An Elastic Gradient Boosting Decision Tree for Concept Drift Learning

Kun Wang[1,2][0000−0003−2711−6233], Anjin Liu[1][0000−0002−0733−7138], Jie Lu[1][0000−0003−0690−4732], Guangquan Zhang[1][0000−0003−3960−0583], and Li Xiong[2][0000−0002−6527−0517]

[1] Australian Artificial Intelligence Institute, University of Technology Sydney, Australia
Kun.Wang-2@student.uts.edu.au
{Anjin.Liu, Jie.Lu, Guangquan.Zhang}@uts.edu.au
[2] School of Management, Shanghai University, China
Xiongli8@shu.edu.cn

Abstract. In a non-stationary data stream, concept drift occurs when different chunks of incoming data have different distributions. Hence, over time, the global optimization point of a learning model might permanently drift to the point where the model no longer adequately performs the task it was designed for. This phenomenon needs to be addressed to maintain the integrity and effectiveness of a model over the long term. In this paper, we propose a simple but effective drift learning algorithm called elastic Gradient Boosting Decision Tree (eGBDT). Since the prediction of a GBDT model is the sum output of a list of trees, we can easily append new trees to perform incremental learning or delete the last few trees to roll back to a previously known optimization point. The proposed eGBDT incrementally fits new data and detect drift by searching for the tree with the lowest residual. If the rollback deletions required would exceed the initial number of trees, a retraining process is triggered. Comparisons of eGBDT with five state-of-the-art methods on eight data sets show the efficacy of eGBDT.

Keywords: Concept drift · Data stream · Incremental learning · Gradient boosting.

1 Introduction

Concept drift describes changes in the data distribution of data streams that means the current model is no longer sufficiently accurate in performing the task it was designed for. It is important for a machine learning model to be able to mine the characteristics of the data stream and adapt to changes in the data distribution [20,19,23,22]. There are two main types of concept drift: real drift and virtual drift [14]. Real drift occurs when there are changes in the class boundaries that make the current model obsolete. This type of drift is illustrated in Fig. 1, the global optimization point of a learning model might permanently drift to the point where the model no longer adequately performs the task it was designed for. By contrast, virtual drift means changes in the marginal distribution, while the class boundaries are not affected. Concept drift will make the machine learning model unable to adapt to changeable data, leading to poor prediction

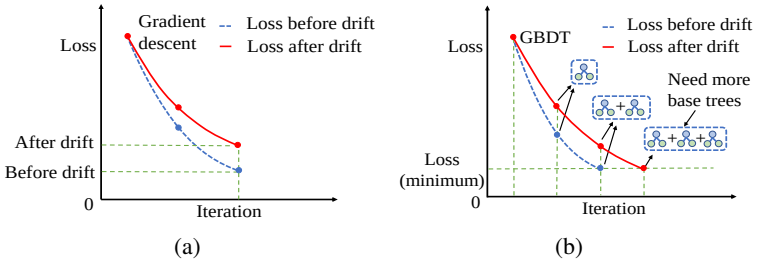


Fig. 1. (a) The loss of the model before and after concept drift occurs. When a real drift occurs, the loss may not decrease quickly. (b) A demonstration of the GBDT model before and after a real drift, more base trees are needed for model learning to help loss get to the minimum point (global optimization point).

and decision outcomes [20]. Our focus in this paper is real drift detection and adaptation for supervised learning.

Algorithms for handling concept drift can be divided into active and passive approaches [8]. Also called drift trigger techniques, active approaches actively detect concept drift in every time step and react after confirming a drift [7]. However, there is no solid strategy for ascertaining the point at which the current model should be retrained for best performance versus fine-tuning the model to account for the drift. For passive methods, the target is to learn data streams based on self-adjustment rather than rely on drift detection results. However, passive methods can perform well on non-drifting or slowly-drifting streams, but they may not suitable for adapting to sudden drifts or streams where concept drifts are frequent [18]. Moreover, combine with the latest machine learning methods, incremental learning methods [25] and ensemble methods [17], are popular in data stream mining. Bagging and boosting are two representative techniques of ensemble learning methods [24], and they also have been extended for online version. However, unlike bagging, which generated base learners in parallel, boosting sequentially constructs a series of base learners. Gradient boosting machine (GBM) [12] is one of the popular boosting models, but it is still an open problem for it to dealing with streaming data since the base learners could not adapt to the changing environment [10]. So, how to measure the impact of concept drift on GBM? Furthermore, how to choose the best model adjustment strategy to maintain model performance, incremental learning, or retraining?

