Incremental Info-Fuzzy Algorithm for Real Time Data Mining of Non-Stationary Data Streams

Lior Cohen Gil Avrahami Mark Last

Ben-Gurion University of the Negev
Department of Information Systems Engineering
Beer-Sheva 84105, Israel
Email:{clior,gilav,mlast}@bgu.ac.il

Abstract

Most real-world data streams are generated by non-stationary processes that may change drastically over time. In our previous work, we have presented a real-time data mining algorithm called OLIN (On-Line Information Network), which adapts itself automatically to the rate of concept drift in a non-stationary data stream by repeatedly constructing a new model from a sliding window of latest examples. In this paper, we introduce an incremental version of the OLIN algorithm, which saves a significant amount of computational effort by updating an existing model as long as no concept drift is detected. The approach is evaluated on large real-world streams of traffic and stock data.

Keywords

Real-time data mining, incremental learning, online learning, concept drift, info-fuzzy networks.

1. Introduction

In the last 20 years, there has been a huge increase in the amount of information and data, which are kept in databases and computer files. The data to be stored is doubling itself every year. Study made by the UC Berkeley's School of Information Management and Systems in 2003 about the storage and flows analyzes of data [3], indicates the following findings:

- Information flow via electronic channels - telephone, radio, TV, and the Internet (The World Wide Web contains about 170 terabytes of information), contained almost 18 exabytes of new information. Three and a half times more than is recorded in storage media.
- Each year, 800 MB of recorded information is produced per person.
- Ninety-two percent of new information is stored on magnetic media, primarily hard disks.

According to another study [12], transaction-processing workload climbed from an average of 2,094 transactions per second (tps) in 2001 to 3,223 tps two years later, an increase of 54 percent.

The modern information technology produces every year more powerful computers, which enable us to collect, store, transfer, and combine huge amounts of data at very low cost. The progress of the digital data acquisition and technology of storage made the growth of the huge databases possible. This progress in database capability has influenced many areas in human's life, from supermarket transaction data and credit card usage records to molecular and medical databases.

The growth in data has also increased the difficulties and challenges of extracting valid and potentially useful information from these databases, which is the main task of data mining and Knowledge Discovery in Databases (KDD). The main difficulty in mining non-stationary data streams is to cope with the changing data concept. , The fundamental processes generating most real-time data streams may change over years, months and even seconds, at times drastically. This change, also known as concept drift [1], causes the data-mining model generated from past data, to become less accurate in the classification of new data. Algorithms and methods, which extract patterns from continuous data streams, are known as online learning [4]. Real-time data mining of high-speed data streams has large potential in fields such as monitoring manufacturing processes, prediction of stock prices, and intrusion detection in computer networks.

In this paper, we propose a new, incremental approach to mining non-stationary data streams. The main idea of the incremental approach is to increase the average classification rate (in records per second) of real-time data mining systems by reducing the number of times a completely new model is generated. The proposed approach is evaluated on an incremental version of the On-Line Information Network (OLIN) algorithm presented by Last in [5]. OLIN is based on the batch Info-Fuzzy Network (IFN) algorithm developed by Last & Maimon [6] [7]. The proposed incremental algorithm keeps updating an

existing model as long as no concept drift is detected in the arriving data. While most of the existing online algorithms, which deal with the problem of concept drift, build a new model from every new window of training examples, our incremental approach saves the expensive CPU time by performing minimal changes in the current structure of the classification model.

2. Related Work

When dealing with non-stationary data streams, the optimal situation is to have KDD systems that operate continuously, constantly processing the data received so that potentially valuable information is never lost. In order to achieve this goal, several methods for extracting patterns from non-stationary streams of data have been developed, all under the general title of online (incremental) learning methods.

The pure incremental learning methods take into account every new instance that arrives. These algorithms may be irrelevant when dealing with high-speed data streams. Widmer & Kubat [2] have described a series of purely incremental learning algorithms that flexibly react to concept drift and can take advantage of situations where context repeats itself. The series of algorithms is based on a framework called FLORA. FLORA maintains a dynamically adjustable window during the learning process and whenever a concept drift seems to occur (a drop in predictive accuracy) the window shrinks (forgets old instances), and when the concept seems to be stable the window is kept fixed. Otherwise, the window keeps growing until the concept seems to be stable. FLORA is a computationally expensive methodology, since it updates the classification model with *every* example added to or removed from the training window.

