Mining Distributed Evolving Data Streams Using Fractal GP Ensembles

Gianluigi Folino, Clara Pizzuti, and Giandomenico Spezzano

Institute for High Performance Computing and Networking, CNR-ICAR
Via P. Bucci 41C
87036 Rende (CS), Italy
{folino,pizzuti,spezzano}@icar.cnr.it

Abstract. A Genetic Programming based boosting ensemble method for the classification of distributed streaming data is proposed. The approach handles flows of data coming from multiple locations by building a global model obtained by the aggregation of the local models coming from each node. A main characteristics of the algorithm presented is its adaptability in presence of concept drift. Changes in data can cause serious deterioration of the ensemble performance. Our approach is able to discover changes by adopting a strategy based on self-similarity of the ensemble behavior, measured by its fractal dimension, and to revise itself by promptly restoring classification accuracy. Experimental results on a synthetic data set show the validity of the approach in maintaining an accurate and up-to-date GP ensemble.

1 Introduction

Ensemble learning algorithms [1,5,2,8] based on Genetic Programming (GP) [11,16,12,3,7] have been gathering an increasing interest in the research community because of the improvements that *GP* obtains when enriched with these methods. These approaches have been applied to many real world problems and assume that all training data is available at once. However, in the last few years, many organizations are collecting a tremendous amount of data that arrives in the form of continuous stream. Credit card transactional flows, telephone records, sensor network data, network event logs are just some examples of streaming data. Processing these kind of data poses two main challenges to existing data mining methods. The first is relative to the performance and the second to adaptability.

Many data stream algorithms have been developed over the last decade for processing and mining data streams that arrive at a single location or at multiple locations. Some of these algorithms, known as centralized data stream mining (CDSM) algorithms, require that the data be sent to one single location before processing. These algorithms, however, are not applicable in cases where the data, computation, and other resources are distributed and cannot or should not be centralized for a variety of reasons e.g. low bandwidth, security, privacy issues, and load balancing. In many cases the cost of centralizing the data can

M. Ebner et al. (Eds.): EuroGP 2007, LNCS 4445, pp. 160–169, 2007.

be prohibitive and the owners may have privacy constraints. Unlike the traditional centralized systems, the distributed data mining (DDM) systems offer a fundamental distributed solution to analyze data without necessarily demanding collection of the data to a single central site. Typically DDM algorithms involve local data analysis to extract knowledge structures represented in models and patterns and the generation of a global model through the aggregation of the local results.

The ensemble paradigm is particularly suitable to support the DDM model. However, to extract knowledge from streaming information the ensemble must adapt its behavior to changes that occur into the data over time.

Incremental or online methods [9,18] are an approach able to support adaptive ensembles on evolving data streams. These methods build a single model that represents the entire data stream and continuously refine their model as data flows. However, maintaining a unique up-to-date model might preclude valuable information to be used since previously trained classifiers have been discarded. Furthermore, incremental methods are not able to capture new trends in the stream. In fact, traditional algorithms 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 ensemble performance and thus its detection allows to design an ensemble that is able to revise itself and promptly restore its classification accuracy.

Another approach to mine evolving data streams is to capture changes in data by measuring online accuracy deviation over time and deciding to recompute the ensemble if the deviation has exceeded a pre-specified threshold. These methods are more effective and allow to handle the concept drift problem in order to capture time-evolving trends and patterns in the stream.

In this paper we a propose a distributed data stream mining approach based on the adoption of an ensemble learning method to aggregate models trained on distributed nodes, and enriched with a change detection strategy to reveal changes in evolving data streams. We present an adaptive GP boosting ensemble algorithm for classifying data streams that maintains an accurate and up-to-date ensemble of classifiers for continuous flows of data with concept drifts. The algorithm uses a DDM approach where not only data is distributed, but also the data is non-stationary and arriving in the form of multiple streams. The method is efficient since each node of the network works with its local data, and communicate the local model computed with the other peer-nodes to obtain the results. A main characteristics of the algorithm is its ability to discover changes by adopting a strategy based on self-similarity of the ensemble behavior, measured by its fractal dimension, and to revise itself by promptly restoring classification accuracy. Experimental results on a synthetic 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 recall the ensemble technique. Section 3 presents the adaptive GP ensemble method, and the technique proposed for change detection. In section 4, finally, the results of the method on a synthetic data set are presented.