Our aim is to develop an active drift handling algorithm that can choose the optimal moment to switch from fine-tuning to retraining via proposing an elastic Gradient Boosted Decision Tree (eGBDT) algorithm. GBDT iteratively constructs a group of weak learners then linearly combines them into a strong learner. Hence, it is easy to add or delete trees from the tree list, which keeps the model flexible. For example, consider a continuous series of data chunks. A GBDT model can be initially constructed 100 trees on data chunk 0 and then tested on data chunk 1, where the cumulative result of base trees is the prediction result of the entire model. The model can then be incrementally fine-tuned by incrementally learning a further 50 trees. If the best prediction result occurs at the 150th tree, the model could still be underfitting. However, if the best

prediction result occurs at the 110th tree, the model might be overfitting, and trees after the 110th can be deleted. Another case is that the best prediction result occurs at the 80th tree, while the initial training only has 100 trees. This means the GBDT model can not adapt to the data. We consider this as a significant drift, and the model is then retrained. As a bonus, this pruning strategy can help to reduce memory consumption and the complexity of calculations without sacrificing performance.

The main contributions of this paper are as follows:

- Based on the characteristics of GBDT framework, we propose a naïve incremental GBDT (iGBDT) algorithm for incremental learning with dynamic data streams.
- We propose a rollback process to find out the poor performance trees in GBDT model. This is also a signal to find out concept drift.
- We propose an elastic Gradient Boosting Descent Tree (eGBDT) algorithm, which can delete redundant trees without increasing the runtime complexity. This eGBDT handles the uncertain of concept drift well.

This paper is organized as follows: related work and preliminaries are discussed in Section 2, our proposed methods are presented in Section 3, Section 4 illustrated our experiment, conclusion and future work are in Section 5.

2 Related Works and Preliminaries

2.1 Concept Drift Detection and Adaptation

For a time step t, a data chunk with chunk instances $D_t = \{(x_i^t, y_i^t)\}_{i=1}^{\mathrm{chunk}}$ is generated by a distribution $P_t(x, y)$, where x is the attribute vector and y is the label. Concept drift is defined as $P_t(x, y) \neq P_{t-1}(x, y)$, the data distribution changed from time $t-1$ to time t [27]. The goal of drift learning is to ensure that the loss $\ell(f(x), y)$ of a learning model is continuously optimized when a concept drift occurs, i.e.,

$$F_t = \arg\min_{f \in \mathcal{H}} \mathrm{E}_{(x,y)\in P_t(x,y)}[\ell(f(x), y)], \tag{1}$$

where $\mathcal{H}$ is the hypothesis set, $\mathrm{E}(\cdot)$ is the expectation of a random variable [27]. Consider a sequence of $P_t(x, y)$, the target of the incremental learning is given as

$$\min_{F_1, F_2, \ldots, F_t, \ldots} \sum_t \mathrm{E}_{(x,y)\in P_t(x,y)}[\ell(F_t(x), y)] \tag{2}$$

i.e., a set of models created at different time step [27]. Since the real concept drift has no established rules to follow and may include multiple types of drift, it is not only necessary to perform preliminary detection, but also to fine-tune model in real time based on the feedback, reducing the memory redundancy caused by passive learning to improve the flexibility of the model.

The purpose of handling concept drift is to ensure the trained model can perform well on the testing set. Like accurately map the knowledge of the source domain to the target domain [21]. Ensemble approaches to deal with concept drift can be divided

into two categories [17,8]: active drift detection with an adaptation ensemble, and passive ensemble with forgetting mechanism. The active method relies on a drift detection mechanism, informing the model to update in time. Some popular drift detection methods are Drift Detection Method (DDM) [13], Heoffding's inequality based Drift Detection Method (HDDM) [11], ADaptive WINdowing (ADWIN) [1]. The passive methods do not have a drift detection. Instead, it learns the new data and fine-tunes the model in time. Typical examples are Streaming Ensemble Algorithm (SEA) [26], the Learn++ algorithm in Nonstationary Environments (Learn++.NSE) [7], Dynamic Weighted Majority algorithm (DWM) [16], Accuracy Updated Ensemble (AUE1) [5] and Accuracy Updated Ensemble (AUE2) [6]. These algorithms learn drift incrementally with newly arriving data, eliminating old ensembles through a forgetting mechanism [27].

