CLASSIFICATION OF EVOLVING STREAM DATA USING IMPROVED ENSEMBLE CLASSIFIER

Seema S.

Associate Professor, Department of Computer Engineering, MSRIT, Bangalore
seemashedole@yahoo.co.in

ABSTRACT

Data mining is a user-centric process that is used to extract useful patterns from large volumes of data. With the growth of the Internet, a data stream is today a key area of advanced analysis and data mining. Handling data streams is a difficult task due to the variations in the data and the frequent occurrences of concept drifts. No single classifier can be relied upon to correctly classify data stream data since they are developed through a specific learning approach. Hence we use a multi-chunk ensemble of classifiers to classify evolving data streams and improve the prediction accuracy over single classifiers. We evaluate our ensemble on synthetic as well as real time data, compute the precision and represent it graphically using both majority voting as well as new proposed weighted averaging and compare its performance against individual classifiers. Current techniques include a single chunk approach, where the entire set of data is considered as a whole, and the used method shows better efficiency.

KEYWORDS

Ensemble, data stream, multi-chunk, weighted averaging.

1. INTRODUCTION

Data mining is the process of extracting useful patterns and information from voluminous data that cannot be obtained through data querying .The patterns that are found can be combined in order to make business decisions. It is used for forecasting future events, associating related events, and sequencing the order of event occurrences. Data mining is broadly categorized into two types, viz. predictive data miming and Descriptive Data mining. Predictive data meaning is a supervised learning approach in which a set of training instances is used to predict the class to which a test instance belongs. Descriptive data mining or Cluster Analysis is an unsupervised learning technique that is used to group together data instances exhibiting similar traits into clusters.

Data stream is an uninterrupted flow of a long sequence of data, which is generated continuously at a rapid rate. If collected over a period of time, it is impractical to store such volume of data. If stored using traditional methods, accessing a particular data instance is expensive in terms of both space as well as time. Hence appropriate models to develop a summary of the data seen so far need to be developed. Data Streams possess high dimensionality due to the large number of attributes and therefore require multidimensional processing.

It is observed that there is a sudden unformulated change in the target variable which is to be predicted when dealing with data streams. This is referred to as a concept drift. Such concept drifts are highly frequent in the case of data streams. Processing such evolving data streams poses a challenge to the user community. A single classifier is built using a specific learning approach with a single direction of hypothesis. This is clearly insufficient to deal with variations in the data

DOI : 10.5121/ijdkp.2012.2401

[1]. Hence, we adopt a better mining approach that combines multiple classifiers built using different learning approaches into an ensemble [5].

Ensemble methods are machine learning algorithms that combine the predictions of multiple classifiers[1]. The accuracy of ensemble methods is greater than the individual classifiers since the disadvantage of the individual classifiers that are built on specific learning approaches, are eliminated. Ensemble methods are better because combining multiple hypotheses and looking at the problem from different perspectives gives a holistic view of the problem[1]. Two popular ensemble methods are bagging and boosting [2].

Related work includes single chunk method and multi-partition multi-chunk method. In single chunk method there is no track of historical data. Hence such techniques cannot handle changes in data streams where new data contains classes from old data as well as some new classes; hence keeping track of historical data is essential. There also exists a method where a window consisting of many chunks is partitioned and fed to individual classifiers. But in such cases some classifiers will perform well every time, whereas some will underperform each time. We want equal contribution from each base classifier, hence we adopt the Sliding window multi-chunk approach, which improves over both methods, and can be used for long running data sets too.

2. CONSTITUENT CLASSIFIER SELECTION

Predictive data mining deals with many response variables. Knowledge models of predictive data mining suggest the use of three broad learning approaches, namely, Rule-Based learning, Bayesian Learning and Instance-Based Learning.

Rule Based learning is the technique of constructing rules in order to classify data instances into a predefined set of classes. The rule-based learning algorithm that we have used in our ensemble is Decision Trees. Decision tree is a flowchart-like tree structure that is built top-down. The internal nodes denote the decisions, i.e conditions on attributes, the branches denote the outcomes of these decisions and the leaf nodes represent the classes that have been predicted. The reason behind the selection of this particular rule-based learning algorithm is its good prediction accuracy and interpretability of results. Also since the testing process is a mere traversal of the constructed tree, it is a fast process.

