subspaces can provide multiple views into the data, and ensembles built on them have been shown to perform comparable to those built through the data partitioning methods [13]. Meanwhile, to reduce the bias that would be introduced through those random feature subspaces, we also include a full feature group F_0 in the set. This design, from another aspect, accommodates the scenario when there is no feature change. Once the feature groups are determined, they remain unchanged in the entire learning process.

The weight of each sub-classifier trained over a feature group is determined by two factors, its discriminative power and stable level. The discriminative power can be estimated by solving a statistical optimization problem as described below. Specifically, we divide the most-recent data chunk into a training set and a holdout set. 85% of the data is used for training $\mathscr{L}_{tr}^{(t)}$[1] and the rest, $\mathscr{L}_{ho}^{(t)}$, is for the estimation purposes.

One can project $\mathscr{L}_{tr}^{(t)}$ to a particular feature group F_i, and obtain a hypothesis (or sub-classifier) $h_i^{(t)}$ via minimizing some loss functions. The holdout set $\mathscr{L}_{ho}^{(t)}$ is then used to estimate the discriminative power $\overrightarrow{W}_d$ of those models. Formally speaking, we estimate the hypothesis's discriminative power with respect to each instance in the holdout set by solving the following convex optimization problem.

$$\sum_{\left(\mathbf{x}_i^{ho}, y_i^{ho}\right) \in \mathscr{L}_{ho}^{(t)}} C_i \cdot \Delta \left(\sum_{j=1}^{r} w_j h_j^{(t)} \left(\mathbf{x}_i^{ho} \right), y_i^{ho} \right) \quad (3.1)$$

[1] In practice, besides those data from the most-recent chunk, training set also contains other positive instances collected by a temporal sliding window to augment the positive set.

subject to $\Sigma_j w_j = 1$ and $w_j \geq 0$ where C_i is the misclassification cost of instance i. The higher the value of w_j, the more discriminative the hypothesis $h_j^{(t)}$ is. This optimization is due to the following motivation. Since $\mathscr{L}_{ho}^{(t)}$ also carries the same degree of skewness, misclassifying a positive instance in $\mathscr{L}_{ho}^{(t)}$ will cause a hypothesis to be penalized more than misclassifying a negative instance. As a result, the discriminative power of a hypothesis should be determined by its performance on each instance in $\mathscr{L}_{ho}^{(t)}$, rather than its overall performance on the entire set of $\mathscr{L}_{ho}^{(t)}$. This design takes the class imbalance issue into account and is different from other practices dealing with balanced data streams [14]. In the implementation, we use the logistic loss function, i.e., $\Delta(f(x), y) = \log(1 + \exp(-yf(x)))$. The misclassification cost of a positive instance is set as $\left\lfloor \frac{N_{ho}^-}{N_{ho}^+} \right\rfloor$, where N_{ho}^+ (or N_{ho}^-) is the number of positive (or negative) instances in the holdout set. The misclassification cost for a negative instance is 1.

On the other hand, the stable level of a sub-classifier or hypothesis $h_i^{(t)}$ is estimated by the similarity between its training distribution D_t[2] and its test distribution D_{t+1}. The feature drift degree between D_t and D_{t+1} can indicate the stable level of $h_i^{(t)}$, since a classifier trained on a stable feature group behaves more consistently than that built on an unstable group[12]. Thus a classifier with consistent performance should have high liability or weight. In this framework, we use distributional similarity measured by Hellinger distance[8] to indicate a sub-classifier's stable level, which is formulated as follows.

$$W_s^i = 1 - \frac{d_H\left(D_t^{F_i}, D_{t+1}^{F_i}\right)}{\sqrt{2}} \quad (3.2)$$

$$d_H = \left(D_t^{F_i}, D_{t+1}^{F_i}\right) = \frac{1}{|F_i|}\sum_{f=1}^{|F_i|} d_H\left(D_{t,f}^{F_i}, D_{t+1,f}^{F_i}\right) \quad (3.3)$$

where $D_t^{F_i}$ is the distribution defined over F_i at timestamp t, and $D_{t,f}^{F_i}$ is the distribution for feature f defined over F_i at timestamp t.

Since in reality, we can only access a limited sample of instances from those distributions, Hellinger distance is computed via its discrete version as defined below.

$$d_H\left(D_{t,f}^{F_i}, D_{t+1,f}^{F_i}\right) = \sqrt{\sum_{i=1}^{k}\left(\sqrt{p_i} - \sqrt{q_i}\right)^2} \quad (3.4)$$

[2]Precisely, $h_i^{(t)}$ is trained on two sets of samples. One are the samples of $\mathscr{S}^{(t)}$ from distribution D_t. The other are the positive instances from the previous batches whose distributions are similar to D_t due to the importance sampling technique.

where k is the number of feature values, $D_{t,f}^{F_i} = (p_1, \ldots, p_k)$ and $D_{t+1,f}^{F_i} = (q_1, \ldots, q_k)$. The distribution divergence of continuous features can be measured by first discretizing the continuous feature into multiple equal intervals via the Binning technique. The pseudo code of the proposed framework is presented in Algorithms 3.1 and 3.2.

ALGORITHM 3.1. **Train_DFGW**

Input: current timestamp t, current data chunk $\mathcal{S}^{(t)}$, window size threshold δ, set of positive instances $\mathcal{P}$, feature group set $\mathcal{F}^3$, sub-classifier LearnH

Output: hypothesis vector $\vec{H}$, discriminative power vector $\vec{W}_d$, updated set of positive instances $\mathcal{P}$

Split $\mathcal{S}^{(t)}$ into two sets, i.e. $\mathcal{P}^{(t)}$ and $\mathcal{N}^{(t)}$
if $|\mathcal{P}| + |\mathcal{P}^{(t)}| > \delta$ **then**
  Let s be the earliest timestamp for set $\mathcal{P}$
  $s = s + 1$
  $\mathcal{P} = \bigcup_{i=s}^{t-1} \mathcal{P}^{(i)}$
end if
$\mathcal{P} = \mathcal{P} \cup \mathcal{P}^{(t)}$
Construct the training set $\mathcal{L}_{tr}^{(t)}$ and holdout set $\mathcal{L}_{ho}^{(t)}$ on $\mathcal{P}$ and $\mathcal{N}^{(t)}$
$\vec{H} = [\,]$
for $i \leftarrow 0$ to $|\mathcal{F}|$ **do**
  Project $\mathcal{L}_{tr}^{(t)}$ to feature group F_i and obtain $\mathcal{L}_{i,tr}^{(t)}$
  Train $h_i^{(t)}$ on $\mathcal{L}_{i,tr}^{(t)}$ via LearnH
  $\vec{H} = [\vec{H}, h_i^{(t)}]$
end for
Obtain discriminative power vector $\vec{W}_d$ based on Eq. 3.1
return $\vec{H}$, $\vec{W}_d$, $\mathcal{P}$