2 Ensemble for Streaming Data

An ensemble of classifiers is constituted by a set of predictors that, instead of yielding their individual decisions to classify new examples, combine them together by adopting some strategy [2,8,5,1]. Boosting is an ensemble technique introduced by Schapire and Freund [8] for boosting the performance of any "weak" learning algorithm, i.e. an algorithm that "generates classifiers which need only be a little bit better than random guessing". The boosting algorithm, called *AdaBoost*, adaptively changes the distribution of the training set depending on how difficult each example is to classify. Given the number T of trials (rounds) to execute, T weighted training sets $S_1, S_2, \ldots, S_T$ are sequentially generated and T classifiers $C^1, \ldots, C^T$ are built to compute a weak hypothesis h_t. Let w_i^t denote the weight of the example x_i at trial t. At the beginning $w_i^1 = 1/n$ for each x_i. At each round $t = 1, \ldots, T$, a weak learner C^t, whose error ϵ^t is bounded to a value strictly less than $1/2$, is built and the weights of the next trial are obtained by multiplying the weight of the correctly classified examples by $\beta^t = \epsilon^t/(1-\epsilon^t)$ and renormalizing the weights so that $\Sigma_i w_i^{t+1} = 1$. Thus "easy" examples get a lower weight, while "hard" examples, that tend to be misclassified, get higher weights.

In the last few years many approaches to processing data streams through classifier ensembles have been proposed. Street and Kim [17] build individual classifiers from chunks of data read sequentially in blocks. They are then combined into an ensemble of fixed size. When the ensemble is full, new classifiers are added only if they improve the ensemble's performance. Concept drift is treated by relying on the replacement policy of the method. Wang et al. [19] propose a framework for mining concept drifting data streams using weighted ensemble classifiers. The classifiers are weighted by estimating the expected prediction error on the test set. The size K of the ensemble is maintained constant by considering after each block of data the first top K weighted classifiers. Chu and Zaniolo [4] present a boosting algorithm modified to classify data streams able to handle concept drift via change detection. The boosting algorithm trains a new classifier on a data block whose instances are weighted by the ensemble built so far. Changes are discovered by modelling the ensemble accuracy as a random variable and performing a statistical test. When a change is detected the weights of the classifiers are reset to 1 and the boosting algorithm restarts. The ensemble is updated by substituting the oldest predictor with the last created.

As regards Genetic Programming, to the best of our knowledge, there is not any approach in the literature that cope with the extension of GP ensemble learning techniques to deal with streaming data. In the next section our adaptive GP boosting ensemble method is described.

3 Adaptive GP Boosting Ensemble

In this section the description of the algorithm *StreamGP* is given. The method builds an ensemble of classifiers by using, at every round of the boosting procedure, the algorithm *CGPC* [6] on each node to create a population of predictors. The ensemble is then used to predict the class membership of new streams of data and updated only when concept drift is detected. This behavior has a twofold advantage. The first is that it saves a lot of computation because the boosting algorithm is executed only if there is a significant deterioration of the ensemble performance. The second is that the ensemble is able to promptly adapt to changes and restore ensemble accuracy. Change identification is handled at every block. This means that each data block is scanned at most twice. The first time the ensemble predicts the class label of the examples contained in that block. The second scan is executed only if the ensemble accuracy on that block is below the value obtained so far. In such a case, in fact, the boosting algorithm is executed to obtain a new set of classifiers to update the ensemble.