2.2 Gradient Boosting Decision Tree

At present, Boosting [4], which is a critical algorithm in ensemble learning, has been widely used in data mining. GBDT [12] is a kind of classification algorithm based on ensemble learning that turns a weak classifier into a strong classifier through training. We input training set $\{(x_i, y_i)\}_{i=1}^{\text{chunk}}$, a differentiable loss function $\ell(y, f(x))$, number of iterations M. For $m = 1$ to M, we calculate the pseudo-residuals

$$r_{im} = -\left[\frac{\partial \ell(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=f_{m-1}(x)}, i = 1, \ldots, \text{chunk} \tag{3}$$

and fit a base learner $h_m(x)$ to pseudo-residuals, then train it use the training set $\{(x_i, r_{im})\}_{i=1}^{n}$ and calculate the multiplier γ_m as

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)), \tag{4}$$

then update the model as

$$F_M(x) = F_{m-1}(x) + \gamma_m h_m(x). \tag{5}$$

In the literature, GBDT has been used for handling stochastic data streams, such as the Streaming Gradient Boosting algorithm (SGM) [15]. To adapt to the changes of data stream, we propose an incremental method to improve model performances. At the same time, considering the redundancy and complexity of the calculation process, we add a pruning process to increase the flexibility of the model and enhance performances.

3 Elastic Gradient Boosting Decision Tree Ensemble

3.1 Incremental Learning of GBDT——iGBDT

GBDT model is trained using a portion of the data instances as the training set, with the remaining data instances divided into several data chunks for subsequent use in iteratively fine-tuning the model. Once initially trained, the model is tested on new chunks

Algorithm 1 Incremental Learning of GBDT (iGBDT)

Input:
1) initial train chunk $D_{\mathrm{chunk_{ini}}}$
2) new slide chunk $D_{\mathrm{chunk_{slide}}}$
Parameter:
1) Config of GBDT, $\mathrm{GBDT_{parm}} =$
(M = 250, $\mathrm{Depth} = 10, \mathrm{MinSamplesLeaf} = 5, \mathrm{SampleRate} = 0.8, \lambda = 0.01$)
2) Num of incremental trees, L
Output:
1) incremental GBDT $F_{M+L}(x)$
1: build GBDT $F_M(x)$ on D_{train}
2: **for** $l = 1$ to L **do**
3: compute the pseudo-residuals r_{li} on D_{test}
4: fit a new regression tree $h_l(X)$ on D_{test} to pseudo-residuals, i.e. train it using $(x_i, \lambda r_{li})$ where $x_i \in X_{\mathrm{test}}$
5: update the mode according to Eq. (6)
6: **end for**
7: **return** $F_{M+L}(x)$

of incoming data to gauge and fine-tune the model's accuracy. Although GBDT models are high-performing ensemble decision tree models when handling classification tasks, no one has applied it for concept drift learning. With data streams that contain concept drift, it is vital to ensure that the model can adapt to the new patterns. Incremental learning is an ideal method for updating the model.

We first train a GBDT model based on the given data sample, then we test the model on the new incoming data chunk and calculate the residual. We use this data chunk and residual as a new train set to fit a new decision regression tree and add this tree to our original GBDT model to update a new model. For binary classification, we initial training chunk and present new sliding chunk as $D_{\mathrm{chunk_{ini}}} = \{(x_i, y_i)\}_{i=1}^{\mathrm{chunk_{ini}}}$, $D_{\mathrm{chunk_{slide}}} = \{(x_i, y_i)\}_{i=1}^{\mathrm{chunk_{slide}}}$, the term "chunk" represents the chunk size, and $x \in \mathbb{R}^d$ that d is the dimensionality, $y \in \{0, 1\}$. We calculate the pseudo-residuals on testing set as based on Eq. (3). Then, we set the learning rate as λ and set the number of incremental trees as L, we fit L new regression tree on $\{(x_i, \lambda r_{i(M+L)})\}_{i=1}^{\mathrm{chunk_{slide}}}$, the incremental learning model is

$$F_{M+L}(x) = F_{M+L-1}(x) + h_{M+L}(x). \tag{6}$$

This is more computational friendly than the conventional GBDT learning given in Eq. (4, 5), because of the fixed learning rate. Repeating this procedure incrementally tunes the GBDT model. The same process is used with newly arriving chunks so the model incrementally adapts to concept drift if present. The pseudocode for GBDT's incremental learning process is presented in Algorithm 1.

