Classifying Evolving Data Streams Using Dynamic Streaming Random Forests

H. Abdulsalam, D. B. Skillicorn, and P. Martin

School of Computing, Queen's University
Kingston, ON Canada, K7L 3N6
hanady,skill,martin@cs.queensu.ca

Abstract. We consider the problem of data-stream classification, introducing a stream-classification algorithm, *Dynamic Streaming Random Forests*, that is able to handle evolving data streams using an entropy-based drift-detection technique. The algorithm automatically adjusts its parameters based on the data seen so far. Experimental results show that the algorithm handles multi-class problems for which the underlying class boundaries drift, without losing accuracy.

1 Introduction

Data-stream based applications are widely exploited by modern organizations. Such applications are required to analyze (or mine) streams of unlimited data records, by observing each data record only once, or possibly few times. Mining data streams require incremental, online, and fast algorithms which extract important information from data records on the fly, and can produce online results. Stream-mining algorithms should also adapt to changes in the distribution of the data, and be able to approximate results when necessary.

This paper addresses the data-stream classification problem. The field of stream classification has received much attention [1–5], but some problems exist:

- Algorithms typically require a large training time as a result of using huge amounts of data to build the classification model.
- Algorithms lack the ability to handle multi-class classification problems.
- Many algorithms have low classification accuracy and cannot handle *concept drift*.

Unlike standard classification algorithms that have three different sequential phases, (training, testing, and deployment), each associated with its own dataset, stream classification algorithms have only one stream of data containing both labelled and unlabelled data records. Based on the distribution of the labelled records through the input stream, possible scenarios are proposed by Abdulsalam *et al.*[6], and shown in Figure 1.

The Streaming Random Forests algorithm [6] is a stream classification algorithm combining the ideas of the standard Random Forests algorithm by Breiman [7] and streaming decision trees [1, 2]. It demonstrates good classification accuracy when tested. It cannot, however, deal with concept drift.

The main contribution of this paper is to define the *Dynamic Streaming Random Forests* algorithm, a stream-classification algorithm that extends the Streaming Random Forests algorithm. It is able to handle Scenario 1 and so successfully deals with

The work reported in this paper has been supported by Kuwait University.

concept changes. We incorporate ideas proposed by Vorburger and Bernstein [8] to define an entropy-based concept-change detection technique. In addition, the algorithm automatically adjusts its parameters based on the data seen so far. It handles only ordinal and numerical attributes, and it is able to handle multi-class problems. Experimental results show that our algorithm successfully handles concept drift.

The remainder of the paper is organized as follows: Section 2 states some background. Section 3 explains the Streaming Random Forests algorithm. Section 4 introduces the Dynamic Streaming Random Forests algorithm. Section 5 shows experimental results. Section 6 presents related work. Finally, Section 7 draws our conclusions.

2 Background

Streaming decision trees. A streaming decision tree under construction consists of internal nodes, containing an inequality on one of the attributes, *frontier nodes* that have not yet been either split or turned into leaves, and leaf nodes. Upon each record arrival, the record is routed down the tree, based on the attributes values and the inequalities of the internal nodes, until it reaches a frontier node. As each frontier node is considered, the Hoeffding bound [9] is used to decide if this node has accumulated enough records for a robust decision [1].

Using an entropy measure to detect concept drift. Shannon's entropy [10] is a measure of the disorder or randomness associated with a random variable. Finding the entropy for a dataset with known distribution requires using the following equation:

$$H(x) = -\sum_{x} P(x) \log_2(P(x)),$$

where x is a discrete random variable, and $P(x)$ is the probability mass function of x. For detecting concept changes using entropy, the two-windows paradigm [8, 11] is typically used, where two sliding windows are tracked; one window is the current window and the other is a reference window that represents the latest distribution. If the entropies of the two windows are different then a change has occurred.

The standard Random Forests algorithm. The Random Forests algorithm [7] is a classification ensemble technique developed by Breiman. It grows a number of binary decision trees and the classification for each new record is the plurality of the votes from the trees. It uses the Gini index for selecting split attributes. For a dataset of n records and m attributes, three conditions must be satisfied when growing each tree:

- A subset of n records, chosen from the original dataset at random with replacement must be used as the training set.
- A randomly-chosen $M \ll m$ attributes are evaluated in building each frontier node.
- No pruning is needed. Each tree is grown to the maximum size possible.

3 The Streaming Random Forests Algorithm

Streaming Random Forests [6] is a stream-classification algorithm that builds streaming decision trees with the techniques from Breiman's Random Forests. Its classification accuracies are approximately equal to those of Random Forests.