ALGORITHM 3.2. **Test_DFGW**

Input: test data chunk $\mathcal{S}^{(t+1)}$, feature group set $\mathcal{F}$, hypothesis vector $\vec{H}$, discriminative power vector $\vec{W}_d$

Output: posterior probabilities of instances in $\mathcal{S}^{(t+1)}$

$\vec{W}_s = [\,]$
for $i \leftarrow 0$ to $|\mathcal{F}|$ **do**
  Calculate the stable level W_s^i of sub-classifier $h_i^{(t)}$ based on Eq. 3.2
  $\vec{W}_s = [\vec{W}_s, W_s^i]$
end for
$\vec{\alpha} = \lambda \vec{W}_s + (1 - \lambda)\vec{W}_d$
for $j \leftarrow 1$ to $|\mathcal{S}^{(t+1)}|$ **do**
  $prob_j = \vec{\alpha} \cdot \vec{H}(\mathbf{x}_j)$
end for
return $\{prob_j\}$

3.2 Combat Imbalance: Importance Sampling Driven Sub-classifier

In our problem setting, the number of negative instances is much su cient to build an accurate model, while the positive instances always need to be amplified to balance the current training data. Therefore, the primary task is, for each feature group, how to build an imbalance-resistant model using the wisely-selected positive instances by considering the restrictions of data stream mining.

First, we proposed to use a sliding window of limited size to collect the positive instances temporally similar to the most recent batch. The size of the window is determined by a predefined threshold δ, which can be set according to the current system memory usage or the users' specifications. In this way, we can control or guarantee a reasonable memory consumption. This design is motivated by the following considerations. First, over the course of time, ancient positive examples could be very different from and irrelevant to those in the recent data chunk due to the drift of the underlying concept. Training models on such data would introduce undesired bias and whereby greatly impairing the models' performance. Second, memory is limited. Although the positive instances are very sparse in each batch, as the time approaches infinity, the total number could still be huge and even surpasses the negative instances in the recent chunk. Such a training set can not reside in the limited memory to build the model. The implementation of the sliding window for positive instances is presented in lines 2-7 of Algorithm 3.1.

Second, in the current training set, the positive instances collected over time by the sliding window should be weighed differently according to the similarity between the distribution of the most recent batch (i.e. D_t) and the distribution from which an instance is generated. Specifically, we assign an equal weight 1 to the positive instances in the most recent batch, and the weight of a positive instance from previous batches is determined by Eq. 3.5, where $\beta_i^{(t)}$ measures the distributional similarity for an instance.

$$w_i^{(t)}=1/\left(1+e^{-\left(\beta_i^{(t)}-0.5\right)}\right) \quad (3.5)$$

According to the diversity-focus rule for sampling weights[9], this formula can guarantee that examples with high weights will be sampled frequently and those with low weights have chance to be sampled.

In the following, we elaborate how to decide $\beta_i^{(t)}$ via the principle of importance sampling. let $h_i^{(t)}=p\left(y|\mathbf{x};\theta_i\right)$ be a predictive model parameterized by θ_i. To optimize θ_i, we usually use the following criterion.

$$R\left(\theta_i\right)=\int\int\Delta\left(\mathbf{x},y;\theta_i\right)D_t\left(\mathbf{x},y\right)dyd\mathbf{x} \quad (3.6)$$

When the labelled instances are drawn from a distribution different from D_t, we can rewrite the above formula as follows with the concentration on the positive class.

$$R(\theta_i) = R(\theta_i, +) + R(\theta_i, -) \quad (3.7)$$

where $R(\theta_i, +) = \int \Delta(\mathbf{x}, +; \theta_i) \frac{D_t(\mathbf{x}, +)}{D'(\mathbf{x}, +)} d\mathbf{x}$ and $R(\theta_i, -) = \int \Delta(\mathbf{x}, -; \theta_i) D_t(\mathbf{x}, -) d\mathbf{x}$. Note that $D'(\mathbf{x}, +) = \sum_{k=t-l+1}^{t} \mathbb{1}(T(\mathbf{x}), k)$. $D_k(\mathrm{x}, +)$, where $T(\mathrm{x})$ denotes the timestamp of x.

Next, we discuss how to approximate $R(\theta_i, +)$ by the limited samples in our scenario. In the empirical risk minimization(ERM) framework, the integral $R(\theta_i, +)$ can be estimated by the following empirical risk.

$$\hat{R}(\theta_i, +) = \frac{1}{Z} \sum_{i=1}^{n_+} \frac{D_t(\mathbf{x}_i, +)}{D'(\mathbf{x}_i, +)} \Delta(\mathbf{x}_i, +) \quad (3.8)$$

where Z is a normalization factor.

Let $\beta_i^{(t)} = \frac{D_t(\mathbf{x}_i, +)}{D'(\mathbf{x}_i, +)}$. Similar to the naive Bayesian learning, we assume that the features are independent for $D_k(\mathrm{x}_i|+)$. Consequently, we can rewrite $\beta_i^{(t)}$ as

$\beta_i^{(t)} = \frac{1}{\sum_{k=t-l+1}^{t} \mathbb{1}(T(\mathbf{x}), k) \cdot \tau_k^{(t)} \cdot \prod_{j=1}^{d} \gamma_{ijk}^{(t)}}$ where $\gamma_{ijk}^{(t)} = \frac{D_k(x_{ij}|+)}{D_t(x_{ij}|+)}$ and $\tau_k^{(t)} = \frac{D_k(+)}{D_t(+)}$.

For a categorical feature, $D_k(x_{ij} \mid +)$ can be estimated by the ratio of the number of instances in class + having the value x_{ij} for the j-th feature to the number of instances in class + with the same feature. If a feature is continuous, we typically assume that it has a Gaussian distribution. Thus, $D_k(x_{ij} \mid +)$ can be estimated by Eq. 3.9

$$D_k(x_{ij}|+) = \frac{1}{\sqrt{2\pi v_j^{(k)}}} e^{-\frac{\left(x_{ij} - \mu_j^{(k)}\right)^2}{2v_j^{(k)}}} \quad (3.9)$$

where $\mu_j^{(k)} = \sum_{i=1}^{N_k} \mathbb{1}\left(y_i^{(k)}, +\right) x_{ij}^{(k)} / \sum_{i=1}^{N_k} \mathbb{1}\left(y_i^{(k)}, +\right)$ and
$\nu_j^{(k)} = \sum_{i=1}^{N_k} \left(\mathbb{1}\left(y_i^{(k)}, +\right)\left(x_{ij}^{(k)} - \mu_j^{(k)}\right)\right)^2 / \left(\sum_{i=1}^{N_k} \mathbb{1}\left(y_i^{(k)}, +\right) - 1\right)$

Lastly, based on the obtained importance weights for positive instances, we can choose to build either a single model or an ensemble according to the specific learning needs. As ensembles often achieve better predictive performance than individual models via variance reduction, in our implementation, we trained a light-weight ensemble on the current training set projected to a particular feature group. Specifically, we generated multiple subsets of positive samples using importance weights. A hypothesis is then built on each of these subsets plus a negative sample subset achieved through under-sampling with replacement. The averaged combination of these hypotheses is the imbalance-proof model for a specific feature group. The pseudo code of learning such an ensemble is summarized in algorithm 3.3.

