# An Adaptive Distributed Ensemble Approach to Mine Concept-drifting Data Streams

Gianluigi Folino, Clara Pizzuti, Giandomenico Spezzano
ICAR-CNR
Via Pietro Bucci, 41C
87036 Rende (CS), Italy
{folino,pizzuti,spezzano}@icar.cnr.it

## Abstract

*An adaptive boosting ensemble algorithm for classifying homogeneous distributed data streams is presented. The method builds an ensemble of classifiers by using Genetic Programming (GP) to inductively generate decision trees, each trained on different parts of the distributed training set. The approach adopts a co-evolutionary platform to support a cooperative model of GP. A change detection strategy, based on self-similarity of the ensemble behavior, and measured by its fractal dimension, permits to capture time-evolving trends and patterns in the stream, and to reveal changes in evolving data streams. The approach tracks online ensemble accuracy deviation over time and decides to recompute the ensemble if the deviation has exceeded a prespecified threshold. This allows the maintenance of an accurate and up-to-date ensemble of classifiers for continuous flows of data with concept drifts. Experimental results on a real life data set show the validity of the approach.*

## 1. Introduction

In recent years, many organizations are collecting tremendous amount of data, often generated continuously as a sequence of events and coming from different locations. Credit card transactional flows, telephone records, sensor network data, network event logs are just some examples of data streams. The design and development of fast, efficient, and accurate techniques, able to extract knowledge from these huge data sets too large to fit into the main memory of computers, poses significant challenges. In fact, traditional approaches assume that data is static, i.e. a concept, represented by a set of features, does not change because of modifications of the external environment. In the above mentioned applications, instead, a concept may drift due to several motivations, for example sensor failures, increases of telephone or network traffic. Concept drift can cause serious deterioration of the performance. In such a case the adopted method should be able to adjust quickly to changing conditions. Furthermore, data that arrives in the form of continuous streams usually is not stored, rather it is processed as soon as it arrives and discarded right away.

Incremental or online methods [6, 17] are an approach to large-scale classification on evolving data streams. These methods build a single model that represents the entire data stream and continuously refine this model as data flows. They are not able to capture new trends in the stream and might preclude that valuable older information be used since it is discarded as new one arrives. In the last few years, in the data mining community, approaches to processing data streams through classifier ensembles have been gaining an increasing interest. Many different proposals are described in [16, 18, 3, 4, 9, 15]. A survey on the most recent research topics can be found in [2]. However, the above approaches are not capable to deal with distributed streams. If data comes from different locations, it is necessary to gather all the data on a single location before processing. In many cases the cost of centralizing the data can be prohibitive and the owners may have privacy constraints. In [8], a method for aggregating decision trees built at distributed sites is presented. As noted in [13], this approach could be able to process distributed data streams if endowed with a streaming decision tree construction.

In this paper we approach the problem of large-scale distributed streaming classification by building an adaptive boosting ensemble of classifiers that combine the results of models trained on nodes of a distributed network, each containing its own local streaming data. The learned local models are obtained by using Genetic Programming ($GP$) [10], that inductively generate decision trees trained on different parts of the distributed training set. The method, named *StreamGP*, assumes that data is distributed, non-stationary, i.e. a concepts may drift, and arrives in the

form of multiple streams. *StreamGP* adopts a distributed co-evolutionary platform to support a cooperative model of GP. It evolves multiple predictors in the form of cooperative sub-populations and exploits the inherent parallelism of $GP$ by sharing the computational workload among computers over the network.

*StreamGP* is enriched with a change detection strategy that permits to capture time-evolving trends and patterns in the stream, and to reveal changes in evolving data streams. The strategy evaluates online accuracy deviation over time and decides to recompute the ensemble if the deviation has exceeded a pre-specified threshold. It is based on self-similarity of the ensemble behavior, measured by its fractal dimension, and allows revising the ensemble by promptly restoring classification accuracy. The method is efficient for two main reasons. First, each node of the network works with its local data, and communicate only the local model computed with the other peer-nodes to obtain the results. Second, once the ensemble has been built, it is used to predict the class membership of new streams of data and updated only when concept drift is detected. This means that each data block is scanned at most twice. The first time to predict the class label of the examples contained in that block. The second scan is executed only if the ensemble accuracy on that block is sensibly below the value obtained so far. In such a case, the *StreamGP* algorithm is executed to obtain a new set of classifiers to update the ensemble. Experimental results on a real life data set show the validity of the approach in maintaining an accurate and up-to-date GP ensemble.

