**Frontiers in Computing and Intelligent Systems**
ISSN: 2832-6024 | Vol. 12, No. 1, 2025

# An Ensemble Multi-Model Voting Method for Adapting to Concept Drift

**Min Wang**

School of Computer Science and Technology, Taiyuan Normal University, Jinzhong Shanxi, 030619, China

**Abstract:** Aiming at the challenges posed by concept drift in streaming data mining, this paper proposes an Ensemble Multi-Model Voting Method for Adapting to Concept Drift (EMVM_ATCD). The method employs integrated multi-classifiers to improve model stability, uses online learning methods to update the model, and adds a dropout layer to force the model to learn different combinations to enhance generalization ability. A voting mechanism is used to process the model prediction results to enhance the ability to cope with concept drift. Experimental results show that the method achieves performance improvements ranging from 0.1% to 10% on multiple datasets, proving that it can effectively handle various types of data.

**Keywords:** Streaming Data Mining; Integration Methods; Voting Mechanisms; Concept Drift.

## 1. Introduction

Concept drift [1-2] is a challenging task in the field of streaming data mining. Concept drift is a phenomenon where the data generation process changes at a point in time. The difference with traditional data mining lies in the fact that data in streaming data is constantly generated and arrives in real time. The underlying model is unable to provide accurate predictions when faced with data distributions that are subject to change, and the emergence of conceptual drift makes many classical learning algorithms become inapplicable when dealing with streaming data, bringing great adjustments to data mining. With the development of research, deep learning [3-6] techniques have gradually emerged to cope with concept drift.

For concept drift, common strategies include incremental learning, adaptive strategies, sliding window methods, and integration methods. These strategies aim to help models cope with changes in data while improving robustness. Incremental learning is a batch learning method that does not rely on learning from the complete dataset, and its updates the model parameters by learning incrementally as new data arrives. Adapting to data changes by means of adjusting the learning rate, regularization and model changes are adaptive strategies. Weighted voting mechanism retains the prediction results of both historical and newly arrived data, and in order to focus the model's attention on the current data, the newly arrived data is given a higher weight to enhance the adaptability. The sliding window approach sets up a fixed-size data window, retains the latest samples to train the model, and as new data arrives, the window slides while removing the oldest data to detect changes in the data distribution.

As the research on conceptual drift is gradually deepening, practical application areas are beginning to use models based on conceptual drift research for data analysis. Long and short-term memory networks (I-LTSM) [7] are used in the field of information security to enhance the adaptive protection of the system against attacks by detecting anomalies in the network. DeepBreath [8] analyzes data from the financial market to mitigate financial risks and has been verified to yield better returns than expert investments. Concept Explorer [9] visualizes concept drift to helps decision makers to visualize and monitor in behavioral analysis and air quality detection, and helps to inform the adoption of subsequent strategies by experts. Thus, the concept drift adaptation approach enables the potential relationships in the data to be recognized, and with the understanding of the data patterns, decision makers can effectively determine the main influencing factors and identify potential dangers in order to make the right decisions. Conceptual drift research applied to practical problems can support decision making in different fields.

In this paper, based on the existing deep learning methods, the network model with fast convergence speed and guaranteed real-time update is selected, in order to improve the prediction performance while improving the ability to adapt to the conceptual drift of the model. In this paper, a classification prediction model integrating multi-model voting is designed based on a single hidden layer neural network, and the overall structure is designed by integrating multiple networks, adding dropout layers and other methods. The results show that the method adapts to the occurrence of conceptual drift while improving the prediction ability.

## 2. Basics

### 2.1. Single Hidden Layer Neural Networks

The design of neural networks originates from the biological nervous system by mimicking the connection between neurons and the process of information transfer. A single hidden layer neural network [10] is a neural network that contains only one hidden layer, which is less computationally intensive and the training process is more efficient compared to deep neural networks. Each layer in the structure consists of multiple neurons and each neuron has its weight, the hidden layer will transform the input information in a non-linear way and the output layer is processed by activation function for prediction. The neural network relies on the back propagation algorithm [11], which adjusts the weight changes by calculating the error between the predicted and true results so that it can approximate the true results. In this paper, the model is constructed by integrating a single hidden layer neural network as a way to cope with complex scenarios containing conceptual drift.

