International Journal of Electrical and Computer Engineering (IJECE)
Vol. 6, No. 4, August 2016, pp. 1811~1817
ISSN: 2088-8708, DOI: 10.11591/ijece.v6i4.10255

Incremental Learning on Non-stationary Data Stream Using Ensemble Approach

Meenakshi Anurag Thalor, Shrishailapa Patil

Article Info

Article history:

Received Jan 20, 2016
Revised Mar 15, 2016
Accepted Apr 1, 2016

Keyword:

Concept drift
Ensemble
Incremental learning
Non-stationary data
Stream

ABSTRACT

Incremental Learning on non stationary distribution has been shown to be a very challenging problem in machine learning and data mining, because the joint probability distribution between the data and classes changes over time. Many real time problems suffer concept drift as they changes with time. For example, an advertisement recommendation system, in which customer's behavior may change depending on the season of the year, on the inflation and on new products made available. An extra challenge arises when the classes to be learned are not represented equally in the training data i.e. classes are imbalanced, as most machine learning algorithms work well only when the training data is balanced. The objective of this paper is to develop an ensemble based classification algorithm for non-stationary data stream (ENSDS) with focus on two-class problems. In addition, we are presenting here an exhaustive comparison of purposed algorithms with state-of-the-art classification approaches using different evaluation measures like recall, f-measure and g-mean.

Corresponding Author:

1. INTRODUCTION

Nonstationary data is time series data where data at time t is not equal to data at time t+1 The time series Y_t is nonstationary if for all values, and every time period, Eq. (1) and Eq. (2) are true.

$E(Y_t) \neq \mu$ (not having constant mean) (1)

$Var(Y_t) \neq \sigma^2$ (not having constant variance) (2)

Conventional data mining and machine learning [1] algorithms assumes that each dataset is produced from a single, static and hidden function. That is, the function (model/classifier) generating data at training time is the same as that of testing time. Whereas in data stream, data is continuously coming and the function which generating instances at time *t* need not be the same function at time *t+1*. This difference in the underlying function is called as concept drift [2]. Thus, past data may become irrelevant for the current context, and it is defined by Eq. (3)

$$p_{tr}(y|x) \neq P_{tst}(y|x) \; and \; P_{tr}(x) = P_{tst}(x) \quad (3)$$

In recent data mining applications, drift can occur at any moment of time, so it is necessary to take some measures to handle drifted data streams. In this paper we are using ensemble based incremental learning algorithm to handle aforemention phenomena.In ensemble based classification [3] a set of classifiers

whose individual predictions are combined in some way to classify unseen data. The approach in ensemble systems [4] is to create many classifiers, and combine their outputs in such a way that this combination will improve the performance as compare to single classifier. Figure 1 shows the basic steps of ensemble based classification.

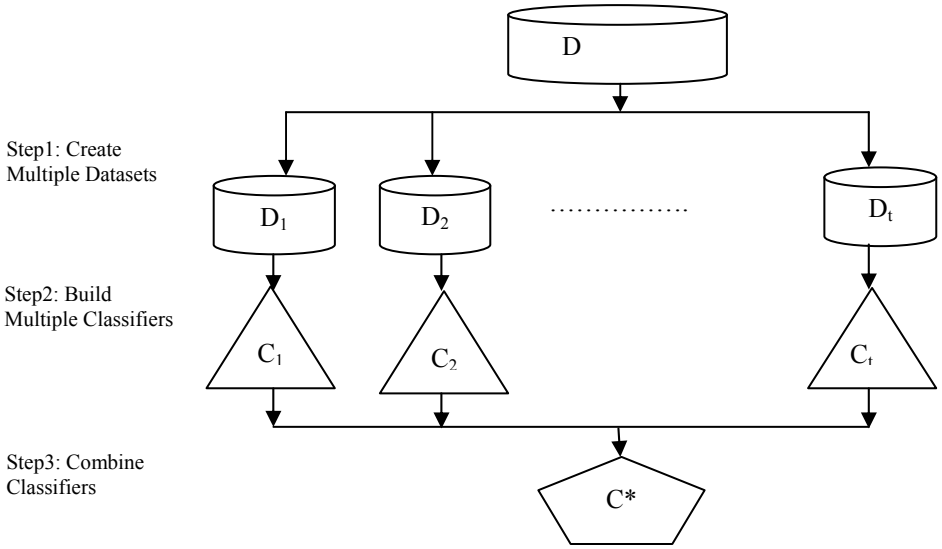

Figure 1. Ensemble based Learning

So this work introduces classification algorithm on non-stationary data using ensemble based approach which will explicitly and simultaneously address the aforementioned phenomena. Classifier used for learning of non-stationary data stream generally uses one of the following methodologies.