The paper is organized as follows. The next section describes the fractal dimension concept. Section 3 describes the algorithm. In section 4, finally, the results of the method on a real life data set are presented.

## 2. Concept drift detection

The detection of changes in data streams is known to be a difficult task. When no information about the data distribution is available, an approach to cope with this problem is to monitor the performance of the algorithm by using the classification accuracy as a performance measure. The decaying of the predictive accuracy below a predefined threshold can be interpreted as a signal of concept drift. In such a case, however, the threshold must be tailored for the particular data set [19], since intrinsic accuracy can depends on background data. Furthermore, a naive test on accuracy does not take into account if the decrease is meaningful with respect to the the past history. We propose to track ensemble behavior by means of the concept of fractal dimension computed on the set of the most recent accuracy results.

*Fractals* [12] are particular structures that present *self-similarity*, i. e. an invariance with respect to the scale used. Many data sets, though are not fractals, exhibit a fractal behavior, that is they are self-similar over a large range of scales or size. This means that parts of any size of the data are similar to the whole data set. Examples of self-similar data sets come from different application domains such as economic markets, network and web site traffics, biology, geophysics, communication systems. Self-similarity can be measured using the *fractal dimension*. Intuitively, the fractal dimension measures the number of dimensions filled by the objects represented by the data set. It can be computed by embedding the data set in a $d$-dimensional grid whose cells have size $r$ and computing the frequency $p_i$ with which data points fall in the $i$-th cell. The fractal dimension $D$ [7] is given by the formula

$$D_q = \begin{cases} \frac{\partial log \sum_i p_i log p_i}{\partial log\ r} & \text{for q=1} \\ \frac{\partial log \sum_i p_i^q}{\partial log\ r} & \text{otherwise} \end{cases}$$

Among the fractal dimensions, the *Hausdorff fractal dimension* (q=0), the *Information Dimension* (q=1), and *Correlation dimension* (q=2) are the most used. The Information and Correlation dimensions are particularly interesting for data mining because the numerator of $D_1$ is the Shannon's entropy, and $D_2$ measures the probability that two points chosen at random will be within a certain distance of each other. Changes in the Information and Correlation dimensions mean changes in the entropy and the distribution of data, thus they can be used as an indicator of changes in data trends. Fast algorithms exist to compute the fractal dimension.The most known is the $FD3$ algorithm of [14] that implements the *box counting method* [11]. In the next section, a description of the ensemble GP algorithm for streaming data is given and the application of the fractal dimension to capture time-evolving trends in the stream is explained.

## 3. The StreamGP algorithm for streaming data

*StreamGP* is an ensemble GP boosting learning algorithm extended to deal with the classification of distributed continuous flows of data with concept drift. Figure 1 illustrates the ideas adopted by *StreamGP* to this end.

We assume a scenario where each node of the network receives continuous new data over time in batches. These batches (or blocks) contains labelled and unlabelled data. Labelled data are used by the learning system to train the ensemble and to update it when changes in data are discovered. Once the ensemble $E$ has been built, by running the boosting method on a number of blocks, it can be used to predict the class of the unlabelled data. As new data flows in, the adaptive *StreamGP* exploits the new training data to discover if concept drift has been occurred. In such a case it trains new classifiers, adds them to the GP ensemble by adopting a FIFO strategy that preserves the most