### 2.2. Dropout Regularization Technique

Dropout enhances the model's ability to learn various data

characteristics by randomly discarding some neurons during neural network computation so that the model does not only look at certain neurons or features. When the training data is unbalanced, the neural network performs well on classes with a high number of occurrences and will ignore the presence of a few classes, resulting in a model that is poor at recognizing a few classes. The proposal of the Dropout layer breaks this neuron dependence so that the network has to learn from a wider range of features, and each neuron learns the data features, increasing the independence of the neurons. Dropout is applicable to a variety of neural networks and has achieved significant results in many tasks.

# 3. Integration Model

## 3.1. Algorithmic Model

EMVM_ATCD obtains an integrated model with high performance by integrating a single hidden layer neural network. The prediction results of the integrated model are weighted and the model is updated in reverse to improve the extraction of data patterns, and a Dropout layer is introduced to improve the generalization performance and prevent overfitting to cope with the concept drift challenge. The model structure is shown in Figure 1:

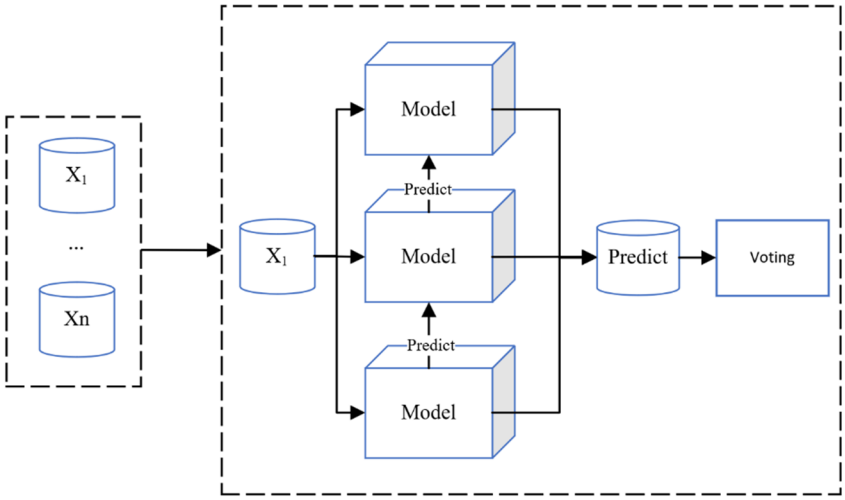


**Fig 1.** Structure of the EMVM_ATCD model

The algorithm enters the model sequentially in the form of data blocks with a serially integrated single hidden layer neural network structure, and after each base model has finished running, it passes the prediction probability to all the base models behind it, and the models behind it will combine the current model prediction probability with the transmitted prediction probability and weight the probabilities to calculate the output result. Subsequently, the output results of multiple base models are voted to get the final result, which improves the robustness of the model. In this, each base model updates the model by calculating the error based on the prediction results to improve the model's adaptability in the event of conceptual drift.

## 3.2. Model Structure

The model structure of the algorithm combines incremental learning and integration methods [12-17], serially integrating multiple single hidden layer neural network models to compensate for the limitations of single hidden layer neural networks in processing complex data. Incremental learning is introduced to update the model parameters every time new information is learned, which improves the generalization performance in the face of concept drift and reduces computational resources through continuous iteration. Each base learner in the integrated approach has a different learning process for the data, which can better capture the diversity and changes in the data and improve the overall stability of the model. As shown in Figure 2:

The integrated model structure is a serial model, first of all the data enters the model sequentially in the form of a fixed data block $X_i$, and the output of the j-th base model is $f_j(x_i, \theta_j), i = 1,2,\cdots,n, j = 1,2,\cdots,m$, in which $\theta_j$ is the parameter set in the corresponding base model, and there is also a corresponding weight $\alpha_j, j = 0,1,\cdots,m$ in each base model, so that The output of each model is:

$$F_j(x) = \sum_{j=1}^{m} \alpha_j f_j(x_i, \theta_j) \tag{1}$$

The model is designed on the basis of incremental learning, and the output in Eq. (1) will be weighted by combining the prediction results of previous base models, which can improve the overall performance. As the data block X_i enters, each base model will make a prediction based on the current parameters, which are the parameters of the updated model after the prediction of the previous data block is finished. The m-1st base learner process is as follows:

$$F_{m-1}(x) = \sum_{j=1}^{m-1} \alpha_{i-1,j} f_j(x_i, \theta_{i-1,j}) \tag{2}$$

The result of the mth base learner is to add the corresponding output to Eq. (2). In the process of weighting the output of multiple models, the input of the mth base learner will combine the original input and the hidden layer output of the m-1st base model, expanding the capacity of the model while increasing the complexity of the model, and ensuring the performance of model adaptation through multiple rounds of computational iterations.