3.1 *StreamGP*

StreamGP is an adaptive GP boosting ensemble algorithm for classifying data streams that applies the boosting technique in a distributed hybrid multi-island model of parallel GP. The hybrid model modifies the multi-island model by substituting the standard GP algorithm with a cellular GP algorithm. In the cellular model each individual has a spatial location, a small neighborhood and interacts only within its neighborhood. In our model we use the *CGPC* algorithm in each island. *CGPC* generates a classifier as a decision tree where the function set is the set of attribute tests and the terminal set are the classes. When a tuple has to be evaluated, the function at the root of the tree tests the corresponding attribute and then executes the argument that outcomes from the test. If the argument is a terminal, then the class name for that tuple is returned, otherwise the new function is executed. CGPC generates a classifier as follows. At the beginning, for each cell, the fitness of each individual is evaluated. The fitness is the number of training examples classified in the correct class. Then, at each generation, every tree undergoes one of the genetic operators (reproduction, crossover, mutation) depending on the probability test. If crossover is applied, the mate of the current individual is selected as the neighbor having the best fitness, and the offspring is generated. The current tree is then replaced by the best of the two offsprings if the fitness of the latter is better than that of the former. After the execution of the number of generations defined by the user, the individual with the best fitness represents the classifier.

The boosting schema is extended to cope with continuous flows of data and concept drift. Let M be the fixed size of the ensemble $E = \{C_1, \ldots, C_M\}$. Once the ensemble has been built, by running the boosting method on a number of blocks, the main aim of the adaptive *StreamGP* is to avoid to train new classifiers as new data flows in until the performance of E does not deteriorate very much, i.e. the ensemble accuracy maintains above an acceptable value.

```
      Given a network constituted by p nodes, each having a streaming data set
1.    E = ∅
2.    F = ∅
3.    for j = 1 ... p  (each island in parallel)
4.       while (more_Blocks)
5.          Given a new block B_k = {(x_1, y_1), ... (x_n, y_n)}, x_i ∈ X
            with labels y_i ∈ Y = {1, 2, ..., d}
6.          evaluate the ensemble E on B_k and let f_k be the fitness value obtained
7.          if | F |< H
8.             F = F ∪ f_k
9.          else F = {F − {f_1}} ∪ f_k
10.         compute the fractal dimension F_d of the set F
11.         if (F_d(F) < τ)
12.            Initialize the subpopulation Q_i with random individuals
13.            Initialize the example weights w_i = 1/n for i = 1, ..., n
14.            for t = 1, 2, 3, ..., T (for each round of boosting)
15.               Train CGPC on the block B_k using a weighted fitness
                  according to the distribution w_i
16.               Learn a new classifier C_t^j
17.               Exchange the p classifiers C_t^1, ..., C_t^p obtained among the p processors
18.               Update the weights
19.               E = E ∪ {C_t^1, ..., C_1^p }
20.            end for
21.            if (| E |> M) prune the ensemble E
22.         end if
23.      end while
24.   end parallel for
```

Fig. 1. The StreamGP algorithm

To this end, as data comes in, the ensemble prediction is evaluated on these new chunks of data, and augmented misclassification errors, due to changes in data, are detected by using the notion of *fractal dimension*, described in the next section, to the set $F = \{f_1, \ldots, f_H\}$ containing the last H fitness values obtained by evaluating the ensemble on the blocks. When an alarm of change is revealed, the GP boosting schema generates new classifiers, thus a decision on which classifiers must be discarded from the ensemble, because no longer consistent with the current concepts, has to be done. A simple technique, often adopted in many existing method, and also in our approach, is to eliminate the older predictors and substitute them with the most recent ones.

The description of the algorithm in pseudo-code is shown in figure 1. Let a network of p nodes be given, each having a streaming data set. 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 obtained so far is evaluated on B_k and the fitness value obtained f_k is stored in the set F (steps 5-6). $F = \{f_1, \ldots, f_H\}$ contains the last H evaluations of E on the data blocks, that is the fitness value set on which the fractal dimension $F_d(F)$ is computed (step 10). If $F_d(F)$ is

below a fixed threshold value τ (step 11), then it means that a change in data is detected, thus the ensemble must adapt to these changes by retraining on the new data set. To this end the boosting standard method is executed for a number T of rounds (steps 12-20). For every node N_i, $i = 1, \ldots, p$ of the network, a subpopulation Q_i is initialized with random individuals (step 12) and the weights of the training instances are set to $1/n$, where n is the data block size (step 13). 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 (step 15). Then the p individuals of each subpopulation (step 16) 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 17-19). If the size of the ensemble is more than the maximum fixed size M, the ensemble is pruned by retiring the oldest $T \times p$ predictors and adding the new generated ones (step 21).