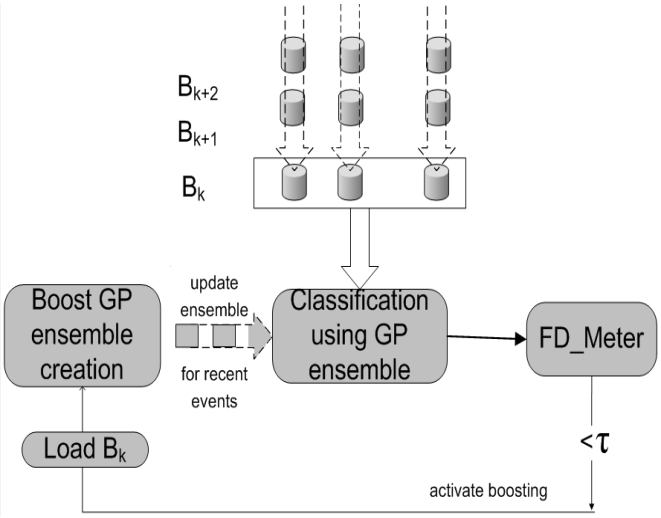


**Figure 1. GP ensemble with FD-meter**

recently generated classifiers and discards the older ones. Let $E = \{C_1, \ldots, C_M\}$ be the fixed ensemble size built so far. As data comes in, the ensemble prediction is evaluated on these new chunks of training data, and augmented misclassification errors, due to changes, are detected by using the module *FD-meter*. Suppose we have already scanned $k-1$ blocks $B_1, \ldots, B_{k-1}$ and computed the fitness values $\{f_1, \ldots, f_{k-1}\}$ of the ensemble on each block. Let $F = \{f_1, \ldots, f_H\}$ be the fitness values computed on the most recent $H$ blocks, and $F_d(F)$ be the fractal dimension of $F$. When the block $B_k$ is examined, let $f_k$ be the fitness value of the GP ensemble on it, and $F' = F \cup \{f_k\}$. FD-meter then checks whether $\mid (F_d(F) - F_d(F') \mid > \tau)$ where $\tau$ is a fixed threshold. In such a case the fractal dimension shows a variation and an alarm of change is set. This means that data distribution has been changed and the ensemble classification accuracy drops down. In the next section we experimentally show that this approach is very effective for the algorithm that is able to quickly adjust to changing conditions.

A detailed description of the algorithm in pseudo-code is shown in figure 2. Let a network of $p$ nodes be given, each having a streaming data set. Suppose $E = \{C_1, \ldots, C_M\}$ (step 1) is the ensemble stored so far and $F = \{f_1, \ldots, f_H\}$ (step 2) be the fitness values computed on the most recent $H$ blocks. As data continuously flows in, it is broken in blocks of the same size $n$. Every time a new block $B_k$ of data is scanned, the ensemble $E$ is evaluated on $B_k$ and the fitness value obtained $f_k$ is stored in the set $F'$ (steps 5-7).

Let $F_d(F)$ be the fractal dimension of $F$ and $F_d(F')$ the fractal dimension of $F$ augmented with the new fitness value $f_k$ obtained on the block $B_k$ (step 8). If it happens that $\mid (F_d(F) - F_d(F') \mid > \tau)$ (step 9), where $\tau$ is a fixed

**Algorithm *StreamGP:*** maintaining a GP ensemble $E$
Given a network constituted by $p$ nodes,
each having a streaming data set $S_i$
1. $E = \{C_1, \ldots, C_M\}$
2. $F = \{f_1, \ldots, f_H\}$
3. **for j = 1 . . . p** (each node in parallel)
4. **while** (more_Blocks)
5. Given a new block $B_k = \{(x_1, y_1), \ldots (x_n, y_n)\}$, $x_i \in X$ with labels $y_i \in Y = \{1, 2, \ldots, d\}$
6. evaluate the ensemble $E$ on $B_k$ and let $f_k$ be the fitness value obtained
7. $F' = F \cup f_k$
8. compute the fractal dimension $F_d(F')$ of the set $F'$
9. **if** $\mid (F_d(F) - F_d(F') \mid > \tau)$
10. Initialize the subpopulation $Q_i$ with random individuals
11. Initialize the example weights $w_i = \frac{1}{n}$ for $i = 1, \ldots, n$
12. **for** $t = 1, 2, 3, \ldots, T$ (for each round of boosting)
13. Train $CGPC$ on the block $B_k$ using a weighted fitness according to the distribution $w_i$
14. Learn a new classifier $C_t^j$
15. Exchange the $p$ classifiers $C_t^1, \ldots, C_t^p$ obtained among the $p$ processors
16. Update the weights
17. $E = E \cup \{C_t^1, \ldots, C_1^p\}$
18. **end for**
19. Update E by retiring the oldest classifiers
until $\mid E \mid < M$
20.**end if**
21. **end while**
22. **end parallel for**