Domingos & Hulten [8] have proposed the VFDT (Very Fast Decision Trees learner) system in order to overcome the longer training time issue of the pure incremental algorithms. The VFDT system is based on a decision tree learning method, which builds the trees based on sub-sampling of a stationary data stream. To deal with changing data streams, Domingos & Hulten [9] have proposed an improvement to the VFDT algorithm which is the CVFDT (Concept-adapting Very Fast Decision Tree learner). CVFDT applies the VFDT algorithm to a sliding window of a fixed size and builds the model in an incremental manner instead of building it from scratch whenever a new set of examples arrives. CVFDT increases the computational overload vs. VFDT by growing alternate sub-trees at its internal nodes. The model is modified when the alternate becomes more accurate than the original.

Last in [5] describes an online classification system that uses an info-fuzzy network (IFN). The system called OLIN (On Line Information Network) gets a continuous stream of non-stationary data and builds a network based on the latest examples (sliding window). OLIN detects a concept drift (an unexpected rise in the classification error rate) and dynamically adjusts the size of the training window and accordingly, the rate of the model reconstruction. The calculations of the window size in OLIN are based on the information theory and statistics. The experimental results of [5] show that in non-stationary data streams, dynamic windowing generates more accurate models than the static (fixed size) windowing approach used by CVFDT.

3. The IFN Algorithm

Many learning methods use information theory to induce classification rules. One of the methods, developed by Last & Maimon [6] [7] is the IFN algorithm. IFN, or Info-Fuzzy Network, is an oblivious tree-like classification model, which is designed to minimize the total number of predicting attributes. The underlying principle of the IFN method is to construct a multi-layered network in order to test the Mutual Information (MI) between the input and output attributes. Each hidden layer is related to a specific input attribute and represents the interaction between this input attribute and the other ones. The IFN algorithm is using the pre-pruning strategy: a node is split if this procedure brings about a statistically significant decrease in the entropy value (or increase in the mutual information) of the target attribute. If none of the remaining input attributes provides a statistically significant increase in mutual information, the network construction stops. The output of this algorithm is a network, which can be used to predict the values of a target attribute similarly to the prediction technique used in decision trees.

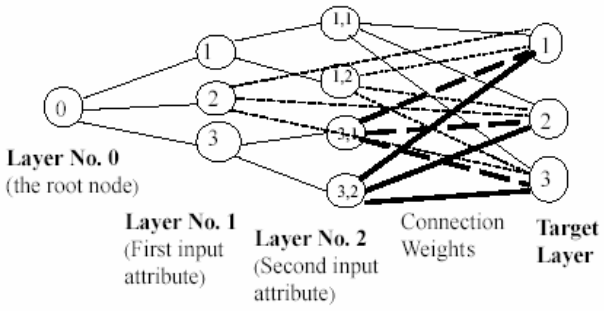

Figure 1. Info-fuzzy Network Two Layered Structure [7]

Fig. 1 illustrates a sample structure of an info-fuzzy network. In this example, the network contains two layers, which represent two input attributes. The first input attribute has three values, represented by nodes no. 1, 2, and 3 in the first layer. Nodes no. 1 and 3 are split by the network construction procedure. The second layer has four nodes, which are the combinations of two values of the second input attribute with two split nodes of the first layer.

The target layer represents the target attribute, which gets three values.

4. Incremental On-Line Information Network

4.1. Algorithm Overview

The online algorithm of Last [5] deals with the concept drift problem simply by generating a new model for every new sliding window. On one hand, this regenerative approach ensures accurate and relevant models over time and therefore an increase in the classification accuracy. On the other hand, the OLIN algorithm has a major drawback, which is the high cost of generating new models. The regenerative on-line IFN does not take into account the costs involved in replacing the existing model with a new one.

In this paper, we present a new algorithm, called Incremental On-Line Information Network (Incremental OLIN), which is an extension, to the regenerative OLIN algorithm of Last [5]. As shown in the evaluation section below, the incremental algorithm achieves almost the same and sometimes even higher accuracy rates than the regenerative algorithm and it is significantly cheaper since it does not require producing a new model for every new window of examples. As long as no concept drift is detected, the Incremental OLIN algorithm is applied repeatedly to a sliding window in order to update the current model rather than replace it completely.