Bayesian learning is a method of statistical inference in which some kinds of evidence or observations are used to calculate the probability that a hypothesis may be true, or else to update its previously calculated probability. Naïve Bayes is the Bayesian learning algorithm that we have employed. It is a probabilistic theorem built on the Bayes theorem of conditional probability. Since data streams are dynamic, the dependencies among the attributes changes continuously with time, hence Naïve Bayes is an appropriate learning approach to deal with this situation since the presence(or absence) of a feature is independent of the presence(or absence) of any other feature in the feature set. It is simple owing to it being a formula-based approach. Despite its simplicity, it is accurate and outperforms many complex algorithms.

Instance Based learning employs the usage of historical information to predict labels of instances. k-Nearest Neighbours (k-NN), classifies the test instances based on their proximity to the training example in the feature space. We use k-NN because it exhibits a highly adaptive spatial behaviour and lends itself easily to parallel implementations.

3. DETAILED DESCRIPTION

3.1. Decision Trees

A decision tree is constructed by selecting an attribute for each node of the tree from an attribute list. The internal nodes denote conditions on selected attributes and the leaf nodes denote the predicted classes. Attributes at each level are selected based on their information gain. The attribute used at every level is discarded after the child nodes have been formed. This process

continues until either the attribute list is exhausted, or terminating conditions are encountered. The process of constructing a decision tree stops by the assigning the class labels to the leaf nodes [18].

Since the depth of the tree is proportional to the number of dimensions of the dataset, we perform feature selection [8] to obtain the best ten attributes that provide the maximum information gain.

3.2. Naive Bayes

The implementation begins with determining the prior probability of each class. The prior probability of a class is a fraction of the number of instances belonging to that class to the total number of instances present in the data set. Further the mean and the variance of every class are determined, for every test instance of the data stream that enters the system the probability of that instance belonging to each class present is computed. This is called the posterior probability of the class which has a formula encapsulating the variance and prior probability calculated above. The class which yields the highest posterior probability is predicted as the target class for the test instance. There is a possibility of the variance being zero if all classes are not covered in the current data set, hence we perform normalization on variance such that instead of zero, it starts with a value very close to zero (of the order of 10^{-5}) [19].

3.3. K-Nearest Neighbour (k-NN)

There is no training process involved in the implementation of this algorithm. For every test instance involved, the entire training set is scanned and the distance of the test instance with each of the training instances is determined using the L infinity method of computation. On the basis of the distances computed above, k nearest neighbours (training examples) are chosen [11]. The user decides upon the value of k which determines the performance and varies from application to application. The output class of the test instance is determined either by majority voting or weighted voting. In majority voting the class, which occurs maximum number of times among k nearest neighbours is the predicted class. In the weighted method every class is assigned weights inversely proportional to distances of the tuples belonging to this class from the test instance. The class with the highest weight is the target class. We have considered the value of k as 15 because it gave us the best performance [17].

3.4. Ensemble Method

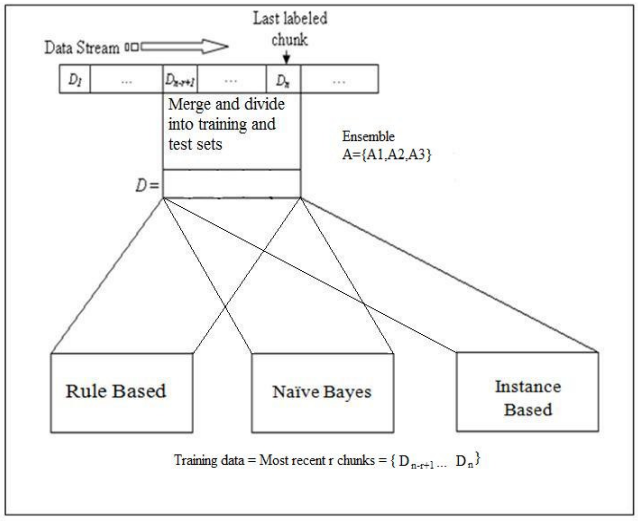

Figure 1: Ensemble of Classifiers

The Ensemble technique we are adopting is a Sliding Window multi-chunk approach proposed by [4]. Most current ensemble techniques apply a single chunk approach, in which the upcoming data chunk is fed as a whole to the ensemble, which means complete replacement of old data with new data. Because of this, no historical information is retained. Hence, if any data comes, which needs training with new data as well as some information about old data; it cannot be provided by the single chunk approach.