**Figure 2. The *StreamGP algorithm***

threshold, then a change is detected, and the ensemble must adapt to these changes by retraining on the new block $B_k$. To this end the boosting standard method is executed for a number $T$ of rounds (steps 10-18). For every node $N_i$, $i = 1, \ldots, p$ of the network, a subpopulation $Q_i$ is initialized with random individuals (step 10) and the weights of the training instances are set to $1/n$, where $n$ is the data block size (step 11). Each subpopulation $Q_i$ is evolved for $T$ generations and trained on its local block $B_k$ by running a copy of the $CGPC$ algorithm (a cellular genetic programming method that generates a classifier as a decision tree) [5] (step 13). Then the $p$ individuals coming from each subpopulation (step 14) are exchanged among the $p$ nodes and constitute the ensemble of predictors used to determine the weights of the examples for the next round (steps 15-17). If the size of the ensemble is more than the maximum fixed

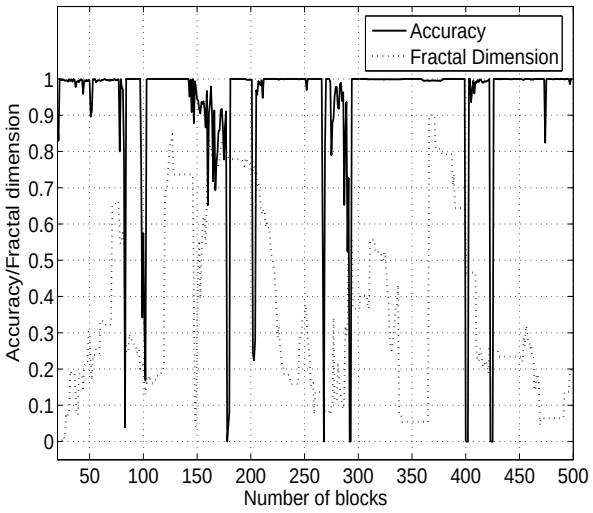


**Figure 3. Accuracy and fractal dimension values with ensemble size 100 and $\tau$ = 0.005 .**

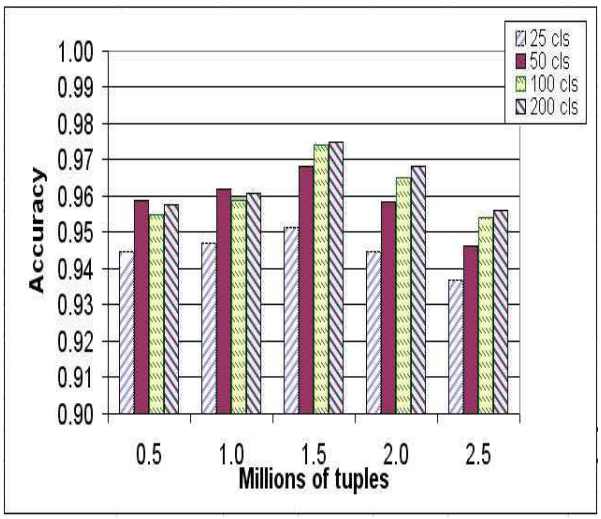


**Figure 4. Classification accuracy for different ensemble sizes.**

size $M$, the ensemble is updated by retiring the oldest $T \times p$ predictors and adding the new generated ones (step 19).