1.1. Adaptive Methods

These are truly incremental algorithms [5],[6] as they learn data incrementally, as the instances arrive (instance-by-instance, and efficiently with a single pass through the data). Adaptive methods are generally used for concept drift detection algorithms. The first is to detect concept drift, such as through the use of novelty detection algorithms, and upon detection, adapt the classifiers to this change of concept.

1.2. Wrapper Methods

These require the data must in some way to be collected into chunk so that a traditional classifier can be used for learning purpose. Wrapper methods are generally used for passive drift detection algorithms. In passive drift detection algorithms it is assumed that drift occurs, and model is constructed taking this in to assumption ; the actual level of concept drift or even if it actually does occur may not be measured. Wrapper methods have following advantages over adaptive ones:

- We can use traditional classifiers type like support vector machine for learning purpose.
- We can construct ensemble in parallel.
- Speed of ensemble design is fast as compare to adaptive as they reliance on change detection algorithms.
- Ability to deal with reoccurring concepts – to learn on past batches of data and reuse this information to classify new instances on new concepts with similar class distributions as old concepts.
- We can use ensemble methods to combine classifiers trained from different intervals of time which further improves upon the performance over the single classifier. Thus ensembles are often built from past subsets (batches) of data that can be reused to classify new instances from the new concept, similar to the class distribution of the old concept.

Performance of all these wrapper algorithms dependent upon the size of the data chunks (block/batch). Bigger blocks can results in accurate classifiers as classifiers are getting more data for training, but can contain too many different concepts drift. Whereas smaller blocks are better for drifted data stream, but usually lead to poorer classifiers as training data is less.

2. RELATED WORK

The first experiment of ensembles in data streams was the purposed by Street and Kim with their Streaming Ensemble Algorithm [7] (SEA) where a chunk of d instances are read from data stream and used to build a classifier. As fixed size of ensemble was used, so they compare new generated classifier against a pool of previously trained classifiers (from previous chunk), and if it current classifier improves the quality of ensemble it is included at the cost of the worst classifier. SEA uses a simple majority vote and may not be able to perform in recurring environments.

Wang et al. proposed Accuracy Weighted Ensemble [8] (AWE) of classifiers on each incoming data chunk and use that chunk to evaluate the performance of all existing classifiers in the ensemble. The weight of each classifier is the difference of error rate of a random classifier and the mean square error of the classifier for the current chunk. The mean square errors of old classifiers are high, and thus the weights of old classifiers are small.

Brzezinski and Stefanowski proposed the Accuracy Updated Ensemble [9] (AUE) which is derived from AWE .It uses same principles of chunk-based ensembles but with incremental base components/classifiers. It not only builds new classifiers, but also conditionally updates existing classifiers on new chunk rather than just adjusting their weights. The updating of existing base classifiers makes AUE better than AWE in case of gradual drift but conditionally updating of base classifiers is less accurate for sudden drift.

Robi Polikar et al. proposed Learn++.NSE [10]-[14] (Nonstationary Environment) which generates classifiers sequentially using batches of examples/instances (Not true online learner as it converts the online data stream into a series of chunks of a fixed size). At each time step one new classifier is trained on recent distribution, using an instance weighting distribution. In Learn++.NSE each classifier's weight is computed using a weighted average of its prediction error on old and current batch and finally uses weighted majority voting to obtain ensemble's output.

Most recently, Brzezinski and Stefanowski proposed AUE2 [15] introduces a new weighting function, does not require cross-validation on the existing classifiers, does not keep a classifier buffer, prunes its base learners, and always unconditionally updates its components. Classifiers are updated after every chunk, so they can react to gradual drifts. It can react to sudden drifts and gradually drifts but not for reoccurring concepts. Compared to Learn++.NSE, AUE2 incrementally trains existing component classifiers, retains only k of all the created components, and uses a different weighting mechanism which ensures that components will have non-zero weights.

3. ENSEMBLE FOR NON-STATIONARY DATA STREAM (ENSDS)

ENSDS uses a similar organizational framework as Learn^{++}.NSE that is we are building an ensemble of classifiers from data arrived at time and evaluating the performance of existing classifiers on recent data then all generated classifiers are combined by using weighted majority voting to provide the predictions of unseen data. One of the major differences in ENSDS is we are not updating a set of weights for each instance rather only uniform weight is considered. The ENSDS algorithm is described in detail below, with its mathematical model given in Figure 2.