The Streaming Random Forests algorithm grows binary decision trees each from a different block of data. The trees are grown using the Hoeffding bounds to decide when

to stop building on each node [1]. The algorithm selects the attribute to split using the Gini index. It takes two parameters, namely the number of trees to be built and the number of records used to grow each tree ($tree\,window$).

Each newly arrived record is routed down the tree under construction, until it reaches a frontier node, where the attribute values of the record contribute to compute the Gini indexes. The values of each attribute (numerical or ordinal) are discretized into fixed-length intervals. The boundaries between the intervals are the possible split points.

As a frontier node of the current tree accumulates n_{min} records, where n_{min} is a defined parameter, both the Hoeffding and Gini tests are applied. If the Hoeffding test is satisfied, then the frontier node is transformed into an internal node with an inequality based on the best attribute and split point reported by the Gini index test.

If the number of records that have reached the frontier node exceeds a threshold ($node\,window$), and the node has not yet been split, the node is transformed into a leaf if the accumulated records are almost all from one class. Otherwise, the node is transformed into an internal node based on the best attribute and split point so far.

By the end of the $tree\,window$, a node might have been split and generated two frontier nodes that might have not seen enough data to get transformed into an internal or a leaf node. A limited form of pruning is therefore required to resolve this situation. A detailed explanation of the tree-building procedure and the tree-pruning method can be found in the Streaming Random Forests paper[6].

4 The Dynamic Streaming Random Forests Algorithm

The Dynamic Streaming Random Forests algorithm is a self-adjusting stream classification algorithm that is able to reflect concept changes. The algorithm, like the basic Streaming Random Forests algorithm, initially grows a defined number of trees. The difference is that the $tree\,window$ is not constant for all trees. Instead, whenever a number of $tree_{min}$ records have contributed to building a tree, the current classification error of the tree is calculated. If the error is greater than a threshold ($tree\,threshold$), then the algorithm stops building this tree and switches to building the next tree. Otherwise, the algorithm continues building the current tree using half the previous number of records, before it enters another test phase. Each tree has, therefore, a different $tree\,window$ that is never greater than $2 * tree_{min}$. The parameters $tree_{min}$ and $tree\,threshold$, are initially assigned defined values. A typical value for $tree\,threshold$ is $1/C$, where C is the number of classes. This threshold ensures that none of the trees performs worse than a random tree.

Once the total number of trees have been grown, the algorithm enters a test phase where the classification accuracy for the forest is calculated using previously unseen labelled data records. The classification error ($tree_{error_i}$), for each individual tree i, is also calculated and used to assign a weight to the tree.

In addition, the algorithm derives new values of the parameters to use in the subsequent building phase at time $t + 1$ (i.e. when a new block of labelled records arrives):

- A new $tree\,threshold$ is calculated as the average error of trees at time t:

$$tree\,threshold(t+1) = \frac{1}{U}\sum_{i=1}^{U} tree_{error_i}(t)$$

  where U is the number of trees in the forest.

– A new n_{min} is calculated using the equation:

$$n_{min}(t+1) = \frac{ln(1/\delta)}{2(\overline{\Delta Gini(t)})^2}$$

where $\Delta Gini(t) = Gini_{highest} - Gini_{second\_highest}$ is computed during the building phase at time t, and $\overline{\Delta Gini(t)}$ is the average value of $\Delta Gini(t)$.

– A new $tree_{min}$ is calculated using the equation:

$$tree_{min}(t+1) = n_{min}(t+1) * \overline{tree_{size}(t)}$$

where $\overline{tree_{size}(t)}$ is the average of all tree sizes from the building phase at time t.

Whenever a new block of labelled data records arrives, the algorithm replaces $w\%$ of the total number of trees with new trees, grown from newly arrived data records. The trees with the largest $tree_{error}$ are replaced. The value of w is obtained empirically. We choose it to be 25%. We believe that this percentage is enough to keep the learned concept up-to-date while not forcing too much time for each building phase. New trees are grown using the derived parameters. If there is a concept change in the data, then the values of these parameters are reset to their initial values since their most recent values were derived based on data records for the old concept. More trees are replaced in the case of concept changes to reflect the new distribution.

Concept drift can be thought of in different ways. There are three categories:

– *Category 1:* changes that are generated from variation in noise.
– *Category 2:* changes that are generated from a gradual change in the underlying rules defining the classes.
– *Category 3:* changes that are generated by a sudden change in the underlying rules.