## 4. Experimental Results

In this section we study the effectiveness of our approach on a real-life application. The experiments were performed using a network composed by 5 1.133 Ghz Pentium III nodes having 2 Gbytes of Memory, interconnected over high-speed LAN connections.

To this end we used the KDD Cup 1999 Data set [1]. This data set comes from the 1998 DARPA Intrusion Detection Evaluation Data and contains training data consisting of 7 weeks of network-based intrusions inserted in the normal data, and 2 weeks of network-based intrusions and normal data for a total of 4,999,000 connection records described by 41 characteristics. The main categories of intrusions are four: Dos (Denial Of Service), R2L (unauthorized access from a remote machine), U2R (unauthorized access to a local super-user privileges by a local unprivileged user), PROBING (surveillance and probing). For the experiment we divided the data set in blocks of size 1k. On each node the algorithm receives a stream of 500 blocks, thus processing 500k tuples. Figure 3 shows the classification accuracy and the value of the fractal dimension when an ensemble of size 50 is used, with $\tau = 0.005$. The figure points out the abrupt alteration of accuracy because of the sudden change of the class distribution of the incoming data and the ability of the algorithm to quickly adapt to these new conditions.

Figure 4 shows the classification accuracy of the algorithm for an increasing number of tuples, when different ensemble sizes are used, namely 25, 50, 100, and 200 classifiers (cls stands for classifiers). Tuples are expressed in millions, thus 0.5 means 500,000 tuples, 1.0 one million of tuples, and so on until 2,500,000 tuples. For this data set increasing the size of the ensemble produces improvements in classification accuracy too, though the difference between 100 and 200 classifiers is minimal. Furthermore, the percentage of blocks on which the ensemble has to retrain because of change detection is 21.82%, 19.79%, 17.28%, 17.08% respectively for ensemble size 25, 50, 100, 200.

Finally we wanted to compare the performance of the algorithm against the simple one-pass algorithm that receives the entire data set at once. To this end we run *StreamGP* with an ensemble size of 50 and simulated the one-pass boosting method by using the entire data set scanned so far as a unique block. However, since the boosting rounds are 5, on 5 nodes, the ensemble generated by the one-pass method contains 25 classifiers. In order to have a fair comparison, the one-pass method had to run for 10 rounds so as to generate 50 classifiers. Figure 5 shows the classification accuracy for an increasing number of tuples, expressed in millions. The figure points out the better performance of the streaming approach. Another advantage to make clear is that the streaming method works on 1k tuples at a time, discarding them as soon as they have been processed. On the contrary, the one-pass method must maintain the entire data set considered so far, with considerable storage and time requirements. For example the one-pass boosting method working on a data set of 2,500,000 tuples needs 45280 seconds, while *StreamGP*, with $\tau = 0.01$, requires 7186 seconds, which is almost a magnitude order less.