3.2 A Residual-based GBDT Pruning Method——eGBDT

The prediction result of a GBDT model is the sum of the prediction of each tree inside the model. Let's denote the prediction value of the mth tree on a given feature

Algorithm 2 Elastic GBDT (eGBDT)

Input:
1) trained GBDT, $F_M(x)$
2) new slide chunk $D_{\text{chunk}_{\text{slide}}}$
Parameter: N/A
Output:
1) pruned GBDT, $F_{M'}(x)$
1: test GBDT model $F_M(x)$ on $D_{\text{chunk}_{\text{slide}}}$
2: calculate the residual based on the output of each tree R_m
3: find the best tree index based on the mean absolute (MA) value of the residual $m = \text{argmin MA}(R_m)$
4: **if** $m < M$ **then**
5: retrain GBDT $F_M(x)$ on $D_{\text{chunk}_{\text{slide}}}$
6: **return** $F_M(x)$ as $F_{M'}(x)$
7: **else**
8: remove redundant trees $F_{M'}(x) = F_{0,m}(x)$
9: **return** $F_{0,m}(x)$ as $F_{M'}(x)$
10: **end if**

vector x as $\hat{y}_m = h_m(x)$. Then the predictions of all GBDT trees on x is $\{\hat{y}_m\}_{m=1}^{M}$. According to the Eq. (6), the final prediction value on the mth tree is the sum of the prediction values from the first tree to the mth tree, i.e.,

$$F_m(x) = \sum_{i=1}^{m} h_i(x) + \bar{y}, \tag{7}$$

where the $\bar{y}$ is the mean of the label, namely the initial prediction of the GBDT. The first regression tree h_0 is trained on $D_{\text{chunk}_{\text{ini}}}$, with the mean of the label $\bar{y}$ as the initial target variable, and the residual is calculated by $R_0 = y - \bar{y}$. The residuals of the previous tree are used to train next tree from h_1 to h_m, where $m \in \mathbb{Z}^+_{m \leq M}$, M is the max number of iterations. We have the residual vector of the mth tree on a sliding data chunk as:

$$R_m = y - F_m(x) = y - \bar{y} - \sum_{i=1}^{m} h_i(x), \tag{8}$$

that $R_m = \{r_{im}\}_{i=1}^{\text{chunk}_{\text{slide}}}$. With the residual of each tree, we can evaluate the performance of the GBDT with any length, which is the reason we call it elastic GBDT. For runtime efficiency, the residuals are stored in a $\text{chunk}_{\text{slide}}$ by M size matrix. Then we can find the best performance sub list of trees by finding the minimum mean absolute (MA) residual.

$$I_{\text{elastic}} = \underset{m \in \mathbb{Z}^+_{m \leq M}}{\text{argmin}} \ \text{MA}(R_m) \tag{9}$$

where the I_{elastic} is the elastic GBDT tree index, i.e., the sub tree list of GBDT that $\text{eGBDT} = \{h_0(x), \ldots, h_{I_{\text{elastic}}}(x)\}$ will be used for later prediction. And the redundant sub tree list $\text{rGBDT} = \{h_{I_{\text{elastic}}+1}(x), \ldots, h_M(x)\}$ will discard. If I_{elastic} is smaller than a predefined threshold, then we say there is a significant concept drift, and a new GBDT will be retrained on the new data.

For eGBDT, the GBDT pruning process is always associated with the incremental learning process. At the beginning of the stream, we build a GBDT with M trees on $D_{\text{chunk}_{\text{ini}}}$. Then for each new data chunk, we make the prediction, perform pruning and drift detection, and incrementally fit the pruned GBDT with L new trees on $D_{\text{chunk}_{\text{slide}}}$. In this setting, we use the initial number of trees M as the drift threshold, i.e., if $I_{\text{elastic}} < M$, a retraining process will be triggered.

4 Experiments

4.1 Experiment Settings

Three GDBT-based methods and the five state-of-the-art concept drift handling methods were compared as follows. The parameters of GBDT were same, $M = 250$, Depth $= 10$, Min Samples Leaf $= 5$, Sample Rate$= 0.8$, $\lambda = 0.01$. For the iGBDT, we set the number of incremental trees as $L = 25$. The five state-of-the-art algorithms were implemented based on MOA prequential evaluation [2], and parameters were set as the default values as suggested by the authors.

Baseline, train a GBDT model on the training chunk and test it on the reset of the streams.

iGBDT, incremental GBDT that train on initial chunk and incremental fit on new data. For example, train on chunk 0 and test on chunk 1, then incremental fit on chunk 1 and test on chunk 2. In our experiment, we set the same number of base learners for each incremental learning.

Learn++.NSE [9], is an ensemble of classifiers for incremental learning to handle the changeable data distribution. It trains one new classifier for each batch of data. It can receive and combine these classifiers by using dynamically weighted majority voting [9].

OnlineAUE [6], maintains a weighted ensemble of base learners and uses a weighted voting rule for its final prediction. It combines accuracy-based weighting mechanisms known from block-based ensembles with the incremental nature of Hoeffding Trees [6].

LeverageBag [3], focuses on randomizing the input and output of the classifier, while increasing accuracy and diversity, construct an ensemble of classifiers. This method combines the simplicity of bagging by adding more randomization to the input and output of the classifiers.