The basic intuition behind the incremental approach is to update the current classification model with the current training window concept as long as no major concept drift has been detected and to build a new model in case of a major concept drift. From applying the regenerative OLIN [5] to several data sets (in transportation, manufacturing, and stock market domains) a few phenomena have been observed. The first one: when there is no concept drift between two adjacent training windows, the differences between the new constructed model and the former one are minor (about 80% of the network structure remains the same). This means that instead of re-constructing a new model (in case of no concept drift), the latest model can be updated in much less time and effort. The second one: when there is no concept drift between two adjacent windows, the main differences between the new model and the former model are in the last hidden layer. The third one: when a concept drift is discovered between two adjacent windows, the new model is almost totally different from the former one (in 90% of the cases the differences propagate up to the root node). The above phenomena indicate that in updating an existing model, it is sufficient to update its last layer and that if a concept drift has been detected, it is preferable to construct a new model.

Several operations for updating the model can be applied. One of them is to check the split validity on each node starting from the root and downwards the network. This check is made in order to ensure that the current split of a specific node actually contributes to the mutual information calculated from the current training set. The elimination of non-relevant splits should decrease the error rate. Another operation is to check which attribute is more appropriate to correspond to the last (terminal) layer: the attribute that is already associated with the last layer or the second best attribute for the last layer of the previous model. The attribute selected for the last layer of the new model will be the one with the greater conditional mutual information based on the current training window. The last operation is attempting to split the nodes of the last layer on attributes, which are not yet participating in the updated network.

4.2. Detailed Description

Following is the pseudo-code outline of the Incremental OLIN algorithm.

The ***IFN_Control*** procedure is responsible for managing the application. It gets as input a continuous stream of examples (*S*). The initial size of the training window W_{init} is calculated by using an equation developed by Last in [5]. Afterwards, the initial IFN model is produced by applying the IN algorithm to the initial window of examples.

IFN_Control (S)
Calculate the initial size of the training window W_{init}
Set $W = W_{init}$
Obtain IFN model by applying the IN algorithm to the sliding window (*W*)
While *S* is not finished
    IFN model = Incremental_IFN (W, IFN model)
    Return *IFN model*

The algorithm will run till the end of the data stream. If the data stream is infinite, the algorithm will keep running and producing IFN models for classification.

The ***Incremental_IFN*** procedure is responsible for calculating both error rates of the training and validation examples after running the current model on those data sets. In addition, the maximum expected difference between those errors *Max_Diff* is calculated at the 99% confidence level using a Normal Approximation to the Binominal distribution (as shown in [5]). It is important to mention that we assume immediate availability of correct classifications for the window of validation examples. This is a reasonable assumption in stock prediction, traffic control, web usage mining, and other real-time data mining domains [2], [5].

A stable concept is observed if the actual difference between the validation and training errors is smaller than the maximum expected difference *Max_Diff*. In this case,

operations for updating the network are applied. If a concept drift occurs, we are forced to create a new network.

Incremental_IFN (W, IFN model)
Calculate the training error rate E_{tr} of *IFN model*
Calculate the validation error rate E_{val} of *IFN model*
Find the maximum expected difference between the last training and the validation errors *Max_Diff*
If (E_{val} - E_{tr}) < *Max_Diff* //concept is stable
    Update_Current_Network (IFN_Model, W)
Else
    Obtain new IFN model by applying the IN algorithm to the sliding window (*W*)
Calculate *New_Training_Window_Size* (*W*) [5]
Return updated *IFN_Model*

The ***Update_Current_Network*** procedure gets as inputs the current network structure and a sliding window. This procedure activates another procedure (*Check_Split_Validity)* for checking the split validity of the current network. Afterwards, it replaces the last layer of the network if needed. Finally, it activates the *New_Split_Process* procedure performing a new split process on the last layer (whether it was replaced or not).

Update_Current_Network (IFN_Model, W)
Check_Split_Validity (IFN_Model, W) on the last layer of the current model
Calculate the conditional MI of *Sec_Best_Attr* based on the current training set (*W*)
IF (conditional MI of the current last layer < conditional MI of *Sec_Best_Attr*)
    Replace last layer with *Sec_Best_Attr*
    New_Split_Process (IFN) on the last layer of the current model

---

Check_Split_Validity is responsible to check if the current split of each node actually contributes to the conditional mutual information calculated from the current training set.