ALGORITHM 3.3. **LearnH**
Input: training set $\mathcal{L}$, ensemble size K
Output: sub-classifier H

1: Split $\mathcal{L}$ into positive set $\mathcal{L}_p$ and negative set $\mathcal{L}_n$
2: Calcalute $\vec{w}$ associated with each positive instance base on Eq. 3.5
3: **for** $i \leftarrow 1$ to K **do**
4: Based on positive weights $\vec{w}$, draw a sample $\mathcal{P}_i$ of size $|\mathcal{L}_p|$ using importance sampling
5: Draw a sample $\mathcal{N}_i$ with replacement ($|\mathcal{N}_i| = |\mathcal{L}_p|$)
6: $\mathcal{L}_i = \mathcal{P}_i \cup \mathcal{N}_i$
7: $h_i = \arg\min \sum_{(\mathbf{x},y)\in\mathcal{L}_i} \Delta(h(\mathbf{x}, y))$
8: **end for**
9: $H = \frac{1}{K}\sum h_i$
10: return H

4 Time Complexity Analysis

Suppose the base learner is a decision tree. Its time complexity is $\mathcal{O}(dN \quad log \quad N)$, where *d* is the data dimensionality and *N* is the total number of training points. Figure 2 presents the architecture of the parallel implementation of DFGW-IS. We can observe that the e ciency of the parallel training depends on the speed of the slowest sub-module, which is S_{0j} in F_0. In our framework, the size of training data of each sub-module S_{ij} is bounded by 2δ. Therefore, the upper bound of the time complexity of S_{0j} is $\mathcal{O}(2d\delta \quad log \quad 2\delta)$. In the training phrase, in addition to the sub-module training, our framework needs to tune the discriminative power of the model obtained from each F_i. The tuning may cost a small amount of time, since it runs over a small portion of labelled data $\mathcal{L}_{ho}^{(t)}$. For example, the time complexity of the tuning through L-BFGS with a block constraint is $\mathcal{O}\left(m\,(r+1)\,|\mathcal{L}_{ho}^{(t)}|\right)$, where *m* is the number of iterations. *m* often is as small as 3-10. As a result, the total time complexity for training phrase is $\mathcal{O}(2d\delta \quad log \quad 2n\delta) + \mathcal{O}\left(m\,(r+1)\,|\mathcal{L}_{ho}^{(t)}|\right)$. In the testing stage, the instance prediction is conducted in batch, and thus can be done in linear time proportional to and dominated by the number of the test instances.

5 Theoretical analysis

In this section, we present the theoretical analysis of the proposed DFGW framework. In order to derive the upper bound of the generalization error, we define the following divergence called $d_{\mathcal{H}\otimes\mathcal{H}}$-distance, for our main result.

Definition 5.1. ($\mathbf{d}_{\mathcal{H}\otimes\mathcal{H}}$-***distance***) *Let $\mathcal{H}$ be a hypothesis space. The $d_{\mathcal{H}\otimes\mathcal{H}}$-distance between two distributions D^1 and D_2 over $\mathcal{X}$ is defind as*

$$d_{\mathscr{H}\otimes\mathscr{H}}\left(D^1,D^2\right)=\sup_{h,h'\in\mathscr{H}}|\epsilon_{D^1}\left(h,h'\right)-\epsilon_{D^2}\left(h,h'\right)|$$

where $\epsilon_{D^1}\left(h,h'\right)=E_{x\sim D^1}\left[|h\left(x\right)-h'\left(x\right)|\right]$ and $\epsilon_{D^2}\left(h,h'\right)=E_{x\sim D^2}\left[|h\left(x\right)-h'\left(x\right)|\right]$

Lemma 5.1. ***(Symmetry)****. For any distributions* D^1 *and* D^2*, we have*

$$d_{\mathscr{H}\otimes\mathscr{H}}\left(D^1,D^2\right)=d_{\mathscr{H}\otimes\mathscr{H}}\left(D^2,D^1\right).$$

Proof. The result can be naturally derived based on Definition 5.1.

Lemma 5.2. ***(Convexity)****. For any distributions* D, D^1, D^2 *and* D'*, where* $D'=\gamma D^1+(1-\gamma)D^2$*, we have*

$$d_{\mathscr{H}\otimes\mathscr{H}}\left(D,D'\right)\leq\gamma d_{\mathscr{H}\otimes\mathscr{H}}\left(D,D^1\right)+(1-\gamma)\,d_{\mathscr{H}\otimes\mathscr{H}}\left(D,D^2\right),0\leq\gamma\leq1.$$

Lemma 5.3. ***(Triangle Inequality)****. For any distributions* D^1, D^2 *and* D'*, we have*

$$d_{\mathscr{H}\otimes\mathscr{H}}\left(D^1,D^2\right)\leq d_{\mathscr{H}\otimes\mathscr{H}}\left(D^1,D'\right)+d_{\mathscr{H}\otimes\mathscr{H}}\left(D',D^2\right)$$

Lemma 5.2 and 5.3 can also be found in [12] along with the proof. Now we can have the following bound using the above three properties of $d_{\mathscr{H}\otimes\mathscr{H}}$-distance, i.e., symmetry, convexity and triangle inequality.

Theorem 5.1. *For the hypothesis* $h^{(t)}=\Sigma_i\alpha_i^{(t)}h_i^{(t)}$ *with* $\Sigma_i\alpha_i^{(t)}=1$ *and* $\alpha_i^{(t)}\geq0$ obtained at timestamp t of the data stream in our framework, the following bound holds,

$$\epsilon_{D_{t+1}}\left(h^{(t)}\right)\leq\epsilon^*+\sum_i\alpha_i^{(t)}\left(\epsilon_{D_t^i}+d_{\mathscr{H}\otimes\mathscr{H}}\left(D_t,D_t^i\right)+d_{\mathscr{H}\otimes\mathscr{H}}\left(D_t^i,D_{t+1}\right)\right)\quad(5.10)$$

where D_t^i is the normalized marginal distribution defined over feature group F_i for the t-th data chunk, $\epsilon^*=\arg\min_{h'\in\mathscr{H}}\left\{\epsilon_{D_t}\left(h'\right)+\epsilon_{D_{t+1}}\left(h'\right)\right\}$ *and* $\epsilon_{D_t^i}=E_{\mathbf{x}\in D_t^i}\left[|h_i^{(t)}\left(\mathbf{x}\right)-f\left(\mathbf{x}\right)|\right]$ *with* $f(\mathbf{x})$ *being the true label of* $\mathbf{x}$.