3.2 Change Detection

An important step of the above described algorithm is the detection of changes in the data distribution that causes significant deterioration of the ensemble accuracy. In this section we propose to use the notion of *fractal dimension* to discover concept drift in streaming data. *Fractals* [14] are particular structures that present *self-similarity*, i. e. an invariance with respect to the scale used. The fractal dimension of fractal sets 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 [10] is given by the formula $D = \frac{1}{q-1} \frac{log \sum_i p_i^q}{log\ r}$. Among the fractal dimensions, the *correlation dimension*, obtained when $q = 2$ measures the probability that two points chosen at random will be within a certain distance of each other. Changes in the correlation dimension mean changes in the distribution of data in the data set, thus it can be used as an indicator of concept drift. Fast algorithms exist to compute the fractal dimension. We applied the $FD3$ algorithm of [15] that implements the *box counting method* [13]. In order to employ the fractal dimension concept to our approach, we proceed as follows. Suppose we have already scanned k blocks $B_1, \ldots, B_k$ and computed the fitness values $\{f_1, \ldots, f_k\}$ 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+1} is examined, let f_{k+1} be the fitness value of the GP ensemble on it. If $F_d(F \cup \{f_{k+1}\}) < \tau$, where τ is a fixed threshold, then the fractal dimension shows a decrease. This means that data distribution has been changed and the ensemble classification accuracy drops down. This approach has been shown to be very effective experimentally. In the next section we show that when the misclassification error of the ensemble increases, the fractal dimension drops down.

4 Experimental Results

In this section we study the effectiveness of our approach on a synthetic data set with two classes introduced in [4]. Geometrically the data set is a 5-dimensional

unit hypercube, thus an example x is a vector of 5 features $x_i \in [0, 1]$. The class boundary is a hyper-sphere of radius r and center c. If an example x is inside the sphere then it is labelled class 1, class 0 otherwise. Furthermore, the data set contains a noise level of 5% obtained by flipping randomly the class of the tuples from 0 to 1 and viceversa, with probability 0.05.

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. On each node a data set consisting of 360k tuples was generated by using as center the point $(0.5, 0.5, 0.5, 0.5, 0.5)$ and radius 0.25. The algorithm receives blocks of size 1k. Every 40k tuples the data set is perturbed by moving the center of 0.15 in a randomly chosen direction. This generates the migration of many points from class 0 to class 1 and viceversa. Thus, at blocks 40, 80, 120, 160 and so, i.e. each 40 blocks, concept drift is forced by provoking a deterioration of the ensemble performance that should be restored by our algorithm.

On each node a population of 100 predictors is evolved with a probability of 0.1 for reproduction, 0.8 for crossover and 0.1 for mutation. The maximum depth of the new generated subtrees is 4 for the step of population initialization, 17 for crossover and 2 for mutation. We used T=5 rounds for the boosting, each round executing for 100 generations. Thus $CGPC$ is trained on each block for 500 generations. At the end of the 5 rounds, each node produces 5 predictors, one for round, thus, since we have 5 nodes, 25 new classifiers are generated every time the fractal dimension diminishes below the threshold τ. This means that the oldest 25 classifiers are substituted by these new ones.

The experiments aimed at evaluating the ability of the approach in discovering concept drift and in restoring classification accuracy. Figure 2 reports the fitness, i.e the classification accuracy, and the value of the fractal dimension respectively when an ensemble of size 100 (on the left) and 200 (on the right) are used. The figure points out the abrupt deterioration of classification accuracy every 40 blocks of data (solid lines) and the corresponding decrease of the fractal dimension (dotted lines), thus allowing to reveal the change and to retrain the ensemble on the new block. Figures 2(a),(c),(e),(g) and (b),(d),(f),(h) show the effect of different values of the threshold τ, i.e. 0.80, 0.85, 0.87, 1.0, when the ensemble size is 100 and 200 respectively. The horizontal solid line on each figure indicates the τ value. The figures point out that, independently the value of τ, a larger ensemble is able to restore classification accuracy more quickly. Furthermore, higher the value of τ, faster the detection of change and more quickly the ensemble performance fixed up. In fact, every time the fractal dimension is below τ, the algorithm executes the boosting method, steps 12-21 in figure 1, and updates the ensemble with the new classifiers obtained by retraining $CGPC$ on the new data set. If $\tau = 1$ the retraining is executed at each block, that is it is assumed that changes occur at each new data block. Thus a higher value of τ implies a heavier computational load for the algorithm but a better classification accuracy. Its choice must take into account the trade-off between these two factors. In table 1 the percentage of blocks on which the boosting phase is executed