## 3.3. Parameter Updates

Each base learner in the integrated model updates its parameters through a back-propagation mechanism, which first calculates the error and updates the error in the opposite direction to optimize all parameters and weights in the base learner.

The data and labels of the sample are divided into $\{X, Y\}$ by data blocks, and the parameter set obtained through

optimization is $\{\alpha_{ij}, \theta_{ij}\}_{j=1}^{m}\{\alpha_{ij}, \theta_{ij}\}_{j=1}^{m}$, at this time, the classification result of the j-th base learner is:

$$\tilde{y}_j(x) = softmax(F_j, x_i) \tag{3}$$

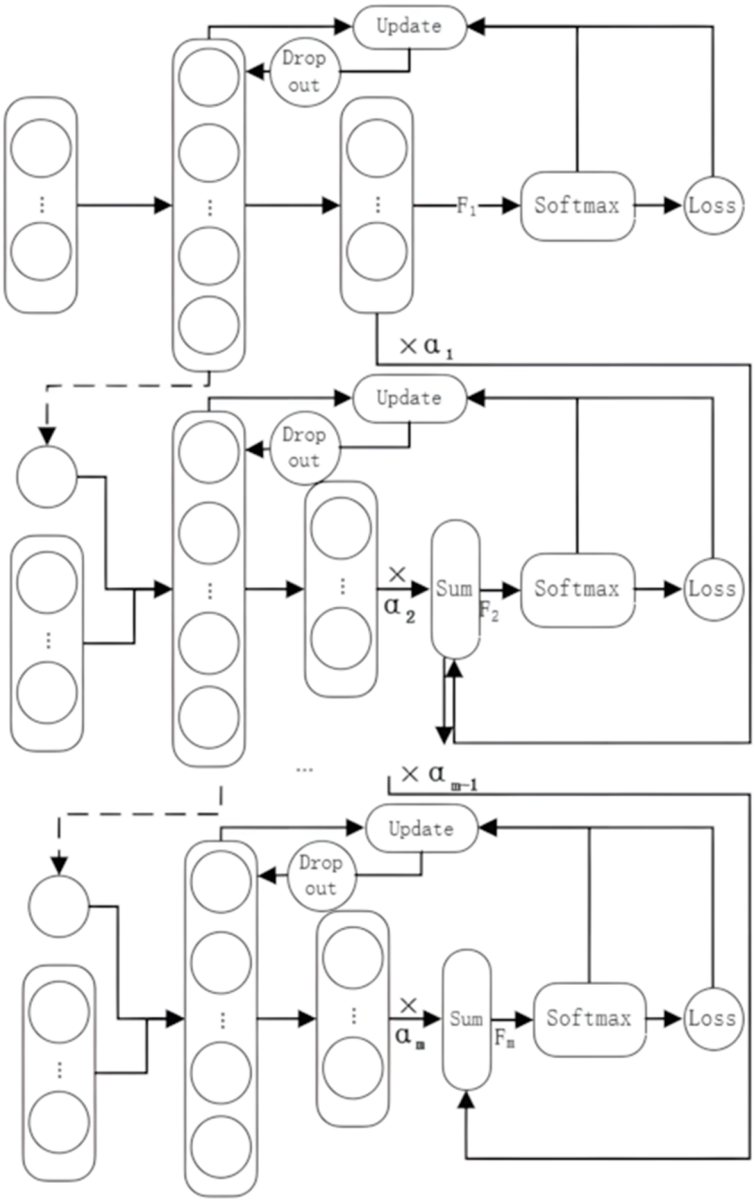


**Fig 2.** Model structure diagram

At this moment a reverse parameter update is performed based on the classification results, utilizing Eq. (4), with η being the learning rate:

$$\alpha_{ij} = \alpha_{i-1,j} - \eta \frac{\partial L(y_j, \tilde{y}_j)}{\partial \alpha_{i-1,j}} \tag{4}$$

and Eq. (5), with λ being the proportion of discarded neurons added to the Dropout layer:

$$\theta_{ij} = \lambda\left(\theta_{i-1,j} - \eta \frac{\partial L(y_j, \tilde{y}_j)}{\partial \theta_{i-1,j}}\right) \tag{5}$$

In the update parameter equation, L represents the cross-entropy loss function, where C is the number of categories:

$$L(y, \tilde{y}) = -\sum_i^C y_i \cdot \log(\tilde{y}_i) \tag{6}$$

After several rounds of updating, the final results for multiple base models are:

$$softmax(F_j(x)) \tag{7}$$

A final vote was taken to obtain the following results:

$$\hat{y} = mode(y_1, \ldots, y_m) \tag{8}$$

The gradient optimization method used in this algorithm updates the model parameters by optimizing the loss function, effectively accelerating the model learning process. The introduction of Dropout regularization method is added to reduce the risk of overfitting and make each base learner more independent. Voting mechanism is used as a core part to enhance robustness. Combining multiple strategies enhances the final decision accuracy.

