StreamGP: Tracking Evolving GP Ensembles in Distributed Data Streams using Fractal Dimension

Gianluigi Folino
ICAR-CNR
Via Pietro Bucci, 41C
87036 Rende (CS), Italy
folino@icar.cnr.it

Clara Pizzuti
ICAR-CNR
Via Pietro Bucci, 41C
87036 Rende (CS), Italy
pizzuti@icar.cnr.it

Giandomenico Spezzano
ICAR-CNR
Via Pietro Bucci, 41C
87036 Rende (CS), Italy
spezzano@icar.cnr.it

ABSTRACT

The paper presents an adaptive GP boosting ensemble method for the classification of distributed homogeneous streaming data that comes from multiple locations. The approach is able to handle concept drift via change detection by employing a change detection strategy, based on self-similarity of the ensemble behavior, and measured by its fractal dimension. It 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 on a synthetic and real life data set showed the validity of the approach in maintaining an accurate and up-to-date GP ensemble.

Categories and Subject Descriptors

H.2.8 [**Database Managment**]: Database Applications — *Data Mining*; I.2.2 [**Artificial Intelligence**]: Automatic Programming

General Terms

Algorithms

Keywords

Genetic Programming, Data Mining, Classification, Ensemble, Distributed Streaming Data.

1. THE GP ENSEMBLE METHOD FOR STREAMING DATA

In this paper we deal with the problem of large-scale distributed streaming classification by building an adaptive *GP* ensemble of classifiers that combine the results of *GP* classifiers, trained on nodes of a distributed network, each containing their own local streaming data. 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* uses a distributed cooperative co-evolutionary algorithm, which evolves multiple predictors in the form of cooperative subpopulations and exploits the inherent parallelism of *GP* by sharing the computational workload among computers over the network.

The algorithm adopts the idea of assigning a subpopulation to each node of the network to build classifiers on local data sites. The classifiers of each subpopulation are trained by using the *CGPC* algorithm and combined together to classify new tuples by applying a majority voting scheme. Classifiers are periodically exchanged among subpopulations to transfer the knowledge acquired on the own local data. *StreamGP* models the system as a collection of cooperative autonomous sub-populations trained on distributed data streams. Sub-populations run on the various hosts within an heterogeneous network that works as a peer-to-peer system. Only the outermost individuals are asynchronously exchanged among the neighbor sub-populations. After a fixed number of generations, from each subpopulation the tree having the best fitness is chosen as representative and output as the hypothesis computed. Then the p individuals computed are exchanged among the nodes of the network and constitute the ensemble of predictors used to determinate the weights of the examples for the next round. The evolutionary process is cooperative because the fitness of the individuals of each sub-population is calculated by using the representative individuals of all the other populations located on the network nodes.

Once the ensemble has been built, the main aim of the method is to avoid to train new classifiers as new data flows in until the performance does not deteriorate too much. The boosting schema is extended to cope with continuous flows and it 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. Each data block is thus 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 sensibly below the value obtained so far. In such a case, the algorithm is executed to obtain a new set of classifiers to update the ensemble.

Acknowledgements. Work partially supported by LOGICA POR 3.16.B2, and SFIDA-PMI 4446/ICT MUR projects.