A MODIFIED LEARN++.NSE ALGORITHM FOR DEALING WITH CONCEPT DRIFT

FAN DONG[†*], JIE LU[*], GUANGQUAN ZHANG[*], KAN LI[†]

[†]*School of Computer, Beijing Institute of Technology, Beijing, 100081, P. R. China*

[*]*Decision Systems & E-Service Intelligence Research (DeSI) Laboratory,*
Center for Quantum Computing and Intelligent System (QCIS),
Faculty of Engineering and Information Technology,
University of Technology, Sydney, NSW 2007 Australia

Concept drift is a very pervasive phenomenon in real world applications. By virtue of variety change types of concept drift, it makes more difficult for learning algorithm to track the concept drift very closely. Learn++.NSE is an incremental ensemble learner without any assumption on change type of concept drift. Even though it has good performance on handling concept drift, but it costs high computation and needs more time to recover from accuracy drop. This paper proposed a modified Learn++.NSE algorithm. During learning instances in data stream, our algorithm first identifies where and when drift happened, then uses instances accumulated by drift detection method to create a new base classifier, and finally organized all existing classifiers based on Learn++.NSE weighting mechanism to update ensemble learner. This modified algorithm can reduce high computation cost without any performance drop and improve the accuracy recover speed when drift happened.

1. Introduction

Concept drift refers to a change in class definitions and a change in underlying data distribution. Data in the non-stationary environments always involve with concept drift, which is a very pervasive phenomenon in real world data-stream applications, such as intrusion detection in telecommunications, email spam filtering and credit card fraud. Learning from such non-stationary environment becomes an indispensable problem to apply machine-learning techniques on real world application. Normally, the change type of concept drift can be normally divided into three categories: sudden drift, gradual drift and reoccur contexts. While Minku, White and Yao [1] presented another drifts categorization based on multiple criteria – drift speed, severity, predictability, frequency and recurrence. In most of real world applications, data is organized in the form of data-stream, in which the nature or rate of drift is various and convoluted [2]. It makes more challenge to learning knowledge from data involving concept drift.

According to a literature review [3], the approaches that deal with concept drift can be divided into two groups: trigger based or evolving. Trigger based approaches focus on when a drift happened. After a drift is detected, there is a mechanism that indicates learner updating for adapting current data environment. Then, the learner will reset and retrain its model by using recently instances that can reflect current data distribution. Drift detection method is usually conducted by statistical theory that monitors the underlying data distribution [4, 5] or the outputs (error) of learners [6, 7]. Evolving approaches can adapt to non-stationary environment without indicating any drift occur point explicitly. Ensemble, which is the most popular evolving technique for reacting concept drift, usually use majority weighting strategies for combining several base classifiers to make a final decision [8, 9]. However these algorithms above usually have a restrictive assumption that there is no reoccurring drift in the data stream.

Learn++.NSE [10] is an incremental ensemble algorithm that assumes data batches are incrementally arrived. Its distinctively weighting strategy and base classifiers association mechanism can track concept drift without any assumption on change type of drift. However, compared with other learning algorithm, Learn++.NSE can react concept drift well with sophisticated weighting strategies, this ensemble algorithm has limitations of high computation costs and significant accuracy drop after drift occurred.

Motivated by these issues above, we propose a modified Learn++.NSE algorithm for dealing with concept drift. Our modification is that integrating drift detection method into Learn++.NSE. By monitoring distance-error-rate of the ensemble learner (Early Drift Detection Method), our modified algorithm has ability of indicating where and when drift happened, so that we can create a new base classifier on demand. At last, we use weighting mechanism of Learn++.NSE to make finally decision. The main contributions of this paper are: 1) maintaining good performance with less computation costs on handling concept drift problem 2) improving the accuracy recover speed when drift happened.

This paper is organized as follows. Section 2 discusses related works including the Learn++.NSE algorithm. Section 3 proposes a modified Learn++.NSE algorithm. Section 4 presents the conclusion, with a discussion of future work.

2. Related Works

2.1. *The Problem of Concept Drift*

The data involving concept drift, compared to data under stationary environment, is extended with time dimension. Learner dealing with concept drift problem must have an assumption that is uncertainty about the future [3]. A sequence data $\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_n$ ($\boldsymbol{X} \in \Re^p$ is a vector in $p-$dimensional feature space), observed from stationary environment with corresponding class label $y \in \Re^1$, could be used for prediction unlabeled data $\boldsymbol{X}_{n+1}$ by using Bayesian posterior probability of a class, $P(y|\boldsymbol{X}) = P(\boldsymbol{X}|y)P(y)/P(\boldsymbol{X})$. While concept drift can be formally defined as data collected from unstable environment in which the underlying posterior probability changes as time shift, i.e., $P_t(y|\boldsymbol{X}) \neq P_{t+1}(y|\boldsymbol{X})$ [10].