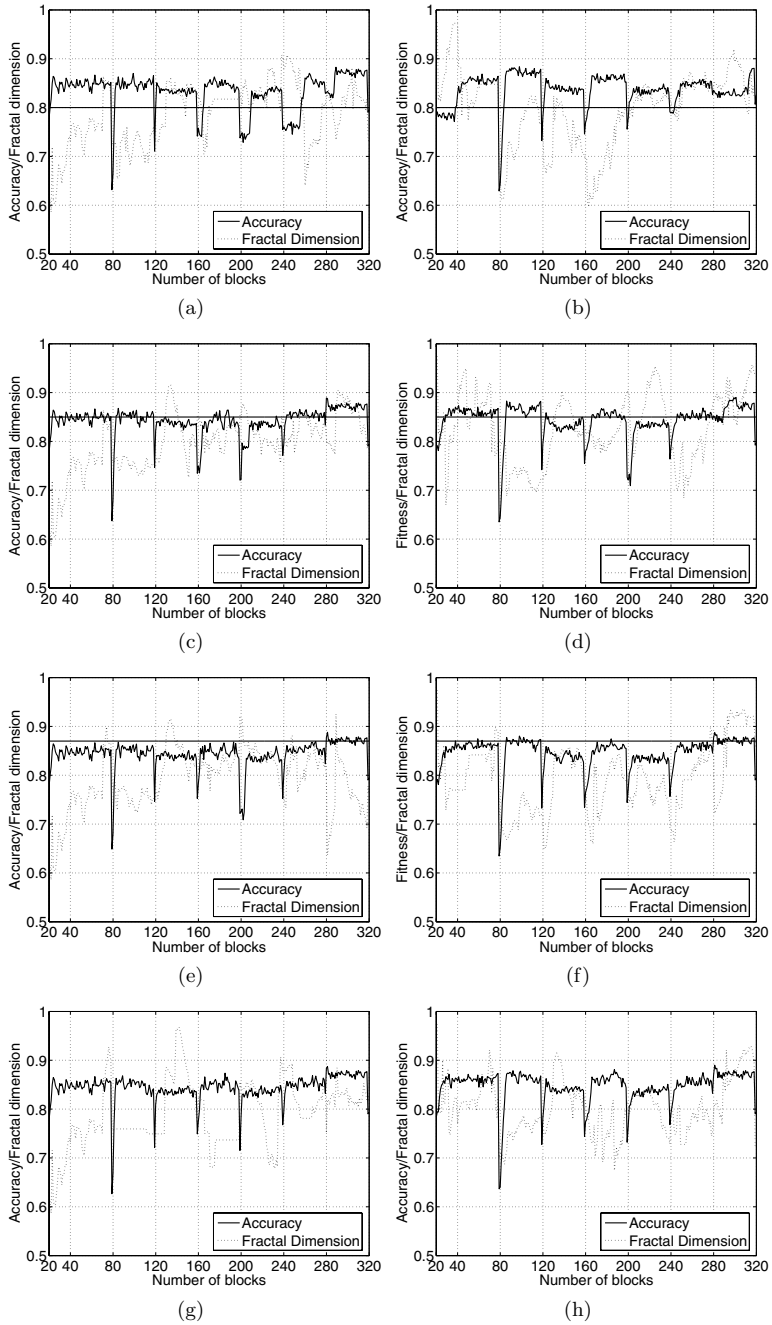

Fig. 2. Fitness (solid lines) and Fractal Dimension (dotted lines) with ensemble size 100, on the left, and 200 on the right, for different thresholds of F_d (a),(b): 0.80, (c),(d): 0.85, (e), (f): 0.87 and (g), (h): 1.0

and the corresponding mean classification accuracy are reported. For example, a 0.80 value for τ permits to save about 50% of computation, maintaining a good accuracy. An ensemble of 200 classifiers ulteriorly reduces the computational load since the retraining is executed for nearly the 35% of blocks. It is worth to note that the gain in accuracy when the boosting method is executed for each new block (last row of table 1) is marginal. For example, an ensemble of 200 predictors that is retrained at each block obtains an accuracy of 84.78, while if it is retrained on about the half of blocks, the accuracy is 84.37. This result substantiate the validity of the approach proposed.