# 4. Experimental Design and Experimental Results

## 4.1. Experimental Details

### 4.1.1. Dataset

Six synthetic datasets and four real datasets generated with MOA stream generator are selected. The dataset information is in Table 1.

**Table 1.** The dataset used for the experiment

| Dataset | Sample /k | Feature dimension | Class label | Drift type |
|---|---|---|---|---|
| Sea | 100 | 3 | 2 | Gradual drift |
| Hyperplane | 100 | 10 | 2 | Incremental drift |
| RBFBlips | 100 | 20 | 4 | Abrupt drift |
| LED_abrupt | 100 | 24 | 10 | Abrupt drift |
| LED_gradual | 100 | 24 | 10 | Gradual drift |
| Tree | 100 | 30 | 10 | Abrupt drift |
| Electricity | 45.3 | 6 | 2 | Unknow |
| Kddcup99 | 494 | 41 | 23 | Unknow |
| Covertype | 581 | 54 | 7 | Unknow |
| Weather | 95.1 | 9 | 3 | Unknow |

### 4.1.2. Evaluation Criteria

This paper evaluates and analyzes the performance of the model from several aspects, mainly including the model's performance in dealing with unknown data (i.e., generalization ability).

(1) Average real-time accuracy is a measure of the average level of model performance over the entire time course, defined as:

$$Avgracc = \frac{1}{T}\sum_{t=1}^{T} acc_t \tag{9}$$

where $acc_t$ represents the accuracy of the model at time node $t$. A higher average real-time accuracy indicates a better real-time classification ability of the model. To cope with datasets with imbalanced categories, $acc_t$ uses the Balanced Accuracy Score [18], which calculates the average of the accuracies for each category:

$$acc_t = \frac{1}{k}\sum_{i=1}^{k} \frac{c_{ii}}{c_{*i}} \tag{10}$$

where $k$ denotes the total number of categories, $c_{ii}$ represents the elements in row $i$ and column $i$ of the confusion matrix, and $c_{*i}$ refers to the sum of all elements in column i of the confusion matrix.

(2) The final cumulative accuracy is a measure of the overall performance of the model over the entire time period and is defined as:

$$Finacc = \frac{1}{T\times n}\sum_{t=1}^{T} n_t \tag{11}$$

Where $n$ represents the number of samples in the data block and $n_t$ is the number of samples that the model is correct at the time node of moment $t$. The final cumulative accuracy reflects the overall performance of the model.

### 4.1.3. Experimental Environment and Setup

The computer used for the experiments was configured with an Intel(R) Core (TM) i9-13900HX 2.20 GHz processor, 16 GB of DDR5 RAM, and an NVIDIA GeForce RTX 4060 Laptop 8 GB graphics card. The data block size is set to 100, the activation function is chosen to be ReLU, the learning rate is fixed to 0.01, the threshold $\lambda$ is set to 0.8, the number of base learners is 3, and the number of hidden layer nodes is 100.