We use an entropy measure to detect concept changes in streams based on the two-windows paradigm [8, 11]. We base our idea on the work done by Vorburger and Bernstein [8]. Since the distribution of the data is not known, the probability mass function cannot be used directly. Instead, we use counters to find the probabilities of occurrences of values associated with a specific interval and specific class within each attribute for the current and reference windows. The difference in entropies of the two windows for each attribute is calculated as:

$$H_i = |\sum_{c=1}^{C}\sum_{k=1}^{K}[u_{kc_{cur}} \log_2 u_{kc_{cur}} - u_{kc_{ref}} \log_2 u_{kc_{ref}}]|$$

where C is the number of classes, K is the number of intervals for attribute i, and $u_{kc_{cur}}$ and $u_{kc_{ref}}$ are the ratios of the number of records within an interval k and assigned to class c, to the total number of records of the current and reference windows, respectively. The absolute value is taken for H_i because the magnitude of the change is what really matters.

For a data stream with m attributes, the final entropy difference between the current and reference windows is calculated as the average of $H_i, 0 \leq i \leq m$:

$$H = \frac{1}{m}\sum_{i=1}^{m} H_i.$$

The possible values of H are in the range $[0, |\log_2 1/C|]$. H is zero when entropies of both windows are equal, and hence, there is no drift, while H is maximized and equal to $|\log_2 1/C|$ when a change appears.

The entropy H is normalized by dividing it by its maximum possible value: $H' = H/|\log_2 1/C|$. The value of H' is hence in the range [0,1] and represents the percentage of the concept change. The algorithm records a change in distribution if $H' > \gamma + H_{AVG}$, where H_{AVG} is the accumulated average entropy of the entropies computed since the last recorded change, and γ is a constant threshold defined empirically. If a change is detected, more trees must be replaced.

We use H' to find the percentage of the number of trees to replace (R):

$$R = \begin{cases} H' * U & if(H' * U + \frac{1}{C}((1-H')U) > U/2), \\ H' * U + U/2 & otherwise \end{cases}$$

This equation considers H' as the percentage of the trees to replace only if the remaining set of unchanged trees, which is calculated as $(1 - H')U$, has a higher probability of contributing to making the correct classification decision when combined with the replaced set of trees. We approximate the fraction of the number of unchanged trees that might positively share in the classification decision as $1/C$, since a tree with uniformly random guessing still has a probability of $1/C$ of getting the correct classification. If the number of trees to replace plus the number of remaining trees that are expected to give correct classification is less than half of the total number of trees, then the number of trees to replace is calculated as $H' * U + U/2$. The equation, therefore, forces the algorithm to replace at least half of the trees when a change occurs.

5 Experimental Settings and Results

We base the implementation of our algorithm on the Streaming Random Forests [6], implemented using Fortran 77. The machine on which we conduct our experiments uses a Pentium 4 3.2 GHz processor and 512MB RAM. We test our algorithm on synthetic datasets generated using the DataGen tool by Melli [12].

We generate a number of datasets and combine them into one dataset having concept changes of the three categories. The combined dataset has 5 attributes and 5 classes. The size of the dataset is 7 million records, with concepts changes described in Figure 2.

For all our experiments, the algorithm grows 50 trees using $M = 3$ attributes for each node. The initial values of $tree\,threshold$, n_{min}, and $tree_{min}$ are $1/C = 0.2$, 200, and 3000 respectively. The sizes of the current and reference windows are 1000 records. The value of γ we use is 0.05, and w is set to 25%.

We evaluate our algorithm based on the following criteria:

1. Entropy-based concept drift detection technique. Figure 3 shows the values of H', H_{AVG}, and $\gamma + H_{AVG}$ with respect to the number of records of the stream. The algorithm records seven concept changes at points 501000, 1506000, 2049000, 5011000, 5533000, 6032000, and 6501000. Note that a window of 1,000 records reflecting the new concept needs to be seen before detecting a change. Actual change points therefore, appear earlier than the detected changes. The first recorded change represents a drift of category 1. The entropy value of the second simulated drift, where the noise drops to 1% at point 1,000,000 does not show any significant change. This drift is therefore not

recorded. The subsequent five recorded changes represent the points during the gradually drifting concept (category 2 drift), where the noise is unchanged and equal to 1%. Some of the boundaries are, however, not recorded.

The last recorded change is the category 3 drift. All the recorded changes have similar entropy values in the range of [0.07,0.1] except for the last change, which has a value of 0.27. This is because both the noise and learning rules change.

2. Classification accuracy. Figure 4 shows the classification error, versus the number of processed records. The recorded points of the graph corresponds to each time the algorithm has finished building/modifying the forest.