2.2. *EDDM: Early Drift Detection Method*

EDDM [7], inspired by Drift Detection Method (DDM) [6], is a drift detection method that monitors the online error-rate of the learning algorithm. Different with DDM, this method considers the distance between two errors classification, which can improve the ability of identifying slow gradual concept drift. This method adopt that a significant decrease in the distance suggests that a drift may occurred. It calculates the average distance between two errors (p_i') and its standard deviation (s_i'). In addition, it obtains $p'_{\max}$ and $s'_{\max}$ when $p_i' + 2 \cdot s_i'$ reaches its maximum value. This method defines two thresholds α for the warning level and β for the drift level. If $(p_i' + 2 \cdot s_i')/(p'_{\max} + 2 \cdot s'_{\max}) < \alpha$, the instances will be cached in advanced of a possible change of context. When $(p_i' + 2 \cdot s_i')/(p'_{\max} + 2 \cdot s'_{\max}) < \beta$, the new model is learnt using the instances cached since the warning level triggered, and then $p'_{\max}$ and $s'_{\max}$ are reset.

2.3. *Learn++.NSE Algorithm*

Learn++.NSE [10] is an incremental learning algorithm for non-stationary environments. Learn++.NSE is a passive ensemble learner that pays no attention to when the drift happened. It assumes that data are incremental received in batches. For each incoming data batch, this algorithm creates a new classifier, and then adjusts each existing classifier weighting based on its time-adjusted accuracy on latest data batch. The final classification decision is determined by weighted majority voting of all base classifiers. This algorithm can track the concept drift closely without any assumption on change type of concept drift.

3. The Modified Learn++.NSE Algorithm

Even though Learn++.NSE algorithm has good performance on dealing variety change type of concept drift, it still has limitations. One of limitation is Learn++.NSE algorithm has no explicitly drift detection, which results that it has a significant accuracy drop after a drift happened. For Learn++.NSE algorithm is a batch ensemble learner, another limitation is that its learning performance will highly depend on the training examples in every coming data batch.

Considering the limitation of Learn++.NSE, we proposes a modified Learn++.NSE algorithm which integrating drift detection method to maintain learning performance of dealing concept drift without too much computation costs. Because of Learn++.NSE already has a sophisticated weighting strategy to deal with various types of concept drift, we decide to use a simple and effective drift detection method, EDDM, incorporating with Learn++.NSE. We assume that the instances arrive one at one time. Learner must predict the label of new instance when it arrived. Once the prediction has been made, learner can access the true label of instance and utilize it to update the learning model. The pseudocode of algorithm is as follow:

[Modified Learn++.NSE Algorithm]

Initialization:

- Training data $\{\boldsymbol{X}_t \in \Re^p, y_t \in \Re^1 = \{1, \cdots, c\}\}, t = 1, \cdots, m$
- Supervised learning algorithm Base Classifier $\boldsymbol{C}$
- Sigmoid parameters a (slope) and b (infliction point)
- Drift detection thresholds α (warning level) and β (drift level)

Call Base Classifier with training data,
obtain $\boldsymbol{C}_j$: $\boldsymbol{X} \rightarrow y$, $W_j^k = \boldsymbol{1}$, where $j = \boldsymbol{1}, k = \boldsymbol{1}$
$buf \leftarrow$ empty //buffer that stores instances involving possible drift
Ensemble learner $\boldsymbol{H}^k \leftarrow \boldsymbol{C_1}$ with W_1^1, where $k = 1$
while true do
  $\boldsymbol{X}_i \leftarrow$ the latest instance
  $y'_{\boldsymbol{i}} \leftarrow$ get prediction $\boldsymbol{H}^k(\boldsymbol{X}_i)$ by using Weighted Majority
  drift $\leftarrow$ DriftDection$(y'_{\boldsymbol{i}}, y_{\boldsymbol{i}}, \alpha, \beta)$
  if drift == warning level **then**
    cache $\boldsymbol{X}_i$ into buf
  else drift == drift level **then**
    $j = j + \boldsymbol{1}, k = k + \boldsymbol{1}$
    call **Base Classifier** with each $\boldsymbol{X}_l \in buf$, obtain $\boldsymbol{C}_j$