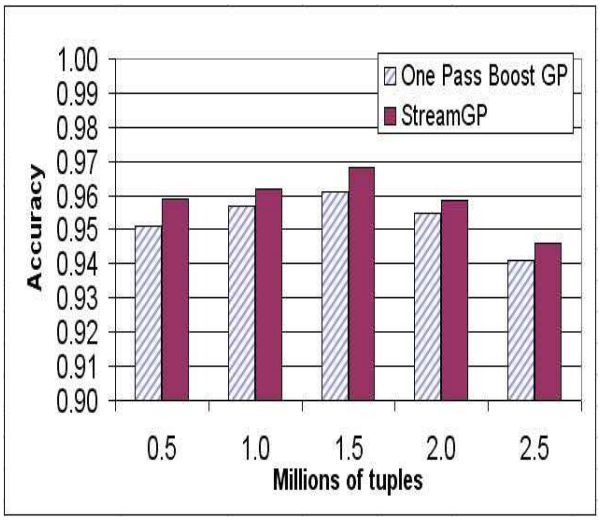


**Figure 5. Accuracy comparison between StreamGP and one-pass boosting method.**

## 5. Conclusions

The paper presented an adaptive GP boosting ensemble method for the classification of distributed homogeneous streaming data that comes from multiple locations. The main novelty of the approach are the extension of GP ensembles to deal with streaming data and the ability to handle concept drift via change detection. The approach is efficient since each node of the network works with its local streaming data, and communicate only the local model computed with the other peer-nodes. Furthermore, once the ensemble has been built, it is used to predict the class membership of new streams of data until concept drift is detected. Only in such a case the algorithm is executed to generate a new set of classifiers to update the current ensemble. Experimental results showed the validity of the approach in maintaining an accurate and up-to-date GP ensemble.

**Acknowledgements.** This work has been partially supported by the LOGICA project funded by Regione Calabria (Programma Operativo Regionale POR, Misura 3.16.B2).

## References

[1] The third international knowledge discovery and data mining tools competition dataset kdd99-cup. In *http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html*, 1999.

[2] C. C. Aggarwal. *Data Streams : Models and Algorithms*. Springer, 2007.

[3] F. Chu and C. Zaniolo. Fast and light boosting for adaptive mining of data streams. In H. Dai, R. Srikant, and C. Zhang, editors, *Proceedings of the 8th Pacific-Asia Conference (PAKDD 2004), May 26-28, 2004, Proceedings*, volume 3056 of *LNAI*, pages 282–292, Sydney, Australia, 2004. Springer Verlag.

[4] W. Fan. Systematic data selection to mine concept-drifting data streams. In *Proceedings of the 10th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining (KDD'04),*, pages 128–137, Seattle, WA, USA, 2004. ACM.

[5] G. Folino, C. Pizzuti, and G. Spezzano. Ensembles for large scale data classification. *IEEE Transaction on Evolutionary Computation*, 10(5):604–616, October 2006.

[6] J. Gehrke, V. Ganti, R. Ramakrishnan, and W. Loh. Boat - optimistic decision tree construction. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD'99)*, pages 169–180. ACM Press, 1999.

[7] P. Grassberger. Generalized dimensions of strange attractors. *Physics Letters*, 97A:227–230, 1983.

[8] H. Kargupta and B.-H. Park. A fourier spectrum-based approach to represent decision trees for mining data streams in mobile environments. *IEEE Transaction on Knowledge and Data Engineering*, 16(2):216–229, 2004.

[9] J. Z. Kolter and M. A. Maloof. Using additive expert ensembles to cope with concept drift. In *Proceedings of the 22nd Int. Conference on Machine Learning (ICML'05)*, pages 449–456, 2005.

[10] J. R. Koza. *Genetic Programming: On the Programming of Computers by means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[11] L. Liebovitch and T. Toth. A fast algorithm to determine fractal dimensions by box counting. *Physics Letters*, 141A(8):–, 1989.

[12] B. Mandelbrot. *The Fractal Geometry of Nature*. W.H Freeman, New York, 1983.

[13] S. Parthasarathy, A. Ghoting, and M. E. Otey. A survey of distributed mining of data streams. In C. C. Aggarwal, editor, *in Data Streams : Models and Algorithms*, pages 289–307. Springer, 2007.

[14] J. Sarraille and P. DiFalco. *FD3*. http://tori.postech.ac.kr/softwares.

[15] M. Scholz and R. Klinkenberg. Boosting classifiers for drifting concepts. *Intelligent Data Analysis*, 11(1):3–28, 2007.

[16] W. N. Street and Y. Kim. A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD International conference on Knowledge discovery and data mining (KDD'01),*, pages 377–382, San Francisco, CA, USA, August 26-29, 2001 2001. ACM.

[17] P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.

[18] H. Wang, W. Fan, P. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the nineth ACM SIGKDD International conference on Knowledge discovery and data mining (KDD'03)*, pages 226–235, Washington, DC, USA, August 24-27, 2003 2003. ACM.

[19] G. Widmer and M. Kubat. Learning in presence of concept drift and hidden contexts. *Machine Learning*, (23):69–101, 1996.