Input: For each dataset D^t where t=1,2,…..
Training data: $\{x_i^t \in X;\ y_i^t \in Y = \{1, \ldots., c\}\}, i = 1 \ldots.. m$ instances
Description: Supervised learning algorithm to handle Non-stationary data stream

Pseudo code:
Do for t=1, 2…

1. Initialize $D^t(i)$=1/m,∀i,
2. Call base classifier with D^t, obtain $h_k: X \rightarrow Y$
3. Evaluate all existing classifiers (h_k) on D^t

$$\varepsilon_k^t = \sum_{i=1}^{m} D^t(i).[|h_k(x_i) \neq y_i|] \text{for } k = 1, \ldots, t$$

If $\varepsilon_{k=t}^t > \frac{1}{2}$ generate a new h_k

If $\varepsilon_{k<t}^t > \frac{1}{2}$ set $\varepsilon_k^t = 1/2$

4. Compute the weight for kth classifier h_k

$$Sig_k^t = \frac{1}{1+e^{-a(t-k-b)}}$$

$$w_k^t = \begin{Bmatrix} 1 & t = k \\ \frac{Sig_k^t}{Sig_k^t + \sum_{j=1}^{t-1} w_k^{t-j}}, & otherwise \end{Bmatrix}$$

5. Calculate classifier voting weights

$$\text{Voting } w_k^t = \ln\left(\frac{1}{\sum_{j=1}^{t} w_k^j \ \varepsilon_k^j}\right) for\ k = 1 \ldots t$$

6. Obtain the final hypothesis

$$H^t(x_i) = \arg\max_c \sum_k w_k^t.[|h_k(x_i) = c|]$$

Figure 2. The Mathematical model of the algorithm ENSDS

We are building a k classifier on data drawn from the current training dataset D^t. After formation of kth classifier, the performance of existing classifiers will be evaluated over the current training dataset D^t and we will get ε_k^t which is error of kth classifier on current D^t. If error generated by current classifier is more than .5 that is half of the predictions are wrong then generate a new classifier for current distribution. If error generated by one of the previous classifier is more than .5 then set its $\varepsilon_k^t = 0.5$ .We are not normalizing the ε_k^t as its value remains between 0 to 0.5 and voting power of a classifier having $\varepsilon_k^t = .5$ will remain low. A nonlinear sigmoid function is used to set weight of a classifier. Because of this if a classifier will be evaluated more than once then its sigmoid weight will get increased. The weight to a classifier is assigned based on its performance on previous distributions as well as on recent distribution so weighted average of classifier is computed in step 4. When a classifier is generated it's $w_k^t = 1$, after its evaluation on recent environment its w_k^t gets keep updated. If a classifier does not performs well on recent environment, then its weighted error ($w_k^t.\varepsilon_k^t$) will gets increased. In step 5 the weight error average is computed to determine the voting weight of classifiers. The voting power of each classifier is computed using logarithm of the inverse of its weighted error average .If weighted error average is high a classifier will get less power of voting. The time complexity of ENSDS is O(t*k*O(x*m)+k*t*m) which is less than Learn^{++}.NSE. Where O(x*m) is the time complexity of Naïve Bayes classifier, x is number of features and m is number of instances in training set, k indicates number of classifiers, t indicates number of data chunks to be predicted.

4. COMPARATIVE EVALUATION AND ANALYSIS

In the following subsections; we describe the tested datasets, evaluation measures, experimental setup, and comparative analysis of experiment results.

4.1. Datasets

For doing the comparison of ENSDS and existing algorithm (Learn^{++}.NSE) we are using different datasets with different batch sizes. The proposed algorithm is tested over synthetic dataset as well as real time datasets.

1. SEA: The SEA dataset [16] with 50000 examples is synthetically generated which consist of two classes and three features. In which only two features are relevant, and the third being noise.
2. IBM_EOD: The IBM_EOD dataset contains stock data of IBM Company where we are considering open, high, low, close, volume and rate of change in closing price to find out the stock index movement (Up, Down) for classification task. For training purpose data from period 2-Jan-2000 to 16-Feb-2016 (3999 examples) is fetched and for testing purpose data from period 2-Jan-2001 to 16-Feb-2016 (3802 examples) is fetched using Google finance.

The purpose of considering stock data is as we know that stock market data is high frequency data which is complex, non-stationary, chaotic and non-linear and suites our research topic .Concept drift can occurs in the stock market for a number of reasons for example traders preference for stocks change over time, increases in a stock's value may be followed by decreases.

4.2. Evaluation Measures

In addition of accuracy, other evaluation metrics [17] are also used for evaluation purpose. These metrics are defined as:

1. Precision: It is a measure of exactness as given in Eq. (4), which states how many positive examples are actually labelled correctly out of total positive prediction.