Check_Split_Validity (IFN_Model, W)
For *i = total_number_of_layers-1* to *i* = 1
    For *j* = 1 to *j = number of nodes in hidden layer i*
        If node *j* is split
            Calculate the estimated conditional MI of *j* and the target attribute [6]
            Calculate the Likelihood-ratio statistic of *j* [6]
            If the Likelihood-ratio statistic of *j* is significant
                Leave the node split
            Else
                Remove the splitting and make *j* a terminal node

---

New_Split_Process is responsible for splitting the nodes of the last layer on attributes, which are not yet included in the updated network.

New_Split_Process (IFN)
Repeat for every candidate input attribute *i'* which is still not an input attribute
    Repeat for every node *z* of the final hidden layer
        Calculate the estimated conditional MI of *i*' and the target attribute given *z*
        Calculate the Likelihood-ratio statistic of *i*' and the target attribute given *z*
        If the Likelihood-ratio statistic of *i*' is significant
            Split *z* on *i*' and increment the conditional MI of the candidate input attribute *i*' and the target

---

The time complexity of the suggested incremental algorithm depends on the number of detected concept drifts. As long as no concept drift has been detected, the major computational cost is to add a new layer to the existing network. Checking the split validity of the nodes in the network is relatively cheap (O(*n*) where *n* is the number of nodes in the network). In case of concept drift, a new network should be constructed from scratch, and the cost of the network construction procedure is linear in the number of records, linear in the number of distinct attribute values, and quadratic in the number of candidate input attributes [6].

5. Evaluation

The Regenerative OLIN reconstructs a new model with every new sliding window of training examples. According to the results presented [5], the classification models produced by this algorithm are relatively accurate but the total processing time is very long due to high frequency of constructing a new model. The Incremental OLIN algorithm presented in this paper is aimed at decreasing the processing time per each new example. The Incremental OLIN was evaluated in comparison to the original Regenerative OLIN on several real-world data streams.

5.1. Traffic Data

The first set of data streams includes traffic flow information on a signaled three-way intersection in Jerusalem. The traffic data is recorded by lane sensors. The vehicles can cross the intersection in five different directions. The five data streams obtained for directions 1 to 5 included the hourly incoming traffic volumes for 24 hours a day, seven days a week during a period of more than 3 years.

Each original data stream has been converted into a data set, where a record contains twelve candidate attributes representing the exact time (date, hour, day in week, etc.)

when the traffic volume was measured and traffic volumes at earlier points of time (the previous hour, the same hour of the previous day, etc.). The target attribute represented the volume of traffic during a given hour. Due to the fact that the target attribute is continuous we have manually discretized it to three intervals of high, medium, and low traffic volume. The traffic data was divided into five separate data tables for the five directions. Each table corresponding to a given direction contained about 30,000 hourly records.

5.2. Stock Data

The second set we used was a stock market data. This data set was also used in [5] for the evaluation of the Regenerative OLIN. The raw data represents the daily stock prices of 373 companies from the Standard & Poor's index [10], over a 5-year period (from 8/29/94 to 8/27/99). The data was obtained from the Microsoft™MoneyCentral web site [11]. An average of 15.64 intervals (with distinct trends) per company have been identified and the classification problem has been defined as predicting the correct length of the current interval based on the known characteristics of the current and the preceding intervals. The data table contains 5,462 records with six candidate attributes, which include the duration, the slope and the fluctuation measured in each interval as well as the sector of the corresponding stock. The target attribute which is the duration of the second interval in an interval-pair, has been discretized to five intervals of nearly equal frequency.

5.3. Initial Results

The runs of both algorithms were carried out on a Pentium IV processor with 256 MB of RAM. In the experiments, the online learning on the traffic data starts after inducing the initial model from the first 500 records, which leaves the system to work with about 30,000 records for each direction. For the stock data, the online learning starts after the first 462 records, which are used for inducing the initial model. Tables 1 and 2 show the results after applying the regenerative and the Incremental OLIN to the traffic data sets and the stock data set respectively.

From the results on the traffic data, it can be seen that the Incremental OLIN has reduced the run time by at least 20% and at most by 87% (the average run time saving was 75%). At the same time, the accuracy rate decreased by 4% in the worst case while the average accuracy loss was only 2%. One can also see that the incremental algorithm has increased the classification rate from 55 to 139 records per second. In the case of the stock data, the incremental version reduced the run time by 72.25% while increasing the error rate by 1.3% only. In addition, the classification rate increased from 47.97 to 172.85 records per second, when we used the incremental version.