Proof. According to the triangle inequality of classification error[3], we have $\epsilon_{D_{t+1}}\left(h^{(t)}\right)\leq\epsilon_{D_{t+1}}\left(h^*\right)+\epsilon_{D_{t+1}}\left(h^{(t)},h^*\right)$ Also,

$$\epsilon_{D_{t+1}}\left(h^{(t)},h^*\right)\leq\epsilon_{D_t}\left(h^{(t)},h^*\right)+|\epsilon_{D_{t+1}}\left(h^{(t)},h^*\right)-\epsilon_{D_t}\left(h^{(t)},h^*\right)|$$

Combining the two inequalities above and using Definition 5.1, we derive

$$\epsilon_{D_{t+1}}\left(h^{(t)},h^*\right)\leq\epsilon_{D_t}\left(h^{(t)},h^*\right)+d_{\mathscr{H}\otimes\mathscr{H}}\left(D_t,D_{t+1}\right)$$

Applying the triangle inequality of classification error on $\epsilon_{D_t}\left(h^{(t)},h^*\right)$, we have

$$\epsilon_{D_{t+1}}\left(h^{(t)}\right)\leq\epsilon^*+\epsilon_{D_t}\left(h^{(t)}\right)+d_{\mathscr{H}\otimes\mathscr{H}}\left(D_t,D_{t+1}\right)\quad(5.11)$$

Then, plugging $D'=\Sigma_i\alpha_i^{(t)}D_t^i$ into Lemma 5.3, and applying symmetry and convexity of $d_{\mathscr{H}\otimes\mathscr{H}}$, we can derive

$$d_{\mathscr{H}\otimes\mathscr{H}}\left(D_t,D_{t+1}\right)\leq\sum_i\alpha_i^{(t)}\left(d_{\mathscr{H}\otimes\mathscr{H}}\left(D_t,D_t^i\right)+d_{\mathscr{H}\otimes\mathscr{H}}\left(D_t^i,D_{t+1}\right)\right)\quad(5.12)$$

Also,

$$\epsilon D_t\left(h^{(t)}\right)\leq\sum_i\alpha_i^{(t)}\epsilon_{D_t^i}\quad(5.13)$$

Substituting Eq. 5.12 and Eq. 5.13 into Eq. 5.11 yields the result.

Theorem 5.1 shows that $\alpha_i^{(t)}$ plays a significant role in the generalization bound. The first term of the upper bound in Eq. 5.10 is the combined error of the ideal hypothesis and thus it can be considered as a constant. As a result, the bound depends on the second term, which is a linear combination of three terms, i.e., the prediction error $\epsilon_{D_t^i}$ of sub-classifier $h_i^{(t)}$ trained on F_i at timestamp t, the distance $d_{\mathscr{H}\otimes\mathscr{H}}\left(D_t,D_t^i\right)$ and the distance $d_{\mathscr{H}\otimes\mathscr{H}}\left(D_t^i,D_{t+1}\right)$. From Theorem 5.1, we can observe that, in order to effectively reduce the upper bound, the sub-classifier with less generalization error and obtained from the distribution more similar to D_{t+1} should be assigned a higher weight. An extreme case would be there is no concept drift, that is, the distributions of D_t and D_{t+1} are identical. In such a case, both distances are zero, and the minimal upper bound can be achieved by putting all weight on sub-classifier h_0 trained from the only full feature space, i.e., $\alpha_0^{(t)}=1$ and $\alpha_i^{(t)}=0$ where $i\neq 0$.

6 Experiments

In this section, we empirically demonstrate the effectiveness of the proposed framework. DFGW-IS is compared with four baseline methods on nine synthetic and real-world benchmark datasets using multiple evaluation metrics.

6.1 Datasets

Table 2 summarizes the characteristics of five synthetic and four real-world datasets used in our experiment. The procedures of each streaming data preparation are presented below.

6.1.1 Synthetic Data *Synthetic Stream*—The concept in this data stream is defined as $g(\mathbf{x}) = \sum_{i=1}^{d} a_i \cdot x_i \cdot x_{d-i+1} - a_0$ where $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and $p(x_i) \sim \mathcal{N}(\mu_i, \delta_i)$. An instance is labeled positive if $g(\mathbf{x}) < 0$. Otherwise, it is labeled negative. Based on the method documented in [7], we created three datasets, i.e. Syn_feature, Syn_cond and Syn dual, each respectively simulating the feature change $p(\mathbf{x})$, conditional change $p(y|\mathbf{x})$ and dual change $p(\mathbf{x}, y)$. In the generation of each dataset, the number of dimensions involved in change is set as two.

Hyper Plane Stream[4]**:** This data stream contains gradually evolving concepts as specified by $f(\mathbf{x}) = \sum_{i=1}^{d-1} a_i$, where a_i controls the shape of the decision surface. In our experiment, we used one vs. rest method to generate two datasets, HyperP1 and HyperP2, where class 3 and class 4 are respectively labeled positive.

6.1.2 Real World Data *Onehr* & *Eighthr*[16]—Both data are ground ozone readings collected over seven years. To simulate the data stream, we split each set into seven chunks by year and then remove the date attribute. Ozone days are labeled positive and the rest are negative. The missing feature values are imputed by the corresponding mean values.

Adult[5]**:** To produce a data stream with su cient concept drifts, we follow the same procedure in [17] and create 14 chunks based on the unique values of the occupation attribute. All categorical attributes are removed and only six continuous features are kept. To make the data more skewed, we further undersample the positive examples in each chunk.

Weather[6]**:** Processed by Polikar et al. [4], this NOAA dataset spans 51 years and initially contains 31% positive instances. In our experiment, we first group yearly data into a chunk, and then we undersample the positive examples in each chunk to create more skewed class distribution.

6.2 Evaluation Metrics

While accuracy is an important evaluation metric for measuring a classifier's performance, it is not an appropriate assessment criterion in learning the highly imbalanced data. Recently, several measures, such as F-measure, G-mean, and AUC, have been proposed to evaluate the classification performance for imbalanced problems [5]. In general, F-measure is defined as the harmonic mean of recall and precision. A high F-measure score signifies a high value for both precision and recall. G-mean is usually used to measure the balanced performance of a learning algorithm between the minority and majority classes. It is the geometric mean of sensitivity and specificity. In addition, AUC is the area under an ROC curve. It provides a single average measure of a classifier's performance as the classification threshold varies. Since each measure is designed to access one particular property [11], we employ all three metrics to rank algorithms.