13: $E^k \leftarrow$ compute error of the current ensemble on each $\boldsymbol{X}_l \in buf$, where the error is $\boldsymbol{H}^{k-1}(\boldsymbol{X}_l) \neq y_l$
14: use E^k to update and normalize instance weights D^k
15: use D^k evaluate all existing classifier $\boldsymbol{C}_1, \cdots, \boldsymbol{C}_j$ normalized error θ_j^k on each instances in the buf
16: $\bar{\theta}_j^k \leftarrow$apply sigmoidal weights ω_j^k to normalized classifier errors θ_j^k
17: calculate each classifier $\boldsymbol{C}_1, \cdots, \boldsymbol{C}_j$ voting weights W_j^k by log-normalized reciprocals of the adjusted normalized error $\bar{\theta}_j^k$
18: construct a new ensemble $\boldsymbol{H}^k$ with classifier $\boldsymbol{C}_1, \cdots, \boldsymbol{C}_j$ and corresponding voting weights W_j^k
19: $buf \leftarrow$ empty //reset the buffer
20: **end if**
21: **end while**

The core idea of modified Learn++.NSE algorithm is using drift detection method to indicate when a drift happened and accumulate relevant instances involving concept drift. The drift detecting method we used is EDDM. It is possible that replacing with other drift detection method. After drift has occurred, we use stored instances to obtain a new base classifier, adjust all existing classifiers' voting weights and update existing ensemble learner. The voting weights adjustment strategy from line 13 to line 17 are inherited from Learn++.NSE.

Compared with original Learn++.NSE algorithm, our modified algorithm has advantage in learning time. The learning time for Learn++.NSE increase linearly since a new base classifier will be created as new batch data arrived, and then adjust weights of all existing base classifier to update ensemble learner. If the learning time is increase linearly, it must take too much time to learn knowledge when the learner has been running for a long time. Moreover, it may waste too much computation resource for a period of data stream without drift. Our modified algorithm only creates new base classifier and adjusts weights when a drift is detected, it will reduces high computation costs and learning time.

Learn++.NSE algorithm assumes that data are incremental arrived in batch. For Learn++.NSE can only update its model when next training instances of data batch arrived, this assumption leads that accuracy drop of learner will be seriously when a drift happened in data batch since many instances are misclassified. By processing instance once a time with drift detection method, our algorithm can identify drift without any delay, and update learner quickly with instances that contains a possible change of context. Our modified Learn++.NSE algorithm can track concept drift timely with less misclassification.

4. Conclusion and Future Work

We present a modified Learn++.NSE algorithm for dealing with concept drift. The novelty of the algorithm is eliminating the limitations of Learn++.NSE algorithm, which are has no explicitly drift detection and only deal with data arrived in batch. Our modified Learn++.NSE algorithm can maintain good performance on handling concept drift with less computation costs, and shortening the recover time of accuracy drop when drift happened. In the future, an empirical study will be made to prove our proposed algorithm is effective，and diversity adjustment mechanism could be incorporated into our algorithm.

Acknowledgments

This work is supported by the Australian Research Council (ARC) under discovery grant DP140101366.

References

1. Minku, L.L., A.P. White, and Y. Xin, *The Impact of Diversity on Online Ensemble Learning in the Presence of Concept Drift.* Knowledge and Data Engineering, IEEE Transactions on, 2010. **22**(5): p. 730-742.
2. Tsymbal, A., *The problem of concept drift: definitions and related work*. 2004, Computer Science Department, Trinity College Dublin, Ireland.
3. Zliobaite, I., *Learning under concept drift: an overview*. 2009, Faculty of Mathematics and Informatics, Vilnius University, Vilnius, Lithuania.
4. Lu, N., G. Zhang, and J. Lu, *Concept drift detection via competence models.* Artificial Intelligence, 2014. **209**(0): p. 11-28.
5. Bifet, A. and R. Gavalda. *Learning from Time-Changing Data with Adaptive Windowing*. in *In Proceedings of the Seventh SIAM International Conference on Data Mining (SDM'07)*. 2007. Minneapolis, MN, USA: SIAM.
6. Gama, J., et al., *Learning with drift detection*, in *Advances in Artificial Intelligence–SBIA 2004*. 2004, Springer. p. 286-295.
7. Baena-García, M., et al., *Early drift detection method.* 2006.
8. Chen, S. and H. He, *Towards incremental learning of nonstationary imbalanced data stream: a multiple selectively recursive approach.* Evolving Systems, 2011. **2**(1): p. 35-50.
9. Ditzler, G. and R. Polikar, *Incremental learning of concept drift from streaming imbalanced data.* IEEE Transactions on Knowledge and Data Engineering, 2013. **25**(10): p. 2283-2301.
10. Elwell, R. and R. Polikar, *Incremental learning of concept drift in nonstationary environments.* IEEE Transactions on Neural Networks, 2011. **22**(10): p. 1517-1531.