Learning from Data Streams

João Gama
University of Porto, Portugal

Pedro Pereira Rodrigues
University of Porto, Portugal

INTRODUCTION

In the last two decades, machine learning research and practice has focused on batch learning usually with small datasets. In batch learning, the whole training data is available to the algorithm that outputs a decision model after processing the data eventually (or most of the times) multiple times. The rationale behind this practice is that examples are generated at random accordingly to some stationary probability distribution. Also, most learners use a greedy, hill-climbing search in the space of models.

What distinguishes current data sets from earlier ones are the continuous flow of data and the automatic data feeds. We do not just have people who are entering information into a computer. Instead, we have computers entering data into each other. Nowadays there are applications in which the data is modelled best not as persistent tables but rather as transient data streams. In some applications it is not feasible to load the arriving data into a traditional DataBase Management Systems (DBMS), and traditional DBMS are not designed to directly support the continuous queries required in these application (Babcock et al., 2002). These sources of data are called *Data Streams*. There is a fundamental difference between learning from small datasets and large datasets. As pointed-out by some researchers (Brain & Webb, 2002), current learning algorithms emphasize variance reduction. However, learning from large datasets may be more effective when using algorithms that place greater emphasis on bias management.

Algorithms that process data streams deliver approximate solutions, providing a fast answer using few memory resources. They relax the requirement of an exact answer to an approximate answer within a small error range with high probability. In general, as the range of the error decreases the space of computational resources goes up. In some applications, mostly database oriented, an approximate answer should be within an admissible error margin. Some results on tail inequalities provided by statistics are useful to accomplish this goal.

LEARNING ISSUES: ONLINE, ANYTIME AND REAL-TIME LEARNING

The challenge problem for data mining is the ability to permanently maintain an accurate decision model. This issue requires learning algorithms that can modify the current model whenever new data is available at the rate of data arrival. Moreover, they should forget older information when data is out-dated. In this context, the assumption that examples are generated at random according to a stationary probability distribution does not hold, at least in complex systems and for large time periods. In the presence of a non-stationary distribution, the learning system must incorporate some form of forgetting past and outdated information. Learning from data streams require incremental learning algorithms that take into account concept drift. Solutions to these problems require new sampling and randomization techniques, and new approximate, incremental and decremental algorithms. In (Hulten & Domingos, 2001), the authors identify desirable properties of learning systems that are able to mine continuous, high-volume, open-ended data streams as they arrive: *i*) incrementality, *ii*) online learning, *iii*) constant time to process each example using fixed memory, *iv*) single scan over the training data, and *v*) tacking drift into account.

Examples of learning algorithms designed to process open-ended streams include predictive learning: Deci-

sion Trees (Domingos & Hulten, 2000; Hulten et al., 2001; Gama et al., 2005, 2006), Decision Rules (Ferrer et al., 2005); descriptive learning: variants of k-Means Clustering (Zhang et al., 1996; Sheikholeslami et al., 1998), Clustering (Guha et al., 1998; Aggarwal et al., 2003), Hierarchical Time-Series Clustering (Rodrigues et al., 2006); Association Learning: Frequent Itemsets Mining (Jiang & Gruemwald, 2006), Frequent Pattern Mining (Jin & Agrawal 2007); Novelty Detection (Markou & Singh, 2003; Spinosa et al. 2007); Feature Selection (Sousa et al., 2006), etc.

All these algorithms share some common properties. They process examples at the rate they arrive using a single scan of data and fixed memory. They maintain a decision model at any time, and are able to adapt the model to the most recent data.

Incremental and Decremental Issues

The ability to update the decision model whenever new information is available is an important property, but it is not enough. Another required operator is the ability to *forget* past information (Kifer et al., 2004). Some data stream models allow delete and update operators. For example, *sliding windows* models require the forgetting of old information. In these situations the incremental property is not enough. Learning algorithms need forgetting operators that reverse learning: decremental unlearning (Cauwenberghs & Poggio, 2000).

Cost-Performance Management

Learning from data streams require to update the decision model whenever new information is available. This ability can improve the *flexibility* and *plasticity* of the algorithm in fitting data, but at some *cost* usually measured in terms of resources (time and memory) needed to update the model. It is not easy to define where is the trade-off between the benefits in flexibility and the cost for model adaptation. Learning algorithms exhibit different profiles. Algorithms with strong variance management are quite efficient for small training sets. Very simple models, using few free-parameters, can be quite efficient in variance management, and effective in incremental and decremental operations (for example naive Bayes (Domingos & Pazzani, 1997)) being a natural choice in the sliding windows framework. The main problem with simple approaches is the boundary in generalization performance they can achieve; they are limited by high bias. Complex tasks requiring more complex models increase the search space and the cost for structural updating. These models require efficient control strategies for the trade-off between the gain in performance and the cost of updating.

Monitoring Learning

Whenever data flows over time, it is highly improvable the assumption that the examples are generated at random according to a stationary probability distribution (Basseville & Nikiforov, 1993). At least in complex systems and for large time periods, we should expect changes (smooth or abrupt) in the distribution of the examples. A natural approach for these *incremental tasks* is *adaptive learning algorithms*, incremental learning algorithms that take into account concept drift.

Change Detection

Concept drift (Klinkenberg, 2004, Aggarwal, 2007) means that the concept about which data is being collected may shift from time to time, each time after some minimum permanence. Changes occur over time. The evidence for changes in a concept is reflected in some way in the training examples. Old observations, that reflect the behaviour of nature in the past, become irrelevant to the current state of the phenomena under observation and the learning agent must forget that information. The nature of change is diverse. Changes may occur in the context of learning, due to changes in hidden variables, or in the characteristic properties of the observed variables.

Most learning algorithms use blind methods that adapt the decision model at regular intervals without considering whether changes have really occurred. Much more interesting are explicit change detection mechanisms. The advantage is that they can provide meaningful description (indicating change-points or small time-windows where the change occurs) and quantification of the changes. They may follow two different approaches:

- Monitoring the evolution of performance indicators adapting techniques used in Statistical Process Control (Gama et al., 2004).
- Monitoring distributions on two different time windows (Kiffer et al., 2004). The method monitors the evolution of a distance function between