To overcome this, we divide the data into many chunks, and feed the most recent r-chunks to the ensemble. When a new chunk arrives, it is fed to the ensemble along with the most recent r-1 chunks before this, so some old data information is retained too. This method helps in handling unpredictable nature of data streams better.

In our method we consider r=4 in each window, in the beginning all four new chunks are present where first three chunks are used for training and the last chunk for testing.

As the window slides, the oldest chunk is removed and the previous test chunk is used for training, and the latest chunk is used as the testing chunk. This process continues as new data keeps coming.

For training the ensemble, the three training chunks are fed to each of the classifiers, viz. Decision Trees, Naïve Bayes and k-NN. Each of these classifiers will train themselves according to the training data and store the results in an appropriate data structure (like a list or an array) for the ensemble to use.

4. ALGORITHM TO BUILD CLASSIFIER ENSEMBLE

Input: *D,* consists of most recently labelled *r* data chunks

1. *A*: Current ensemble consisting of the classifiers.

Output: Updated ensemble *A*

Let *D* = Most recent *r* chunks

1. **for** j=1 to 3 **do**
2. $A^n_j \leftarrow$ Train a classifier with training data *D* -r^{th} *chunk*
3. Test A^n_j on its test data r^{th} *chunk* and compute its expected error
4. **end for**
5. *A* ← The ensemble resulting out of majority and weighed voting of all classifiers.

Methodology:

The implementation uses a Sliding-Window approach. The data is divided into equal sized chunks. The sliding window contains the most recent r chunks (i.e. 4) chunks and slides by one chunk at a time. These consist of the most recent r-1 chunks as well as the newly arriving chunk that is to be classified. Each test instance is fed to all the three classifiers and the output of the ensemble is determined by both the methods mentioned below:

Majority Voting: the output class is the one predicted by a majority of the classifiers, i.e. either two or all three. Weighted Voting: Weights are assigned to the individual classifiers based on their performance (prediction accuracy) and the output class is determined by a formula which is a function of the calculated weights. The performance of the ensemble is represented theoretically as well as graphically.

5. TEST APPROACH

The three classifier ensemble is always kept up to date before it is employed for testing. After the most recent (r-1) chunks are used for training, the r^{th} data chunk will be tested and used for prediction.

Each classifier determines a class for every test instance. The output of the ensemble is thereafter determined by both majority as well as weighted voting.

1. Majority voting:

If any two or all of da three classifiers in the ensemble denote one particular class, the output of the ensemble is none other than that class. Else precision is used to solve the dispute.

2. Weighted averaging(Proposed by us):

For each classifier we have assigned a weight proportional to the square of its precision as follows:

Weight (classifier1) =(precision(classifier1))^2/(precision(classifier2)+precision(classifier3)))

Simple majority voting can be used for testing the ensemble, but if majority of the individual classifiers have a high error rate, so will the ensemble and hence the efficiency of the ensemble will drop. Hence we propose the weighted ensemble method.

The above given formula is proposed by us for the calculation of the weights. The weights are assigned as square of the precisions of the classifiers obtained. This is because there should be a considerable difference in weights if there is a lot of difference in the efficiencies of the individual classifiers. If one classifier outperforms the others by a large margin, then the ensemble decision should be dominated by the decision of that classifier. If all the classifiers give around the same efficiency, weights should not be very different from each other. Assigning weights proportional to square of the precision helps in achieving both these objectives.

For every test instance, the output of the classifier which yields the maximum weight is taken to be the final output of the ensemble.

Thus the prediction precision of each of the individual classifiers is determined on the test data followed by that of the ensemble. Further a tabular statistics as well as graph (precision versus sliding window) displaying the superiority of the ensemble technique over individual classifiers is plotted.