HDDM-W-Test, HDDM-A-Test [11], is an error-based online drift detection method based on McDiarmid's bounds. HDDM-A-Test is based on Hoeffding's bounds. HDDM-W-Test is based on McDiarmid's bounds and uses the EWMA statistic as an estimator.

The proposed GBDT-based methods were implemented by Python. The base decision tree learners of GBDT framework are learned by using the Sklearn Decision Tree Regression package. The parameter setting on each data set was the same. The $\text{chunk}_{\text{ini}} = 365$ and $\text{chunk}_{\text{slide}} = 365$, the chunk size is established according to the periodicity of the data, and this will be explained in the data sets description. To be fair, we uniformly use the decision tree model as the base learner in other algorithms. Using the Hoeffding trees as the base learners in Learn++.NSE, LeverageBag, OnlineAUE, HDDM-W-Test, HDDM-A-Test algorithms. During the experiment, all algorithm parameters are set uniformly.

Table 1. Real-world data sets statistics (Due to missing data in some years and months, the number of data samples in the same duration are different.)

Number	Data sets	Location	Time Duration	Samples	Features	Class	Ratio
042700	Narsarsuaq	Greenland/Europe	1942-2020	27,592	8	2	0.84:1
100370	Schleswig	Germany/Europe	1942-2020	27,238	8	2	0.44:1
265090	Klajpeda	Lithuania/Europe	1942-2020	25,591	8	2	0.77:1
424750	Allahabad	India/Asia	1942-2018	18,087	8	2	0.32:1
567780	Kunming	China/Asia	1942-2020	25,631	8	2	0.75:1
606560	Tindouf	Algeria/Africa	1943-2020	16,660	8	2	0.04:1
702220	Galena	Illinois/North America	1942-2020	27,197	8	2	0.85:1
802220	Bogota	Colombia/South America	1942-2020	24,604	8	2	0.56:1

4.2 The Data Sets

Each approach was tested on eight real-world data sets. Information about these data sets was summarized in Table 1. A brief description of each follows. The real-world data sets were the WMO world weather data for different region area. Since the data set records the daily weather changes of each region, we define a year as a cycle and set the experimental chunk size as 365. We selected eight data sets from five regions, including Europe, Asia, Africa, North America, South America. The missing values in the data have been filled with the mean value. And we retained eight attributes: temperature (we convert Fahrenheit to Celsius), dew point, sea level pressure, visibility, average wind speed, maximum sustained wind speed, maximum temperature, and minimum temperature. Third, in order to simplify the calculation, the labels of the data sets have been converted into two classes. There are six kinds of weather changes in the original, they are fog, rain, snow, hail, thunder, tornado. We define the first class as the weather without these changes, and the second class as the weather including these changes. In addition, due to the imbalance of the sample categories, we will also calculate the accuracy, F1-score, MCC score, and Friedman test in the experiment and make a comprehensive evaluation of the initial experimental results. The data sets and the source code of this research are available online[3].

4.3 Finding and Discussion

Comparing eGBDT with GBDT-based concept drift handling methods The detailed experiment results of **Baseline**, **iGBDT**, **eGBDT** are shown in Table 2, 3, 4. We calculate the average macro F1-score and accuracy score of Baseline, iGBDT and eGBDT. The average F1-score of each algorithm is 57.33% (Baseline), 76.67% (iGBDT), 76.76% (eGBDT). The average accuracy score of each algorithm is 67.14% (Baseline), 81.78% (iGBDT), 82.05% (eGBDT). The average runtime of each algorithm is 1s (Baseline), 59s (iGBDT), 56s (eGBDT). Moreover, the MCC score of eGBDT is generally higher than Baseline and iGBDT.

Besides, we also summarized the number of trees pruned from the GBDT model in accordance with the accuracy, as shown in Fig. 2. The number of trees of the model

[3] https://github.com/kunkun111/AJCAI-eGBDT

Table 2. Average macro F1-score of eight real-world data sets (%)