[4] http://www.cse.fau.edu/~xqzhu/stream.html
[5] http://archive.ics.uci.edu/ml/datasets/Adult
[6] ftp://ftp.ncdc.noaa.gov/pub/data/gsod/

6.3 Experimental Setup

We compared the performance of the proposed algorithm with the following baseline methods. First is the Gao's approach [7] based on Uncorrelated Bagging, and thus we denote it as "UB" hereafter. As suggested by [7], we set the skewness ratio to be 0.4 and the ensemble size to be 5 for its optimal performance. We also implement HUWRS.IP, and all parameters are set using recommended values [8]. To evaluate the effectiveness of the proposed technique for dealing with class imbalance, we implemented another two variants of our DFGW framework. One is DFGW with undersampling technique [10], and the other is DFGW with SMOTE approach [1]. We refer to them as DFGW-Under and DFGW-SMOTE respectively. Specifically, in DFGW-Under, the sub-classifier built from each feature group is an ensemble trained on multiple balanced datasets. Each set contains all current positive examples and the equal number of under-sampled negative examples from the most recent chunk. On the other hand, in DFGW-SMOTE, the sub-classifier built from each feature group is a single model trained on one balanced dataset. In this balanced set, the most up-to-date positive examples plus the number of the synthetic positive instances generated by SMOTE is the same as the number of negative examples in the most recent chunk.

In our experiments, the base learner is J48 decision tree implemented in Weka [15]. For DFGW-IS and its variants, we set 10 for the ensemble size, 30 for the bin size and 0.5 for the weight parameter. The default value of the size of sliding window is the number of negative examples in the current batch. Due to the diverse characteristics of employed data, it is hard to use domain knowledge or other techniques to obtain optimal feature subspaces for each set. Therefore, for each data, 50 feature groups are randomly generated in each run and the reported results are the averages over five independent runs. We also utilized the interleaved Test-Then-Train (or Prequential)[6] scheme so that the over-time dynamic learning curves for all algorithms can be obtained.

6.4 Comparative Results

Table 3 presents the average chunk performance of the compared algorithms on different synthetic and real world data streams. Table 4 summarizes each algorithm's overall rank by averaging the means of three evaluation metrics over all sets. From these empirical results, we can draw the following conclusions.

First, DFGW-IS statistically significantly outperforms both UB and HUWRS.IP on all datasets in terms of AUC, F-measure and G-mean. The average gains achieved by DFGW-IS over UB (HUWRS.IP) on AUC, F-measure and G-mean are 5% (23%), 50.8% (116%), and 11.8% (238%), respectively. We attribute this to our better strategies of handling the class imbalance in the concept drifting environment. We also examine the time series comparisons of these three algorithms as shown in Figure 3. The over-time AUC and F-measure learning curves of DFGW-IS dominate the corresponding curves of UB and HUWRS.IP for most of the timestamps. The G-mean curves of DFGW-IS and UB interweave with each other in the initial stage of the streams. As more and more data chunks arrive, however, DFGW-IS tends to achieve higher G-mean scores since its curves consistently prevail over UB's curves for most of the datasets. Second, DFGW-IS also

consistently achieves higher scores in AUC, F-measure and G-mean than the two variants, i.e. DFGW-Under and DFGW-SMOTE, over most of the datasets. The only exception is that, on Eighthr dataset, the G-mean score of DFGW-IS is slightly lower than that of DFGW-Under. Nevertheless, this difference is not statistically significant. This, from another aspect, demonstrates the efficacy of our technique in dealing with the skewed distribution. **Third**, DFGW-Under has also passed UB and HUWRS.IP, hitting the second place among the five algorithms. This observation further indicates that (1) the inappropriate use of all positive instances could introduce undesired bias in the streaming data mining process; (2) positive instance selection via the rigorous threshold setting can fail to identify and proliferate adequate instances when the distribution is extremely skewed and(or) the underlying concepts drift rather heavily; and (3) the proposed DFGW framework provides a more robust way to address the underlying drifting concepts. **Lastly**, DFGW-SMOTE is the weakest performer compared to DFGW-IS and DFGW-Under. As most of our streams are quite skewed, this suggests that oversampling via synthetic positive instance creation may not be an effective way to battle imbalance when there are only few positive instances available.

6.5 Study on the impact of λ

Now we study the impact of the parameter λ in our DFGW framework with respect to different performance metrics. Parameter λ is a trade-o term in the weight of a sub-classifier that balances the contributions from the sub-classifier's discriminative power and stable level, as defined in line 6 of Algorithm 3.2. For this set of experiments, DFGW-IS was trained on the datasets of syn_feature, syn_cond and syn_dual as λ varies in the range [0,1] with step length being 0.1. The corresponding learning curves for AUC, F-measure and G-mean are presented in Figure 4. We can observe that, for most of the cases, the highest scores of three metrics are achieved when λ is approximately 0.5. This indicates that a balanced tradeo between those two factors would lead to the best generalization performance of the proposed algorithm, which also well corroborates with the theoretical analysis established in the paper.

6.6 Running Time Efficiency

Since DFGW-IS, UB and HUWRS.IP are all ensemble classifiers, we record their parallel running time on a Mac Pro of 6-core Intel Xeon 3.33GHz and 32G memory for fair comparison. In the experiment, we generate 1000 chunks from Synthetic Stream with 1000 data points in each chunk. The skewness ratio is fixed at 0.5%. The ensemble size for each algorithm is set to be 100. As shown in Figure 2, in the parallel execution, the running time of DFGW-IS is primarily determined by the slowest sub-module, that is, S_{0j} in the only full feature space F_0. Therefore, we recorded the sub-module's running time for DFGW-IS. Similarly, the running time of UB and HUWRS.IP was respectively recorded by a single sub-classifier's time. Figure 5 presents the parallel running time of three algorithms over 1000 chunks. It is evident that HUWRS.IP consumes less time than UB and DFGW-IS as its sub-classifier always operates on a sub-feature space. UB needs longer time than DFGW-IS since over-time more data points are involved in its training. In addition, the running time of DFGW-IS remains rather stable after around timestamp 180. This can be ascribed to the

fixed size of training set determined by the temporal sliding window for the positive examples.