Table 1. Percentage of blocks used in the training phase for different values of fractal threshold, and average classification accuracy

	100 classifiers		200 classifiers	
τ	Blocks	Accuracy	Blocks	Accuracy
0.80	47.21 %	83.28 %	34.90 %	83.73 %
0.85	82.72 %	84.18 %	54.84 %	84.37 %
0.87	93.69 %	84.36 %	82.11 %	84.69 %
1	100.0%	84.42%	100.0%	84.78 %

5 Conclusions

The paper presented a GP boosting ensemble method for the classification of distributed streaming data that comes from multiple locations. The method is able to handle concept drift via change detection. Changes are discovered by adopting a strategy based on self-similarity of the ensemble behavior, measured by its fractal dimension. This allows the ensemble to revise itself and promptly restore classification accuracy. Experimental results on a synthetic data set showed the validity of the approach in maintaining an accurate and up-to-date GP ensemble. Future work aims at studying parameter tuning and to test the approach on real streaming data sets.

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. Eric Bauer and Ron Kohavi. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, (36):105–139, 1999.
2. Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
3. E. Cantú-Paz and C. Kamath. Inducing oblique decision trees with evolutionary algorithms. *IEEE Transaction on Evolutionary Computation*, 7(1):54–68, February 2003.

4. Fang Chu and Carlo Zaniolo. Fast and light boosting for adaptive mining of data streams. In Honghua Dai, Ramakrishnan Srikant, and Chengqi 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.
5. Thomas G. Dietterich. An experimental comparison of three methods for costructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, (40):139–157, 2000.
6. G. Folino, C. Pizzuti, and G. Spezzano. A cellular genetic programming approach to classification. In *Proc. Of the Genetic and Evolutionary Computation Conference GECCO99*, pages 1015–1020, Orlando, Florida, July 1999. Morgan Kaufmann.
7. G. Folino, C. Pizzuti, and G. Spezzano. A scalable cellular implementation of parallel genetic programming. *IEEE Transaction on Evolutionary Computation*, 10(5):604–616, October 2006.
8. Y. Freund and R. Scapire. Experiments with a new boosting algorithm. In *Proceedings of the 13th Int. Conference on Machine Learning*, pages 148–156, 1996.
9. 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.
10. P. Grassberger. Generalized dimensions of strange attractors. *Physics Letters*, 97A:227–230, 1983.
11. Hitoshi Iba. Bagging, boosting, and bloating in genetic programming. In *Proc. Of the Genetic and Evolutionary Computation Conference GECCO99*, pages 1053–1060, Orlando, Florida, July 1999. Morgan Kaufmann.
12. W.B. Langdon and B.F. Buxton. Genetic programming for combining classifiers. In *Proc. Of the Genetic and Evolutionary Computation Conference GECCO'2001*, pages 66–73, San Francisco, CA, July 2001. Morgan Kaufmann.
13. L. Liebovitch and T. Toth. A fast algorithm to determine fractal dimensions by box counting. *Physics Letters*, 141A(8):–, 1989.
14. B. Mandelbrot. *The Fractal Geometry of Nature*. W.H Freeman, New York, 1983.
15. J. Sarraille and P. DiFalco. *FD3*. http://tori.postech.ac.kr/softwares.
16. Terence Soule. Voting teams: A cooperative approach to non-typical problems using genetic programming. In *Proc. Of the Genetic and Evolutionary Computation Conference GECCO99*, pages 916–922, Orlando, Florida, July 1999. Morgan Kaufmann.
17. W. Nick Street and YongSeog 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.
18. P. E. Utgoff. Incremental induction of decision trees. *Machine Learning*, 4:161–186, 1989.
19. H. Wang, Wei Fan, P.S. 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.