Data sets	Baseline	iGBDT	eGBDT	Learn++	Leverage	HDDM_A	HDDM_W	OnlineAUE
Narsarsuaq	32.69(8)	78.07(3)	79.15(2)	71.60(7)	77.73(4)	74.37(6)	75.15(5)	**79.23(1)**
Schleswig	63.67(8)	81.36(2)	**81.38(1)**	65.51(7)	69.99(4)	68.99(6)	69.24(5)	71.29(3)
Klajpeda	69.93(6)	76.03(2)	**77.01(1)**	68.01(8)	72.79(4)	70.14(5)	69.86(7)	73.47(3)
Allahabad	73.92(5)	82.56(2)	**82.96(1)**	65.09(8)	75.82(4)	71.38(7)	71.86(6)	78.45(3)
Kunming	64.96(8)	79.42(2)	**80.14(1)**	71.55(7)	76.44(4)	73.22(6)	73.69(5)	78.83(3)
Tindouf	50.31(8)	67.16(2)	**67.95(1)**	52.25(7)	55.90(6)	63.92(3)	58.49(5)	59.23(4)
Galena	48.36(8)	**76.49(1)**	74.65(3)	66.63(7)	73.99(4)	70.01(6)	71.55(5)	76.46(2)
Bogota	54.81(8)	**72.30(1)**	70.91(4)	68.22(7)	70.69(5)	70.60(6)	70.96(3)	71.99(2)
AvgRank	7.3	1.8	**1.7**	7.2	4.3	5.6	5.1	2.6

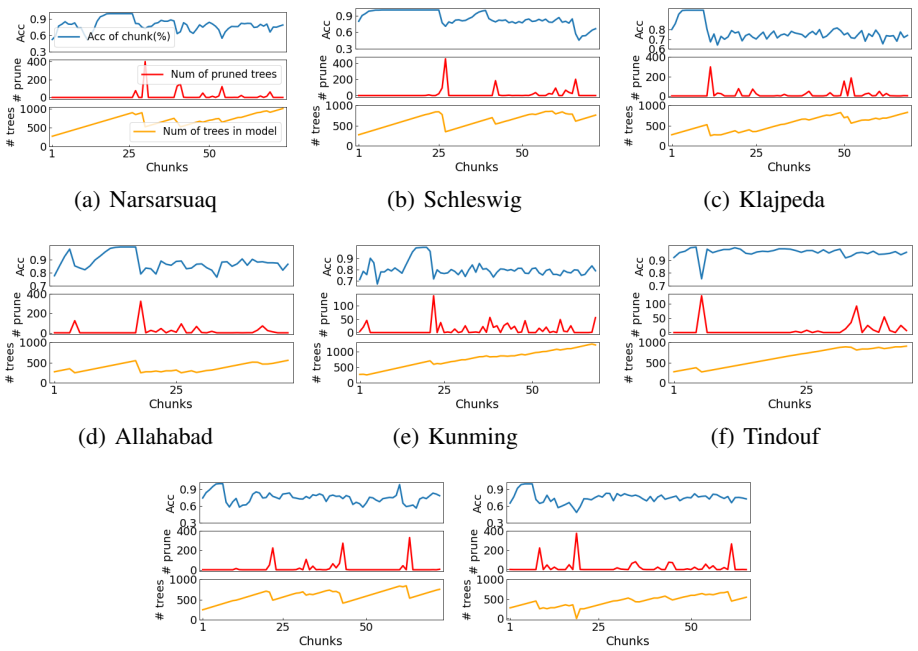
Table 3. Accuracy of eight real-world data sets (%)

Data sets	Baseline	iGBDT	eGBDT	Learn++	Leverage	HDDM_A	HDDM_W	OnlineAUE
Narsarsuaq	46.10(8)	78.08(3)	79.16(2)	71.63(7)	77.81(4)	74.52(6)	75.49(5)	**79.38(1)**
Schleswig	72.74(7)	85.01(2)	**85.11(1)**	72.69(8)	76.72(4)	73.72(5)	73.62(6)	78.14(3)
Klajpeda	70.03(7)	76.36(2)	**77.40(1)**	68.13(8)	73.24(4)	70.22(5)	70.07(6)	73.92(3)
Allahabad	82.15(5)	87.38(2)	**88.00(1)**	76.35(8)	83.60(4)	78.08(7)	78.68(6)	85.12(3)
Kunming	65.21(8)	79.71(2)	**80.43(1)**	71.81(7)	76.87(4)	73.49(6)	74.06(5)	79.27(3)
Tindouf	72.96(8)	95.01(6)	95.86(2)	95.36(5)	95.83(3)	94.75(7)	95.44(4)	**96.13(1)**
Galena	58.89(8)	**76.90(1)**	75.30(3)	66.77(7)	74.22(4)	70.04(6)	71.60(5)	76.67(2)
Bogota	69.05(8)	75.83(2)	75.19(3)	72.22(7)	75.01(4)	73.08(6)	73.92(5)	**75.85(1)**
AvgRank	7.3	2.5	**1.7**	7.1	3.8	6	5.2	2.1

Table 4. Matthews correlation coefficient (MCC) of eight real-world data sets (%)