## 4.2. Experimental Results

**Table 2.** Comparison of average real-time accuracy of different methods

| Dataset | Average real-time accuracy (ranked) | | | | | | |
|---|---|---|---|---|---|---|---|
| | DNN2 | DNN4 | DNN8 | Resnet | Highway | HBP | EMVM_ATCD |
| Sea | 0.6217(7) | 0.6782(6) | 0.6782(5) | 0.7698(4) | 0.8064(2) | 0.8056(3) | **0.8462(1)** |
| Hyperplane | 0.8913(5) | 0.8930(4) | 0.8905(6) | 0.8892(7) | 0.9053(2) | 0.8969(3) | **0.9262(1)** |
| RBFBlips | 0.9460(4) | 0.9448(6) | 0.9451(5) | 0.9406(7) | 0.9581(2) | 0.9517(3) | **0.9720(1)** |
| LED_abrupt | 0.6122(2) | 0.6030(4) | 0.5741(7) | 0.5912(6) | 0.6016(5) | 0.6068(3) | **0.6183(1)** |
| LED_gradual | 0.6206(3) | 0.6134(5) | 0.5873(8) | 0.6067(7) | 0.6116(6) | 0.6191(4) | **0.6310(1)** |
| Tree | 0.3301(3) | 0.3268(4) | 0.2378(8) | 0.2877(6) | 0.2952(5) | 0.2799(7) | **0.6007(1)** |
| Electricity | 0.6657(2) | 0.6464(4) | 0.6031(7) | 0.5821(8) | 0.6337(5) | 0.6115(6) | **0.6939(1)** |
| Kddcup99 | 0.8796(2) | 0.7186(6) | 0.4763(7) | 0.6534(5) | 0.7537(4) | 0.7670(3) | **0.9379(1)** |
| Covertype | 0.5251(7) | 0.5739(6) | 0.6243(4) | 0.6183(5) | 0.6354(3) | 0.6465(2) | **0.8452(1)** |
| Weather | 0.8478(2) | 0.8050(6) | 0.8057(5) | 0.8034(7) | 0.7813(8) | 0.8139(3) | **0.8393(1)** |
| Average ranking | 3.7 | 5.1 | 6.2 | 6.2 | 4.2 | 3.7 | 1.0 |

Table 2 shows the average real-time accuracies and rankings of several models on different datasets.The EMVM_ATCD method performs well on all datasets, and the average real-time accuracy ranking is always in the first place, with its stability and adaptability. Its overall average ranking is 1.0, indicating that its combined performance on multiple

datasets far exceeds that of other models. The method's relatively good performance and advantages in datasets such as Sea, Hyperplane, and RBFBlips are especially obvious, and EMVM_ATCD is able to capture and adapt to their changes better.

The average ordinal value is for the j-th algorithm, ranked $r_i^j$ on the $i$ dataset, the average ordinal value $R_j = \frac{1}{n}\sum_i r_i^j$ of the j-th algorithm.

Table 3 demonstrates the comparison of the final cumulative accuracies of the different methods on each dataset, as well as the corresponding rankings. Taken together, the EMVM_ATCD method performs outstandingly on all datasets with an average ranking of 1.5, which is significantly better than the other methods. The datasets Sea, Hyperplane, RBFBlips, and LED_abrupt, etc., on which the EMVM_ATCD method performs particularly well, all of them obtaining high final cumulative accuracies and rankings, which proves that the method is highly adaptable and effective in dealing with these datasets.

**Table 3.** Comparison of final cumulative accuracies of different methods

| Dataset | Final cumulative accuracy (ranked) | | | | | | |
|---|---|---|---|---|---|---|---|
| | DNN2 | DNN4 | DNN8 | Resnet | Highway | HBP | EMVM_ATCD |
| Sea | 0.6579(7) | 0.7303(6) | 0.7349(5) | 0.8062(4) | 0.8307(3) | 0.8309(2) | **0.8472(1)** |
| Hyperplane | 0.8912(5) | 0.8933(4) | 0.8908(6) | 0.8887(7) | 0.9053(2) | 0.8970(3) | **0.9260(1)** |
| RBFBlips | 0.9479(6) | 0.9489(5) | 0.9478(7) | 0.9433(8) | 0.9600(3) | 0.9542(4) | **0.9719(1)** |
| LED_abrupt | 0.6125(3) | 0.6033(5) | 0.5747(8) | 0.5915(7) | 0.6016(6) | 0.6069(4) | **0.6190(1)** |
| LED_gradual | 0.6208(3) | 0.6137(5) | 0.5879(8) | 0.6067(7) | 0.6117(6) | 0.6186(4) | **0.6319(1)** |
| Tree | 0.5589(3) | 0.5505(4) | 0.4725(8) | 0.5069(6) | 0.5340(5) | 0.5042(7) | **0.6772(1)** |
| Electricity | 0.6821(2) | 0.6715(4) | 0.6411(7) | 0.6303(8) | 0.6621(5) | 0.6504(6) | **0.6894(1)** |
| Kddcup99 | **0.9832(1)** | 0.9195(6) | 0.7812(7) | 0.9276(5) | 0.9414(4) | 0.9823(2) | 0.9811(3) |
| Covertype | 0.6984(7) | 0.7336(6) | 0.7677(5) | 0.7730(4) | 0.7824(2) | **0.7903(1)** | 0.7762(3) |
| Weather | **0.8872(1)** | 0.8743(5) | 0.8754(4) | 0.8708(6) | 0.8362(7) | 0.8824(3) | 0.8835(2) |
| Average ranking | 3.8 | 5.0 | 6.5 | 6.2 | 4.2 | 3.7 | 1.5 |