Precision is given by the formula:

$$Precision=tp/(tp+fp)$$

Where tp=true positive, fp=false positive

6. TESTING AND PERFORMANCE EVALUATION

6.1. Synthetic Data

Synthetic data is any production data applicable to a given situation that are not obtained by direct measurement. They are usually generated with drifting concepts. Synthetic data sets have been developed as a way to rate classification algorithms. Synthetic data has two key advantages over real data when the description of algorithms is considered:

1. As FCS files for which the classification of each event is known with absolute certainty.
2. To model specific confounding factors that each

Classification algorithm should be able to handle with varying degrees of success. Matching particular data sets to algorithms that are well-suited to the characteristics of said data should thus improve the quality of the resulting classifications.

Data is divided into multiple chunks each chunk of size 20 instances. Synthetic data is passed chunk by chunk to the individual classifiers as well as ensemble for testing.

The ensemble is always trained with the most recent data before it is employed for testing. When a test chunk arrives, all the three classifier run on the chunk and predict respective classes for each of the individual instances. Finally the ensemble performs its prediction by both majority voting as well as weighted averaging method. The precision of the ensemble as well as individual classifiers are determined each time before the training window slides to accommodate the recently tested data. The measurements are written into an excel file and a line graph is plotted for the same.

Results:

The synthetic data set has relatively few concept drifts. Hence the performance of the majority and weighted ensemble is the same on this data.

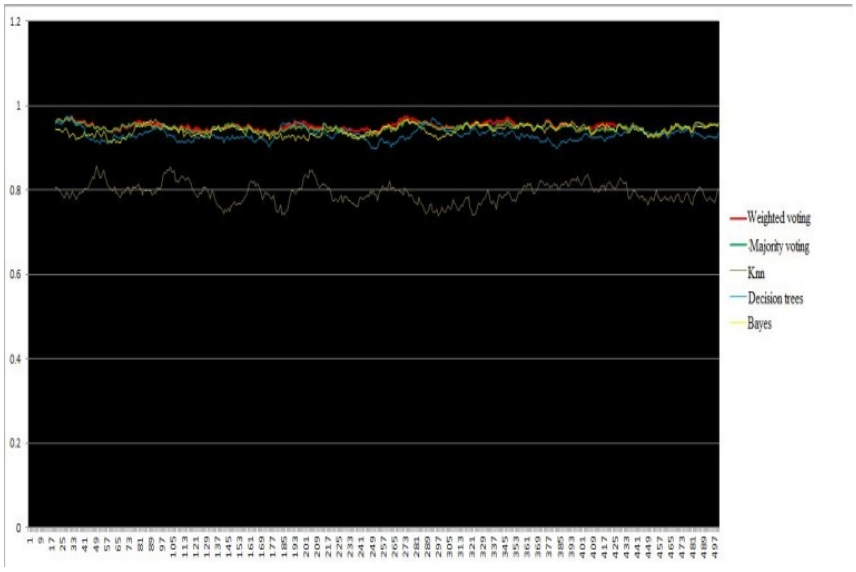

Figure 2. Precision Vs Sliding Window for synthetic data

Table 1. Average precision values for synthetic data over 500 windows

Algorithm	Decision	Bayes	k-NN	Majority	Weighted
Precision (%)	93.2	94.15	79.31	94.8	95.15

6.2. Network Intrusion Detection Stream Dataset

The Network Intrusion Detection data set consists of a series of TCP connection records from two weeks of LAN network traffic managed by MIT Lincoln Labs. Each record can correspond to a normal connection, an intrusion or an attack. This data set evolves rapidly, and is useful in testing the effectiveness of the ensemble approach in situations in which the characteristics of the data set change rapidly over time.

Data is divided into multiple chunks each chunk of size 20 instances. Synthetic data is passed chunk by chunk to the individual classifiers as well as ensemble for testing.

The ensemble is always trained with the most recent data before it is employed for testing. When a test chunk arrives, all the three classifier run on the chunk and predict respective classes for each of the individual instances. Finally the ensemble performs its prediction by both majority voting as well as weighted averaging method. The precision of the ensemble as well as individual classifiers are determined each time before the training window slides to accommodate the

recently tested data. The measurements are written into an excel file and a line graph is plotted for the same.

Results:

The network intrusion data has concept drifts occurring over a long period of time (infrequently).