Data sets	Baseline	iGBDT	eGBDT	Learn++	Leverage	HDDM_A	HDDM_W	OnlineAUE
Narsarsuaq	2.92(8)	56.81(3)	**59.03(1)**	43.21(7)	55.60(4)	48.86(6)	50.55(5)	57.61(2)
Schleswig	29.81(8)	63.42(2)	**63.61(1)**	31.42(7)	41.31(4)	37.89(6)	38.41(5)	43.48(3)
Klajpeda	40.48(6)	52.06(2)	**54.03(1)**	36.41(8)	45.50(4)	41.03(5)	40.00(7)	46.19(3)
Allahabad	48.67(5)	65.22(2)	**66.34(1)**	30.45(8)	52.59(4)	42.99(7)	43.87(6)	57.19(3)
Kunming	36.89(8)	58.93(2)	**60.34(1)**	42.75(7)	52.58(4)	46.54(6)	47.34(5)	56.91(3)
Tindouf	16.69(7)	34.46(2)	**37.68(1)**	8.14(8)	21.59(6)	29.40(3)	23.32(5)	29.09(4)
Galena	20.13(8)	**53.43(1)**	50.35(3)	32.84(7)	47.99(4)	40.42(6)	43.30(5)	51.95(2)
Bogota	26.33(8)	45.42(2)	43.45(4)	37.98(7)	43.75(3)	42.16(6)	42.72(5)	**45.77(1)**
AvgRank	7.2	2	**1.6**	7.3	4.1	5.6	5.3	2.6

increased suddenly, with the accuracy reduced abruptly on the data set, which contains sudden drift is obviously. It suggests that the data changes can be observed clearly according to the fluctuating number of pruned trees and model performances. The number of pruned trees depends on the drift severity. Drift with a relatively lower severity can be handled by the incremental process until it reaches a significant accuracy drop. Furthermore, because of the removal of some poorly-performing base trees, the model's



(a) Narsarsuaq (b) Schleswig (c) Klajpeda

(d) Allahabad (e) Kunming (f) Tindouf

(g) Galena (h) Bogota

Fig. 2. A plot of the chunk accuracy, the number of pruned trees, and the number of trees in GBDT during drift learning. We can clearly see that the accuracy dropped, the number of pruned trees increased, and the number of trees in eGBDT decreased.

Table 5. Friedman test of eGBDT with five state-of-the-art methods based on MCC

Methods	Learn++	Leverage	HDDM_A	HDDM_W	OnlineAUE
P-value	0.00468	0.03389	0.00468	0.00468	0.1573
Significance	$P < 0.05$	$P < 0.05$	$P < 0.05$	$P < 0.05$	$P > 0.05$

overall performance is maintained, and the redundant base trees in the model are also reduced.

Evaluate eGBDT with Five Concept Drift Handling Algorithms The stream classification results of eGBDT ensemble and five state-of-the-art concept drift handling algorithms are summarized in Table 2, 3, 4, 5. The F1-score, accuracy, and MCC score are measured by using the Sklearn package. The experiment shows that eGBDT has competitive performance. For the average macro F1-score, eGBDT performs well on Schleswig, Klajpeda, Allahabad, Kunming, Tindouf, and have the highest average rank. For the accuracy, eGBDT performed well on Schleswig, Klajpeda, Allahabad, Kunming, and also got the highest average rank. For the MCC score, eGBDT also achieved better results with an average ranking 1.6, followed by iGBDT and OnlineAUE. In order to compare the performance of eGBDT statistically, we also perform Friedman tests on

eGBDT and five methods based on MCC score, as shown in Table 5. The results show that at a significance level of 0.05, although the performance of eGBDT is not obvious when compared with OnlineAUE method, the overall performance is better. Overall, we find that eGBDT with both the incremental learning and self-adjusting tree pruning process is beneficial for maintaining model stability and generally produces better results.

Discussion Our eGBDT is a chunk-based drift learning algorithm, it still can outperform the prequential learning algorithms in some cases. However, there are two factors that we need further evaluation. One is model optimization, although our method has the highest average ranking, it is undeniable that it has not achieved better results on some data sets. One reason is that our eGBDT is a linear combination of decision regression trees, when face with remarkably sudden drift, the model adaptability may be lacking. Another one is programming language (Our Baseline, iGBDT, and eGBDT are implemented in Python, while MOA implements five state-of-the-art methods in Java).

5 Conclusion

In this paper, we proposed an elastic GBDT drift learning algorithm. The proposed eGBDT integrates incremental learning and tree pruning to dynamically adjust the number of trees for different stream situations. The incremental learning of GBDT can help the model to improve performance. The process of tree pruning is a simple but efficient way to increase model stability and detect drift. Experiments on real-world data sets have shown the potential of eGBDT. In future research, we will continue to improve our eGBDT method for concept drift adaptation, we will mainly focus on enhancing the adaptability, processing efficiency, and stability of our eGBDT method when dealing with different types of concept drift.