The results show that the algorithm successfully reacts to concept changes of categories 1 and 3. This can be seen at the first two drift points of the data (category 1), and the last point (category 3). Although the entropy change-detection technique did not detect a change for the second change point (at record 1,000,000), the algorithm still performs well, giving an error that is approximately equal to the data noise.

The experiments record poor classification during the period of category 2 drift, shown at the point where the classification error is about 7.5% and the subsequent few points. The justification for this is that, since the data from new learning rules are added into the old dataset gradually, with the first block of mixed data having only 10% of the new records, the model does not see enough records to learn the new concept. Instead, it considers the data generated from new learning rules as an increase in the noise presented in the data. The classification error for this block of data should therefore, be around 10%. Our algorithm performs better, and records a classification error of 7.5%, which drops as the number of records from the new concept increases until it reaches classification errors in the range of [1.65%,2.55%].

3. Dynamic adjustment of parameters. We illustrate the dynamic nature of the algorithm by testing the variation of the parameter values during the execution of the algorithm. We consider $tree\,threshold$, n_{min}, $tree_{min}$, and the number of trees to replace. Figures 5, 6, 7, and 8 represent the values of the four parameters respectively.

The figures show that, when a concept change is detected, the values of the parameters are reset to their initial values. As shown in Figures 5, 6, and 7, the values of $tree\,threshold$, n_{min}, and $tree_{min}$ decrease when the model is better representing the data records for the following reasons: 1) The stronger the trees, the higher their classification accuracy, and therefore, the smaller the value of $tree\,threshold$. 2) Small values of n_{min} and $tree_{min}$ mean that the algorithm is performing well and does not require a large number of data records in order to update its model.

Figure 8 shows that 50 trees are grown initially, then 25% of the trees (12 trees) are normally replaced, unless a concept drift is detected, where the number of replaced trees increases based on the change significance. For the first six drift points, the number of trees to be replace varies between 29 and 30 trees, which shows that the significance of the changes are almost equal. The number replaced trees for the last drift is 39. This is because the value of the entropy difference for this drift is the highest.

6 Related Work

Decision trees based on Hoeffding bounds have been widely used for stream classification [1, 2, 5, 13]. Trees ensembles have also been used for stream classification [3–5].

Many of these algorithms are designed for only two class problems. Single decision tree algorithms typically require either significant processing time or memory to handle concept changes [2, 4]. CVFDT [2] handles concept changes by growing an alternative tree for each node. A node is replaced with its alternative tree whenever it records poor classification accuracy. Fan proposed an algorithm that depends on learning four models from combining old and new data [4]. The best model that represents the current data is selected for classification.

Ensemble classifiers have better utilization of data and are more accurate than single classifiers. Many of them are, however, tested only on two-class problems.

7 Conclusion

This paper has presented the *Dynamic Streaming Random Forests* algorithm, a self-adjusting stream-classification algorithm that handles evolving streams using an entropy-based change-detection technique. The algorithm extends the Streaming Random Forests algorithm [6]. It gives the expected behavior when tested on synthetic data with concept drift; the concept drift points were detected; the parameters were automatically adjusted, and the classification error was approximately equal to the noise in the data.

References

1. Domingos, P., Hulten, G.: Mining high-speed data streams. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining. (2000) 71–80
2. Hulten, G., Spencer, L., Domingos, P.: Mining time-changing data streams. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data mining. (2001) 97–106
3. Chu, F., Wang, Y., Zaniolo, C.: An adaptive learning approach for noisy data streams. In: Proceedings of the IEEE International Conference on Data Mining. (2004) 351–354
4. Fan, W.: A systematic data selection to mine concept-drifting data streams. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining. (2004) 128–137
5. Zhu, X., Wu, X., Yang, Y.: Dynamic classifier selection for effective mining from noisy data streams. In: Proceedings of the IEEE International Conference on Data Mining. (2004) 305–312
6. Abdulsalam, H., Skillicorn, D.B., Martin, P.: Streaming random forests. In: Proceeings of the International Database Engineering and Applications Symposium. (2007) 225–232
7. Breiman, L.: Random forests. Technical Report (1999) Available at www.stat.berkeley.edu.
8. Vorburger, P., Bernstein, A.: Entropy-based concept shift detection. In: Proceedings of the International Conference on Data Mining. (2006) 1113–1118
9. Hoeffding, W.: Probability inequalities for sums of bounded random variables. Journal of American Statistical Association **58**(1) (1963) 13–30
10. Shannon, C.E.: A mathematical theory of communication. In: ACM SIGMOBILE Mobile Computing and Communications Review. (2001) 5(1):355
11. Dasu, T., Krishnan, S., Venkatasubramanian, S., Yi, K.: An information-theoretic approach to detecting changes in multi-dimensional data streams. Technical Report (2005)
12. Melli, G.: Scds-a synthetic classification data set generator. Simon Fraser University, School of Computer Science (1997)
13. Gama, J., Rocha, R., Medas, P.: Accurate decision trees for mining high-speed data streams. In: Proceedings of the 9th ACM International Conference on Knowledge Discovery and Data Mining. (2003) 523–528