7 Conclusion

In this paper, we introduced a new importance sampling driven, dynamic feature group weighting framework for classifying data streams with skewed distributions. Several useful strategies, a temporal sliding window with the awareness of memory usage and importance sampling to remedy skewness, are tightly integrated into the proposed approach to tackle the class imbalance problem. The over-time drifting feature change, can be first detected by a set of pre-defined feature groups, and then, along with the conditional change, is successfully addressed by dynamically weighing the sub-classifiers of the ensemble trained on those feature groups. Our approach provides a unified and adjustable treatment to the different types of drifting concepts present in the imbalanced streaming data. This design is motivated by the theoretical analysis, and its empirical efficacy has been demonstrated on both synthetic and real-world benchmark datasets, where our algorithm significantly outperforms several state-of-the-art methods and techniques on standard performance metrics.

The use of feature groups in DFGW-IS is somewhat analogous to the domain adaptation where more or less labeled examples always available in the target domain[12], which is clearly not the case in our setting. Although we have shown that a weighted ensemble trained on randomly generated feature groups can outperform competing methods in classifying imbalanced data streams, it is worth further exploring the e cient algorithms for generating the best feature groups by incorporating prior domain knowledge in the context of general data stream mining. We leave this as future work.

Acknowledgement

This work is supported by an US Dept. of Army grant (W911NF-12-1-0066), a NIH grant (NIMHD RCMI 8G12MD007595) and the Louisiana Cancer Research Consortium (LCRC).

References

[1]. Chawla N, Bowyer K, Hall L, Kegelmeyer W. Smote: Synthetic minority over-sampling technique. JAIR. 2002; 16:321–357.

[2]. Chen S, He H. Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach. Evolving Systems. 2011; 2(1):35–50.

[3]. Crammer K, Kearns M, Wortman J. Learning from multiple sources. JMLR. 2008; 9:1757–1774.

[4]. Ditzler G, Polikar R. Incremental learning of concept drift from streaming imbalanced data. IEEE Transactions on Knowledge and Data Engineering. 2012; PP(99):1–1.

[5]. Fawcett T. An introduction to roc analysis. Pattern Recognition Letters. 2006; 27(8):861–874.

[6]. Gama, J.; Sebastião, R.; Rodrigues, PP.; ACM. Issues in evaluation of stream learning algorithms; Proceedings of KDD ’09; New York, NY, USA. 2009; p. 329-338.

[7]. Gao, J.; Fan, W.; Han, J.; Yu, PS. A general framework for mining concept-drifting data streams with skewed distributions; Proceedings of SDM’07; 2007; p. 3-14.

[8]. Hoens, T.; Chawla, N. Learning in non-stationary environments with class imbalance; Proceedings of KDD ’12; ACM. 2012; p. 168-176.

[9]. Liu, J.; Chen, R.; Logvinenko, T. Sequential Monte Carlo Methods in Practice. springer; 2001. A theoretical framework for sequential importance sampling and resampling.

[10]. Liu, X-Y.; Wu, J.; Zhou, Z-H. Exploratory under-sampling for class-imbalance learning; Proceedings of ICDM '06; 2006; p. 965-969.

[11]. Liu, Y.; Shriberg, E. Comparing evaluation metrics for sentence boundary detection; Proc. of ICASSP; 2007; p. 451-458.

[12]. Samdani, R.; tau Yih, W. IJCAI. 2011. Domain adaptation with ensemble of feature groups; p. 1458-1464.

[13]. Tumer, K.; Ghosh, J. Classifier combining: Analytical results and implications; Proceedings of the AAAI-96 Workshop; AAAI Press. 1995; p. 126-132.

[14]. Wang H, Fan W, Yu PS, Han J. Mining concept-drifting data streams using ensemble classifiers. KDD'03. 2003:226–235.

[15]. Witten, IH.; Frank, E.; Hall, MA. Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann; 2011.

[16]. Zhang K, Fan W. Forecasting skewed biased stochastic ozone days: analyses, solutions and beyond. Knowl. Inf. Syst. 2008; 14(3):299–326.

[17]. Zhang Y, Jin X. An automatic construction and organization strategy for ensemble learning on data streams. SIGMOD Rec. Sep; 2006 35(3):28–33.

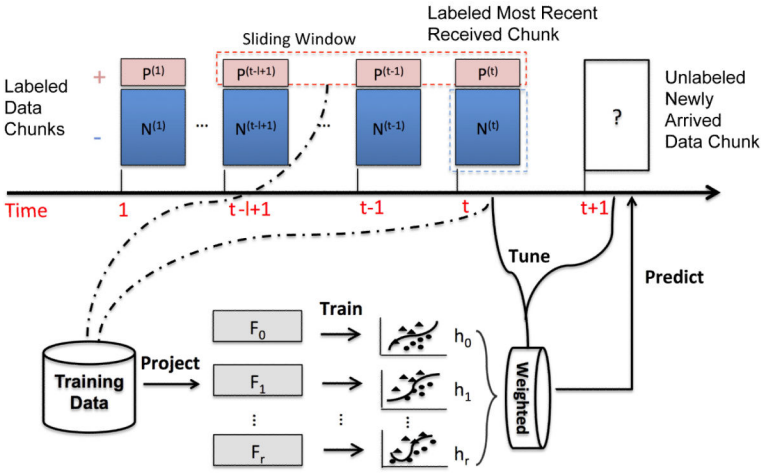

Figure 1.
The learning flow of DFGW-IS

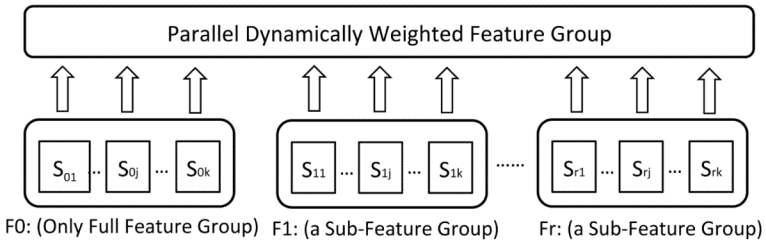

Figure 2.
Structure of the parallel implementation

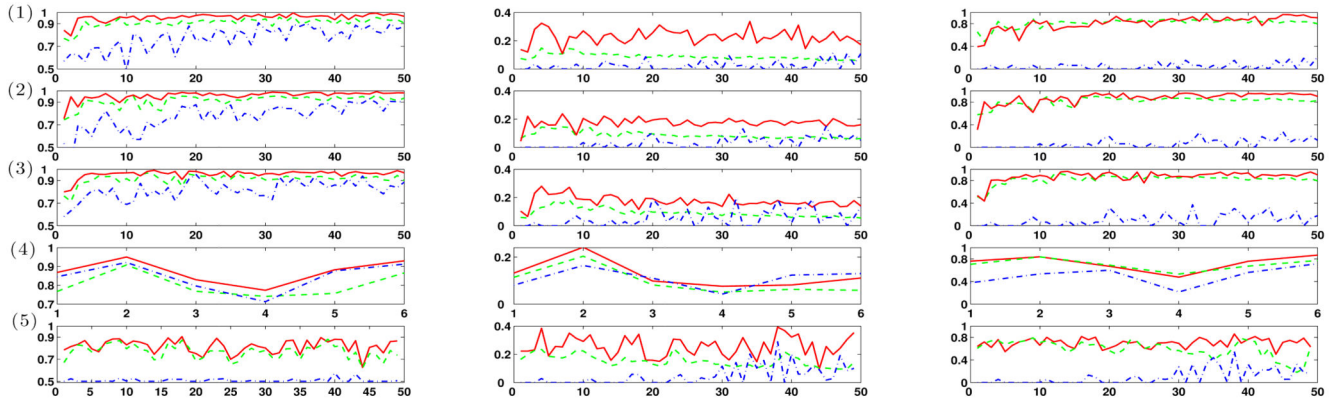

Figure 3.