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

2. Recall: It is a measure of completeness as given in Eq. (5), which states how many positive examples are actually labelled correctly.

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

3. F-measure: It is used to evaluate the balance between Recall and Precision as given in Eq. (6). Here β is a coefficient to adjust the relative importance of precision versus recall can vary from 0 to 1

$$F - Measure = \frac{(1+\beta)^2.Recall.Precision}{\beta^2.Recall.Precision} \tag{6}$$

4. G-Mean: It is used to evaluate the degree of inductive bias in terms of a ratio of positive accuracy and negative accuracy as given in Eq. (7). It is used to measure the balanced performance of a model between the classes.

$$G - mean = \sqrt{\frac{TP}{TP+FN} \times \frac{TN}{TN+FP}} \tag{7}$$

4.3. Experimental Setup

For experiment analysis, all tested algorithms are implemented in Java using MOA and WEKA framework. The experiments were conducted on a machine equipped with Processor Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz, 2 Core(s), 4 Logical Processor(s) and 4 GB of RAM. Here we have used different batch size for comparison purpose. However, the optimal batch size is different for each stream.

Figure 3 shows the performance improvement of ENSDS over. Learn^{++}.NSE to classify SEA dataset where we are considering Naïve Bayes as base classifiers, different batch size and no pruning strategy is used.Learn^{++}.NSE gives accuracy approximate 88% while ENSDS gives accuracy upto 95% on SEA dataset which comprises sudden drifts. The Precision, Accuracy, F-measure and G-mean of ENSDS is high as fore each batch size as compare to Learn^{++}.NSE. There is continuous improvement in precision and accuracy as with slightly decrease in recall of ENSDS.

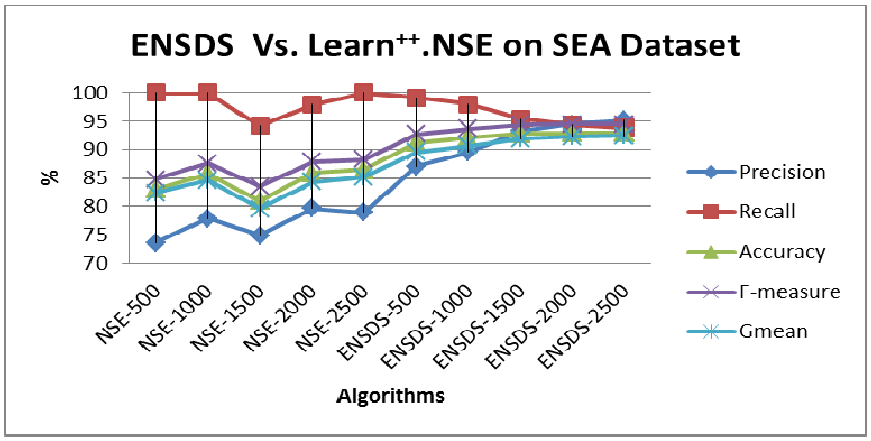

Figure 3. Comparative analysis of Learn^{++}.NSE and ENSDS over SEA dataset

After analysis of ENSDS and Learn++NSE on SEA dataset, we can conclude that 2500 is optimal batch size for ENSDS algorithm as we are obtaining maximum values for all tested evaluation measures.

Figure 4 depicts the performance of Learn^{++}.NSE and ENSDS respectively to classify the stock index movement over IBM_EOD dataset where we are considering Naïve Bayes as base classifiers, different batch size and no pruning strategy is used. Here the Accuracy, Recall of ENSDS is continous better as compare to Learn^{++}.NSE and Precision shows fluctuation for both Learn^{++}.NSE as well As for ENSDS.