Acknowledgments

This work was supported by the Australian Research Council through the Discovery Project under Grant DP190101733.

References

1. Bifet, A., Gavalda, R.: Learning from time-changing data with adaptive windowing. In: SIAM. pp. 443–448. SIAM (2007)
2. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: MOA: Massive online analysis. Journal of Machine Learning Research **11**, 1601–1604 (2010)
3. Bifet, A., Holmes, G., Pfahringer, B.: Leveraging bagging for evolving data streams. In: ECML PKDD. pp. 135–150. Springer (2010)
4. Breiman, L.: Bias, variance, and arcing classifiers. , Tech. Rep. 460, Statistics Department, University of California, Berkeley (1996)
5. Brzeziński, D., Stefanowski, J.: Accuracy updated ensemble for data streams with concept drift. In: HAIS. pp. 155–163. Springer (2011)

6. Brzeziński, D., Stefanowski, J.: Reacting to different types of concept drift: The accuracy updated ensemble algorithm. IEEE Transactions on Neural Networks and Learning Systems **25**(1), 81–94 (2013)
7. Ditzler, G., Polikar, R.: Incremental learning of concept drift from streaming imbalanced data. IEEE Transactions on Knowledge and Data Engineering **25**(10), 2283–2301 (2012)
8. Ditzler, G., Roveri, M., Alippi, C., Polikar, R.: Learning in nonstationary environments: A survey. IEEE Computational Intelligence Magazine **10**(4), 12–25 (2015)
9. Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks **22**(10), 1517–1531 (2011)
10. Feng, J., Xu, Y.X., Jiang, Y., Zhou, Z.H.: Soft gradient boosting machine. arXiv preprint arXiv:2006.04059 (2020)
11. Frías-Blanco, I., del Campo-Ávila, J., Ramos-Jimenez, G., Morales-Bueno, R., Ortiz-Díaz, A., Caballero-Mota, Y.: Online and non-parametric drift detection methods based on hoeffding's bounds. IEEE Transactions on Knowledge and Data Engineering **27**(3), 810–823 (2014)
12. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics pp. 1189–1232 (2001)
13. Gama, J., Medas, P., Castillo, G., Rodrigues, P.: Learning with drift detection. In: SBIA. pp. 286–295. Springer (2004)
14. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys **46**(4), 44 (2014)
15. Hu, H., Sun, W., Venkatraman, A., Hebert, M., Bagnell, A.: Gradient boosting on stochastic data streams. In: AISTATS. pp. 595–603. PMLR (2017)
16. Kolter, J.Z., Maloof, M.A.: Dynamic weighted majority: An ensemble method for drifting concepts. Journal of Machine Learning Research **8**, 2755–2790 (2007)
17. Krawczyk, B., Minku, L.L., Gama, J., Stefanowski, J., Woźniak, M.: Ensemble learning for data stream analysis: A survey. Information Fusion **37**, 132–156 (2017)
18. Liu, A., Lu, J., Liu, F., Zhang, G.: Accumulating regional density dissimilarity for concept drift detection in data streams. Pattern Recognition **76**, 256–272 (2018)
19. Liu, A., Lu, J., Zhang, G.: Concept drift detection via equal intensity k-means space partitioning. IEEE Transactions on Cybernetics (2020). https://doi.org/10.1109/TCYB.2020.2983962
20. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering **31**(12), 2346–2363 (2018)
21. Lu, J., Zuo, H., Zhang, G.: Fuzzy multiple-source transfer learning. IEEE Transactions on Fuzzy Systems (2019). https://doi.org/10.1109/TFUZZ.2019.2952792
22. Lu, N., Lu, J., Zhang, G., De Mantaras, R.L.: A concept drift-tolerant case-base editing technique. Artificial Intelligence **230**, 108–133 (2016)
23. Lu, N., Zhang, G., Lu, J.: Concept drift detection via competence models. Artificial Intelligence **209**, 11–28 (2014)
24. Oza, N.C.: Online bagging and boosting. In: SMC. pp. 2340–2345. IEEE (2005)
25. Schlimmer, J.C., Granger, R.H.: Incremental learning from noisy data. Machine Learning **1**(3), 317–354 (1986)
26. Street, W.N., Kim, Y.: A streaming ensemble algorithm (SEA) for large-scale classification. In: ACM SIGKDD. pp. 377–382. ACM (2001)
27. Sun, Y., Tang, K., Zhu, Z., Yao, X.: Concept drift adaptation by exploiting historical knowledge. IEEE Transactions on Neural Networks and Learning Systems **29**(10), 4822–4832 (2018)