Time series comparison (Left: AUC; Middle: F-measure; Right: G-mean) of UB, DFGW-IS and HUWRS.IP on Syn_feature(1), Syn_cond(2), Syn_dual(3), Eighthr(4) and Weather(5) datasets. UB: green dashed lines. DFGW-IS: red solid lines. HUWRS.IP: blue dash-dot line. The x-axis represents the timestamp of a data chunk.

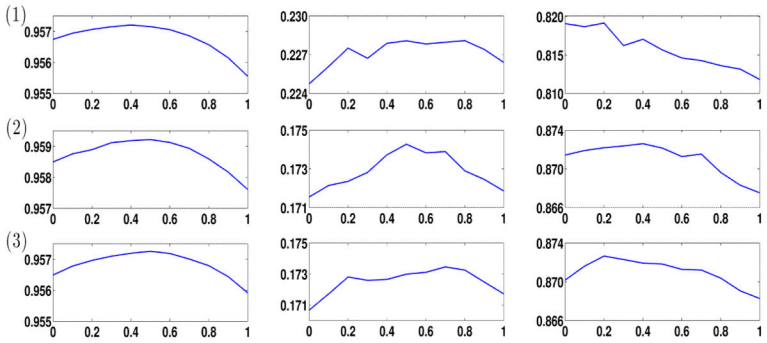

Figure 4.

AUC(left),F-measure(middle),G-mean(right) of DFGW-IS on Syn_feature(1), Syn_cond(2) and Syn_dual(3) datasets with different λ.

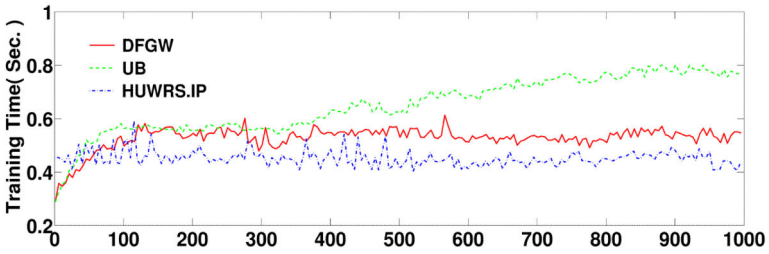

Figure 5.
Parallel running time comparison: DFGW-IS, UB and HUWRS.IP

Table 1

Major notations

Notation	Description
X	Feature space
Y	Class label
$S^{(t)}$	Data chunk at timestamp t
$P^{(t)}$	Positive set at timestamp t
$N^{(t)}$	Negative set at timestamp t
D_t	Data distribution at timestamp t
$x_i^{(t)}$	*i*th data point in data chunk at timestamp t
$y_i^{(t)}$	*i*th class label in data chunk at timestamp t
N_t	Size of the data chunk at timestamp t
$\mathbb{1}(\cdot, \cdot)$	Indicator function
Δ	Convex loss function
F	Full feature set
F_i	A feature subset or group in F
$H_{(t)}^{i}$	Hypothesis space defined over F_i
$h_{(t)}^{i}$	Hypothesis built over F_i on data chunk at timestamp t
$L_{tr}^{(t)}$	Training set at timestamp t
$L_{ho}^{(t)}$	Holdout set at timestamp t
C_i	Misclassification cost of *i*th instance
d_H	Hellinger distance

Table 2

Dataset Description

data sets	two classes	#inst.	#feature	#minority inst.	%Minority	#chunk	chunk size
Syn_feature	< 0 vs. ≥0	51,000	10	510	1.0	51	1,000
Syn_cond	< 0 vs. ≥0	51,000	10	510	1.0	51	1,000
Syn_dual	< 0 vs. ≥ 0	51,000	10	510	1.0	51	1,000
HyperP1	C3 vs. others	100,000	10	10,811	8.85-12.55	50	2,000
HyperP2	C4 vs. others	100,000	10	17,705	15.40-20.80	50	2,000
Onehr	ozone vs. normal	2,536	72	73	0.56-6.03	7	356-366
Eighthr	ozone vs. normal	2,534	72	160	2.81-10.96	7	356-366
Adult	>50k vs. ≤50k	35,760	6	1,149	0.42-9.09	14	11-4,920
Weather	rain vs. no rain	13,094	8	633	4.62-5.00	50	186-295

Table 3

Performance comparison of different algorithms on all datasets(average ± standard deviation(rank)).