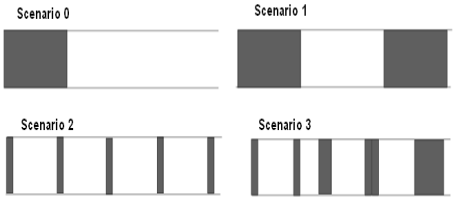

Fig. 1. Possible scenarios for data streams

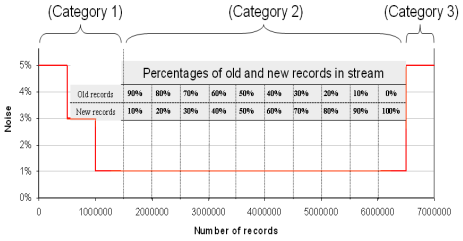

Fig. 2. Concept drift of the synthetic data

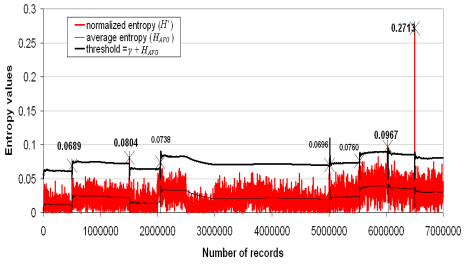

Fig. 3. Entropy versus number of records

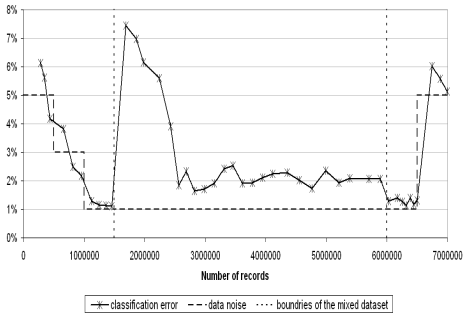

Fig. 4. Classification error of the forest

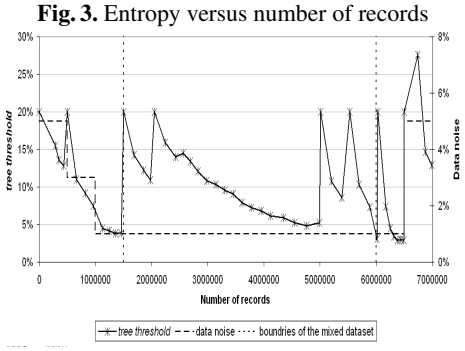

Fig. 5. $tree\,threshold$ values

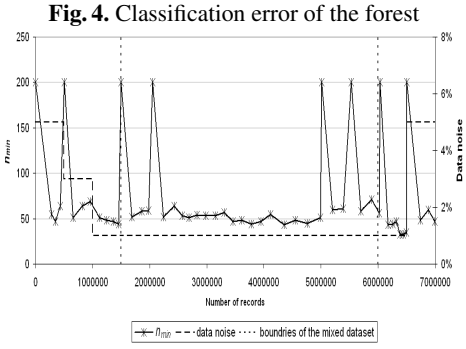

Fig. 6. n_{min} values

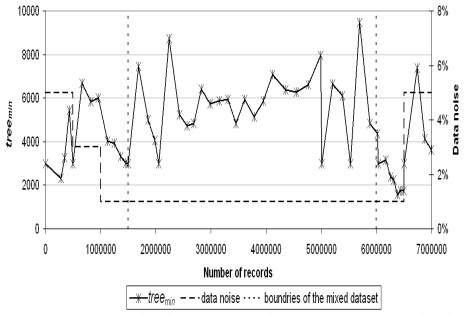

Fig. 7. $tree_{min}$ values

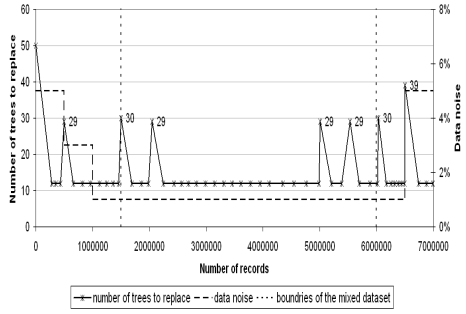

Fig. 8. Number of trees to replace