# 5. Conclusion

Classical algorithms have average performance when computing data containing conceptual drift, an integrated multi-model voting method adapted to conceptual drift is proposed to cope with the conceptual drift problem by integrating a single hidden-layer neural network to take advantage of multi-models and to improve accuracy in the changing environment of data. The neural network has a simple structure, which can save a lot of computational resources, and is able to quickly learn and adapt to changes in data distribution. A voting mechanism is used along with weighted prediction computation to increase the stability of the model and overcome the limitations of a single model. The results show that the proposed algorithm in the dataset containing conceptual drift, the improvement of the average real-time accuracy and the final cumulative accuracy indicate that the EMVM_ATCD method can flexibly cope with a variety of types of data and realize the overall performance improvement.

# References


[1] RUTKOWSKI L, JAWORSKI M, DUDA P.Stream data mining: algorithms and their probabilistic properties[M]. Germany: Springer Publishing Company, 2020:13-33.

[2] GABER M M, ZASLAVSKY A, KRISHNASWAMY S.A survey of classification methods in data streams[J].Data Streams: Models and Algorithms, 2007, 31: 39-59.

[3] LECUN Y, BOTTOU L, BENGIO Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[4] ELMAN J L. Finding structure in time[J]. Cognitive science, 1990, 14(2): 179-211.

[5] HOCHREITER S. Long Short-term Memory[J]. Neural Computation MIT-Press, 1997, 9(8), 1735-1780.

[6] KRIZHEVSKY A, SUTSKEVER I, HINTON G E. ImageNet classification with deep convolutional neural networks[J]. Communications of the ACM, 2017, 60(6): 84-90.

[7] Xu, R.; Cheng, Y.; Liu, Z.; Xie, Y.; Yang, Y. Improved Long Short-Term Memory based anomaly detection with concept drift adaptive method for supporting IoT services. Futur. Gener. Comput. Syst. 2020, 112, 228–242.

[8] Soleymani, F.; Paquet, E. Financial portfolio optimization with online deep reinforcement learning and restricted stacked autoencoder—DeepBreath. Expert Syst. Appl. 2020, 156, 113456.

[9] Wang, X.; Chen, W.; Xia, J.; Chen, Z.; Xu, D.; Wu, X.; Xu, M.; Schreck, T. ConceptExplorer: Visual Analysis of Concept Drifts in Multi-source Time-series Data. In Proceedings of the IEEE Conference on Visual Analytics Science and Technology (VAST), Salt Lake City, UT, USA, 25–30 October 2020; pp. 1–11. [CrossRef]

[10] Huang G B, Zhu Q Y, Siew C K. Extreme learning machine: theory and applications[J]. Neurocomputing, 2006, 70(1-3): 489-501.

[11] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J]. nature, 1986, 323(6088): 533-536.

[12] Bernardo A, Della Valle E. Smote-ob: Combining smote and online bagging for continuous rebalancing of evolving data streams[C]//2021 IEEE International Conference on Big Data (Big Data). IEEE, 2021: 5033-5042.

[13] Lv Y, Peng S, Yuan Y, et al. A classifier using online bagging ensemble method for big data stream learning[J]. Tsinghua science and technology, 2019, 24(4): 379-388.

[14] Bayram B, Köroğlu B, Gönen M. Improving fraud detection and concept drift adaptation in credit card transactions using incremental gradient boosting trees[C]//2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA). IEEE, 2020: 545-550.

[15] Wang K, Lu J, Liu A J, et al. Evolving gradient boost: A pruning scheme based on loss improvement ratio for learning under concept drift [J]. IEEE Transactions on Cybernetics, 2021, 151: 1-14.

[16] Wang K, Lu J, Liu A J, et al. Elastic gradient boosting decision tree with adaptiveiterations for concept drift adaptation [J]. Neurocomputing, 2022, 491: 288-304.

[17] Kang Q, Chen X S, Li S S, et al. A noise-filtered under-sampling scheme for imbalanced classification[J]. IEEE transactions on cybernetics, 2016, 47(12): 4263-4274.

[18] BRODERSEN K H, ONG C S, Stephan K E, et al. The balanced accuracy and its posterior distribution[C]//2010 20th international conference on pattern recognition. IEEE, 2010: 3121-3124.