Data sets	Methods	AUC	F-measure	G-mean	Average Rank
Syn_feature	UB	0.9091 ± 0.0032(2)•	0.0870 ± 0.0018(3)•	0.8155 ± 0.0057(2)•	2.3
	HUWRS.IP	0.7729 ± 0.1050(4)•	0.0256 ± 0.0106(5)•	0.0488 ± 0.0210(5)•	4.7
	DFGW-Under	0.7769 ± 0.0035(3)•	0.0960 ± 0.0029(2)•	0.5400 ± 0.0046(3)•	2.6
	DFGW-SMOTE	0.6893 ± 0.0086(5)•	0.0749 ± 0.0013(4)•	0.3034 ± 0.0055(4)•	4.3
	DFGW-IS	**0.9577±0.0031**(1)	**0.2281±0.0056**(1)	**0.8806±0.0071**(1)	1.0
Syn_cond	UB	0.9124 ± 0.0039(2)•	0.0870 ± 0.0018(4)•	0.8167 ± 0.0080(2)•	2.7
	HUWRS.IP	0.7781 ± 0.0164(4)•	0.0331 ± 0.0227(5)•	0.0679 ± 0.0488(5)•	4.7
	DFGW-Under	0.8194 ± 0.0016(3)•	0.1185 ± 0.0034(2)•	0.5501 ± 0.0065(3)•	2.7
	DFGW-SMOTE	0.7098 ± 0.0080(5)•	0.0919 ± 0.0048(3)•	0.3632 ± 0.0072(4)•	4.0
	DFGW-IS	**0.9643±0.0033**(1)	**0.1743±0.0025**(1)	**0.8846±0.0097**(1)	1.0
Syn_dual	UB	0.9139 ± 0.0033(2)•	0.0920 ± 0.0021(2)•	0.8219 ± 0.0074(2)•	2.0
	HUWRS.IP	0.8170 ± 0.0077(3)•	0.0536 ± 0.0287(5)•	0.1028 ± 0.0524(5)•	4.3
	DFGW-Under	0.8085 ± 0.0023(4)•	0.0643 ± 0.0011(4)•	0.7055 ± 0.0063(3)•	3.7
	DFGW-SMOTE	0.7179 ± 0.0098(5)•	0.0781 ± 0.0044(3)•	0.2958 ± 0.0134(4)•	4.0
	DFGW-IS	**0.9599±0.0021**(1)	**0.1730±0.0044**(1)	**0.8817±0.0050**(1)	1.0
HyperP1	UB	0.7481 ± 0.0014(3)•	0.2743 ± 0.0006(4)•	0.5820 ± 0.0013(3)•	3.3
	HUWRS.IP	0.6319 ± 0.0291(5)•	0.0004 ± 0.0007(5)•	0.0031 ± 0.0050(5)•	5.0
	DFGW-Under	0.7868 ± 0.0013(2)•	0.3506 ± 0.0007(2)•	0.7227 ± 0.0021(2)•	2.0
	DFGW-SMOTE	0.7173 ± 0.0041(4)•	0.2971 ± 0.0042(3)•	0.5413 ± 0.0051(4)•	3.7
	DFGW-IS	**0.8107±0.0012**(1)	**0.3862±0.0014**(1)	**0.7351±0.0010**(1)	1.0
HyperP2	UB	0.9269 ± 0.0007(3)•	0.5462 ± 0.0011(4)•	0.7864 ± 0.0011(4)•	3.7
	HUWRS.IP	0.7803 ± 0.0373(5)•	0.3900 ± 0.0868(5)•	0.5105 ± 0.0803(5)•	5.0
	DFGW-Under	0.9393 ± 0.0010(2)•	0.7194 ± 0.0014(3)•	0.8663 ± 0.0011(2)•	2.3
	DFGW-SMOTE	0.9200 ± 0.0012(4)•	0.7219 ± 0.0031(2)•	0.8336 ± 0.0046(3)•	3.0
	DFGW-IS	**0.9536±0.0016**(1)	**0.7646±0.0008**(1)	**0.8864±0.0007**(1)	1.0
Onehr	UB	0.8196 ± 0.0187(2)•	0.1466 ± 0.0093(3)•	0.6087 ± 0.0285(2)•	2.3
	HUWRS.IP	0.8001 ± 0.0151(3)•	0.1392 ± 0.0315(4)•	0.3896 ± 0.0552(5)•	4.0
	DFGW-Under	0.7622 ± 0.0100(4)•	0.1364 ± 0.0100(5)•	0.5375 ± 0.0043(3)•	4.0
	DFGW-SMOTE	0.6962 ± 0.0296(5)•	0.2014 ± 0.0118(2)	0.4112 ± 0.0591(4)•	3.7

Data sets	Methods	AUC	F-measure	G-mean	Average Rank
	DFGW-IS	**0.8558±0.0182**(1)	**0.2024±0.0069**(1)	**0.7004±0.0130**(1)	1.0
Eighthr	UB	0.8011 ± 0.0110(4)•	0.2382 ± 0.0080(5)•	0.7017 ± 0.0104(3)•	4.0
	HUWRS.IP	0.8442 ± 0.0033(2)•	0.2711 ± 0.0358(3)•	0.5006 ± 0.0535(5)•	3.3
	DFGW-Under	0.8365 ± 0.0100(3)•	0.2736 ± 0.0052(2)•	**0.7440** ± **0.0117**(1)	2.0
	DFGW-SMOTE	0.7399 ± 0.0349(5)•	0.2711 ±0.0354(3)•	0.5172 ± 0.0590(4)•	4.0
	DFGW-IS	**0.8724** ± **0.0095**(1)	**0.3084** ± **0.0038**(1)	0.7330 ± 0.0076(2)	1.3
Adult	UB	0.7902 ± 0.0090(3)•	0.1412 ± 0.0043(4)•	0.6431 ± 0.0082(2)•	3.0
	HUWRS.IP	0.5869 ± 0.0106(5)•	0.2706 ± 0.0400(1)	0.3845 ± 0.0662(4)•	3.3
	DFGW-Under	0.7968 ± 0.0012(2)•	0.1418 ± 0.0009(3)•	0.6376 ± 0.0036(3)•	2.7
	DFGW-SMOTE	0.7470 ± 0.0019(4)•	0.1363 ± 0.0097(5)•	0.2487 ± 0.0223(5)•	4.7
	DFGW-IS	**0.8334±0.0009**(1)	0.1704±0.0021(2)	**0.7185±0.0019**(1)	1.3
Weather	UB	0.7693 ± 0.0036(3)•	0.1524 ± 0.0028(4)•	0.5982 ± 0.0068(3)•	3.3
	HUWRS.IP	0.5086 ± 0.0073(5)•	0.0470 ± 0.0221(5)•	0.1003 ± 0.0478(5)•	5.0
	DFGW-Under	0.7731 ± 0.0016(2)•	0.1699 ± 0.0015(3)•	0.6836 ± 0.0028(2)•	2.3
	DFGW-SMOTE	0.7571 ± 0.0034(4)•	0.2360 ± 0.0061(2)•	0.5656 ± 0.0082(4)•	3.3
	DFGW-IS	**0.8176±0.0019**(1)	**0.2549±0.0033**(1)	**0.7091±0.0051**(1)	1.0

• indicates that DFGW-IS significantly outperforms the corresponding method wrt a paired t-test with a 95% confidence interval.

Table 4

Rank of Algorithms

	Syn_feature	Syn_cond	Syn_dual	HyperP1	HyperP2	Onehr	Eighthr	Adult	Weather	Average	Final Rank
UB	2.3	2.7	2.0	3.3	3.7	2.3	4.0	3.0	3.3	3.0	3
HUWRS.IP	4.7	4.7	4.3	5.0	5.0	4.0	3.3	3.3	5.0	4.4	5
DFGW-Under	2.6	2.7	3.7	2.0	2.3	4.0	2.0	2.7	2.3	2.7	2
DFGW-SMOTE	4.3	4.0	4.0	3.7	3.0	3.7	4.0	4.7	3.3	3.9	4
DFGW-IS	1.0	1.0	1.0	1.0	1.0	1.0	1.3	1.3	1.0	1.1	1