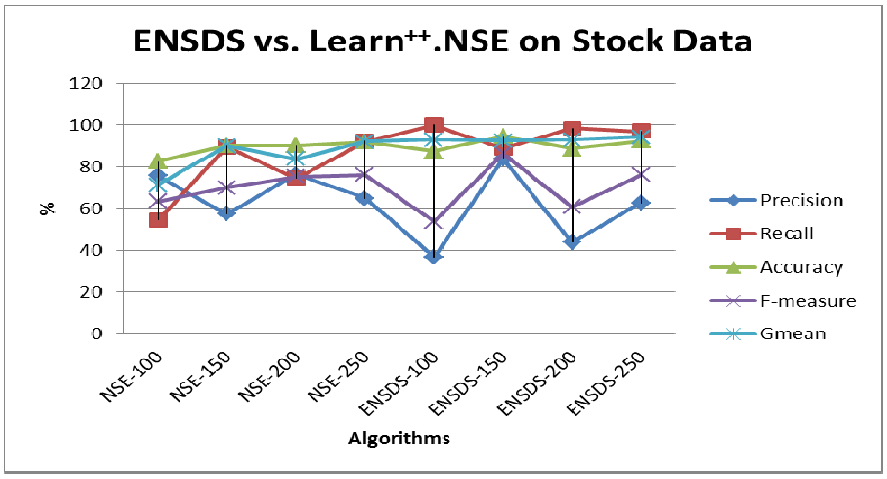

Figure 4. Performance Analysis of Learn^{++}.NSE and ENSDS over Stock dataset

After analysis of ENSDS and Learn++NSE on Stock dataset, we can conclude that 150 is optimal batch size for ENSDS algorithm.

5. CONCLUSION

From the implementation and analysis of ENSDS we can conclude that the performance of ENSDS on different evaluation measures is better as compare to Learn^{++}.NSE. The selection of optimal batch size is varies from dataset to datasets. For SEA dataset the optimal batch size is 2500 for ENSDS algorithm and for Stock dataset the optimal batch size is 150 for ENSDS algorithm. The state of art Learn^{++}.NSE algorithm performs well in case of sudden and gradual drifts but performance can still improved using ENSDS algorithm as time complexity is less. This paper does not emphasis on drift detection mechanism as we

believe in real time environment drift occurs very frequencty so drift detection mechanism can be added to proposed algorithm as future work.

REFERENCES

[1] Y. Wang and Q. Li, "Review on the Studies and Advances of Machine Learning Approaches," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol/issue: 12(2), pp. 1487-1494, 2014.

[2] J. M. Torres, *et al.*, "Unifying view on dataset shift in classification," *Pattern Recognition,* vol. 45, pp. 521–530, 2011.

[3] M. A. Thalor and S. T. Patil, "Review of Ensemble Based Classification Algorithms for Nonstationary and Imbalanced Data," *IOSR Journal of Computer Engineering*, vol. 16, pp. 103-107, 2014.

[4] R. Polikar, "Ensemble Based Systems in Decision Making," *IEEE Circuits and Systems Magazine*, vol/issue: 6(3), pp. 21-45, 2006.

[5] J. Read, *et al.*, "Batch-incremental versus instance-incremental learning in dynamic and evolving data," *IDA,* pp. 313-323, 2012.

[6] F. Han, *et al.*, "A New Incremental Support Vector Machine Algorithm," *TELKOMNIKA Indonesian Journal of Electrical Engineering,* vol/issue: 10(6), pp. 1171-1178, 2012.

[7] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," *Intellegent Conference on Knowledge Discovery & Data Mining,* pp. 377-382, 2001.

[8] H. Wang, *et al.*, "Mining concept-drifting data streams using ensemble classifiers," *Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min,* pp. 226–235, 2003.

[9] D. Brzezinski and J. Stefanowski, "Accuracy Updated Ensemble for Data Streams with Concept Drift," *Proceedings of the 6th international conference on Hybrid artificial intelligent systems,* pp. 155-163, 2011.

[10] R. Elwell and R. Polikar, "Incremental Learning of Concept Drift in Non-stationary Environments," *IEEE Trans. on Neural Networks*, vol. 22, pp. 1517-1531, 2011.

[11] R. Elwell and R. Polikar, "Incremental Learning of Variable Rate Concept Drift," *International Workshop on Multiple Classifier Systems (MCS 2009) in Lecture Notes in Computer Science*, vol. 5519, pp. 142-151, 2009.

[12] M. Karnick, *et al.*, "Learning concept drift in nonstationary environments using an ensemble of classifiers based approach," *International Joint Conerence.on Neural Network*, pp. 3455-3462, 2008.

[13] M. Muhlbaier and R. Polikar, "An Ensemble Approach for Incremental Learning in Nonstationary Environments," *Multiple Classifier Systems,* pp. 490-500, 2007.

[14] M. D. Muhlbaier and R Polikar, "Multiple Classifiers Based Incremental Learning Algorithm For Learning In Nonstationary Environments," *Proceedings of the Sixth International Conference on Machine Learning and Cybernetics,* vol. 6, pp. 3618–3623, 2007.

[15] D. Brzezinski and J. Stefanowski, "Reacting to Different Types of Concept Drift: The Accuracy Updated Ensemble Algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol/issue: 25(1), pp. 81-94, 2014.

[16] http://users.rowan.edu/~polikar/research/nse/ for SEA Dataset.

[17] J. Davis and M. Goadrich, "The Relationship between Precision-Recall and ROC Curves," in *Proceedings of the 23rd international conference on Machine learning,* pp. 233-240, 2006.