Table 1. Summary of Experiments Using the Regenerative OLIN

	Total Number of Records	Run Time (sec.)	Average Classification Rate (records/second)	Error Rate	Number of Observed Concept Drifts
Direction1	28,761	487.68	58.975	0.107	55
Direction2	30,480	466.38	65.354	0.215	55
Direction3	30,480	1354.42	22.5	0.103	17
Direction4	30,480	1829.57	16.66	0.114	7
Direction5	30,480	271.96	112.075	0.111	51
Traffic Average	30,136	882	55.11	0.13	37
Stock Market	5462	113.87	47.97	0.415	8

Table 2. Summary of Experiments Using the Incremental OLIN

	Total Number of Records	Run Time (sec.)	Average Classification Rate (records/second)	Error Rate	Number of Observed Concept Drifts
Direction1	28,761	222.3	129.38	0.122	79
Direction2	30,480	211.69	143.98	0.256	119
Direction3	30,480	192.23	158.56	0.103	20
Direction4	30,480	242.12	125.89	0.145	25
Direction5	30,480	217.23	140.31	0.119	65
Traffic Average	30,136	217.11	139.62	0.149	61.6
Stock Market	5462	31.6	172.85	0.428	11

6. Conclusions and Future Work

This paper has presented a new, incremental approach to real-time classification of continuous non-stationary data streams. The incremental approach applies the classification algorithm repeatedly to a sliding window of examples, in order to update the existing model or to construct a new model. A concept drift is detected by an unexpected rise in the classification error rate. When the concept appears to be stable, the system updates the current classification model by applying several operations to it. If a concept drift has been detected, the system re-generates a new model.

The proposed incremental approach was evaluated using an incremental version of the On-Line Information Network (OLIN) algorithm, which constructs an oblivious tree-like classification model. The efficiency of the Incremental OLIN has been confirmed by experiments on real-world sets of online data. The data in use included a set of five traffic data streams, which contained about 30,000 records for each traffic direction and a set of stock data, which contained 5,462 records. It is clear that the incremental version of OLIN outperforms the Regenerative OLIN in terms of processing rate. Another encouraging finding was the low decrease in accuracy of the incremental version compared to the regenerative one.

The future work includes implementation of additional incremental classifiers and evaluating them on more real-world datasets, as well as on artificially built data streams. Finding a better trade-off between the accuracy rate and the processing time can also be examined.

Acknowledgments. We would like to thank the Traffic Control Center of Jerusalem for granting us the permission to use their traffic database. This work was partially supported under a research contract from the Israel Ministry of Defense.

7. References

[1] D.P Helmbold and P.M. Long, Tracking Drifting Concepts by Minimizing Disagreements, Machine Learning, No. 14, pp. 27-45, 1994.

[2] G. Widmer and M. Kubat, “Learning in the Presence of Concept Drift and Hidden Contexts”, Machine Learning, Vol. 23, No. 1, pp. 69-101, 1996.

[3] P. Lyman and H. R. Varian, "How Much Information", 2003. Retrieved from http://www.sims.berkeley.edu/how-much-info-2003.

[4] M. Black and R. J. Hickey, “Maintaining the Performance of a Learned Classifier under Concept Drift”, Intelligent Data Analysis, No. 3, pp. 453-474, 1999.

[5] M. Last, “Online Classification of Nonstationary Data Streams”, Intelligent Data Analysis, Vol. 6, No. 2, pp. 129-147, 2002.

[6] M. Last and O. Maimon, “A Compact and Accurate Model for Classification”, IEEE Transactions on Knowledge and Data Engineering, Vol. 16, No. 2, pp. 203-215, February 2004.

[7] O. Maimon and M. Last, Knowledge Discovery and Data Mining - The Info-Fuzzy Network (IFN) Methodology, Kluwer Academic Publishers, December 2000.

[8] P. Domingos and G. Hulten, “Mining High-Speed Data Streams”, Proc. of KDD 2000, pp. 71-80, 2000.

[9] P. Domingos and G. Hulten, “Mining Time-Changing Data Streams”, Proc. of KDD 2001, pp. 97-106, ACM Press, 2001.

[10] Standard & Poor’s Index at http://www.spglobal.com.

[11] The Microsoft™MoneyCentral home page at http://windowsmedia.com/mediaguide/gbhome.

[12] R. Winter and K. Auerbach, “Contents Under Pressure”, Intelligent Enterprise, May 2004, available at http://www.intelligententerprise.com/showArticle.jhtml?articleID=18902161