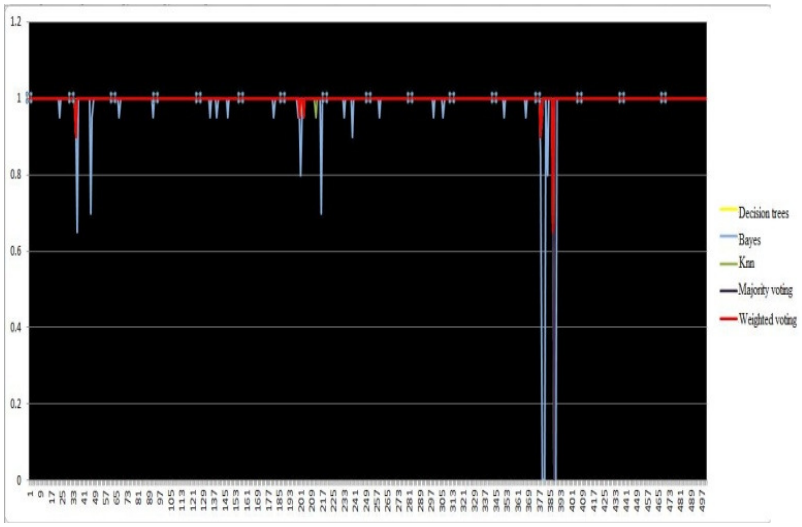

Figure 3. Precision Vs Sliding Window for network intrusion detection

Table 2. Average precision value for network intrusion detection over 500 windows

Algorithm	Decision	Bayes	k-NN	Majority	Weighted
Precision (%)	99.85	98.42	99.66	99.66	99.87

6.3. Forest Cover Data

This data set contains 581012 observations and each observation consists of 54 attributes, including 10 quantitative variables, 4 binary wilderness areas and 40 binary soil types variables. This data set is converted into a data stream by taking the data input order as the order of streaming and assuming that they flow-in with a uniform speed.

Data is divided into multiple chunks each chunk of size 20 instances. Synthetic data is passed chunk by chunk to the individual classifiers as well as ensemble for testing.

The ensemble is always trained with the most recent data before it is employed for testing. When a test chunk arrives, all the three classifier run on the chunk and predict respective classes for each of the individual instances. Finally the ensemble performs its prediction by both majority voting as well as weighted averaging method. The precision of the ensemble as well as individual classifiers are determined each time before the training window slides to accommodate the recently tested data. The measurements are written into an excel file and a line graph is plotted for the same.

Results:

The forest cover data portrays huge concept drifts occurring very frequently .Since none of the three classifiers are designed to deal extremely well with evolving data, majority voting fails to prove its effectiveness .On the other hand, weighted technique prioritizes the classifiers according to their performance. Hence the most efficient classifier outperforms the others and influences the final output thereby yielding better prediction.

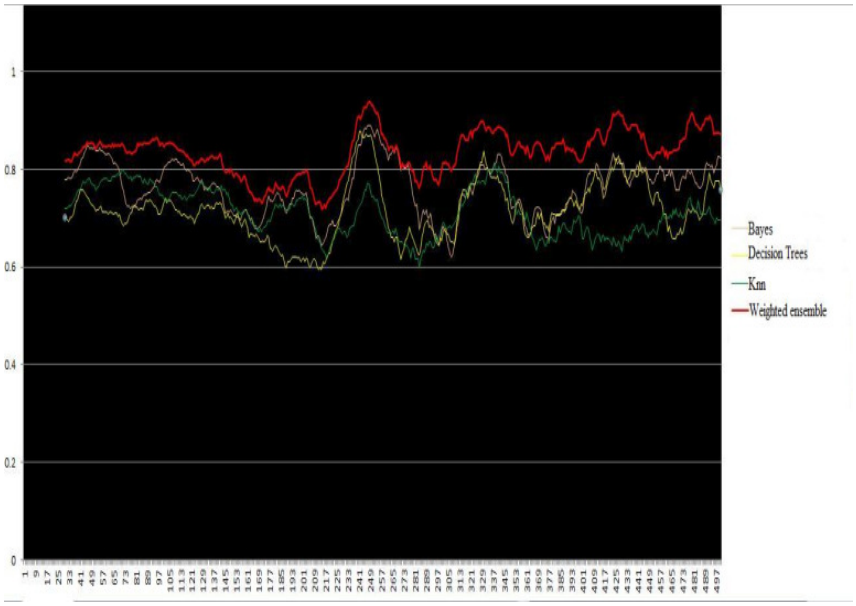

Figure 4. Precision Vs Sliding Window for Forest Cover Data

Table 3. Average precision values for forest cover data over 500 windows are given below:

Algorithm	Decision	Bayes	k-NN	Majority	Weighted
Precision (%)	71.53	76.4	71.08	78.14	83.42

7. CONCLUSION AND FUTURE ENHANCEMENTS

Thus in this project we described a multi chunk ensemble for predictive data stream processing. Even though the individual classifiers perform reasonably well in common instances, one or more of them fail during concept drifts, whereas ensemble manages to give good efficiency in such cases. Our sliding window multi chunk approach handles the concept drifts and keeps the ensemble up to date for future prediction. We can see that the weighted ensemble technique is much more robust and efficient than the individual classifiers as well as majority voting technique. Hence we suggest using an ensemble of classifiers rather than individuals, especially in case of evolving data streams where concept drifts occur frequently. In future work we would like to parallelize the ensemble, such that all the three classifiers run simultaneously reducing the overall time complexity.

REFERENCES

[1] Zhong-Hui W, Wan-Gui Li, Yun-Ze Cai, Xiaoiming Xu, "An empirical comparison of Ensemble Classification algorithms with Support Vector Machine", Third International Conference on Machine Learning and Cybernetics, Shanghai, 25-29 August 2004.

[2] Carlo Zaniolo and Hetal Thakkar, "Mining Data Bases and Data Streams" (2010) Pg 1,2,18, 19, 20.

[3] Data Mining, Introductory and Advanced Topics – Margaret H. Dunham.

[4] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, Ricard Gavaldà "New Ensemble Methods For Evolving Data Streams" – International Conference on Knowledge Discovery and Data Mining, 2009 Pg 1, 2.

[5] Zhi-Hua Zhou "Ensemble Learning" – Proceedings of the 9th IEEE International Conference on Data Mining (ICDM'10), Sydney, Australia, 2010 Pg 1-4.

[6] Albert Bifet - "Adaptive Stream Mining: Pattern Learning And Mining From Evolving Data Streams", Proceedings of the International conference on Adaptive Stream Mining: Pattern Learning and Mining from Evolving Data Streams, 2010

[7] Prof. Navneet Goyal – "Ensemble Classifiers (ppt)".

[8] David Opitz, Richard Maclin "Popular Ensemble Methods: An Empirical Study" - Journal of Artificial Intelligence Research 11, 1999, Pg 1-5.

[9] Richard A. Berk, "An Introduction to Ensemble Methods for Data Analysis", Sociological Methods & Research 2006, Pg 29, 30, 31.

[10] Sharma Chakravarthy, Qing Chun Jiang "Stream data processing: a quality of service perspective : modeling ".-google books ,2009.pg 1-15

[11] Feng Tan "Improving Feature Selection Techniques for Machine Learning", 11-27-2007

[12] Huan Liu, Hiroshi Motoda , Rudy Setiono , Zheng Zhao "Feature Selection: An Ever Evolving Frontier in Data Mining".

[13] Tom M. Mitchell - "Gaussian Naïve Bayes, and Logistic Regression", - Machine Learning 10-701, 2010

[14] H. Wang, W. Fan, P. Yu and J. Han (2003). "Mining Concept-Drifting Data Streams using Ensemble Classifiers" In: Proc. of the 9th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Aug. 2003, Washington DC, USA

[15] J. Gama, R. Rocha and P. Medas (2003). "Accurate Decision Trees for Mining High-Speed Data Streams"' In: Proc. of the Ninth International Conference on Knowledge Discovery and Data Mining, Edited by P.Domingos and C. Faloutsos, ACM Press, 2003.

[16] K-nearest neighbour classifier – www.cs.aau.dk /~tdn /idev /uploads /media /DM-KNN.pdf.

[17] Jiawei Han and Micheline Kamber - Data Mining: Concepts and Techniques, 2nd ed.

[18] J. Gama and P. Medas and P. Rodrigues (2005). Learning Decision Trees from Dynamic Data Streams, In: Proc. of ACM Symposium on Applied Computing - SAC05, 2005.

[19] Naïve Bayes Classifier – Statsoft, "http://www.statsoft.com/textbook/naive-bayes-classifier/".