Evaluating Algorithms that Learn from Data Streams

João Gama JGAMA@LIAAD.UP.PT
LIAAD-INESC Porto, Portugal

Pedro Pereira Rodrigues PPRODRIGUES@FC.UP.PT
LIAAD-INESC Porto & Faculty of Sciences, University of Porto, Portugal

Gladys Castillo GLADYS@MAT.UA.PT
University Aveiro, Portugal

Abstract

In the past years, the theory and practice of machine learning and data mining have been focused on static and finite data sets from where learning algorithms generate a static model. In this setting, evaluation metrics and methods are quite well defined. Nowadays, several sources produce data in a stream at high-speed, creating environments with possibly infinite, dynamic and transient data streams. Currently, there is no standard for evaluating algorithms that learn from data streams. In this paper we try to present major issues that bind the evaluation strategy to data stream environments, proposing evaluation methods for both supervised and unsupervised learning.

1. Introduction

The last twenty years or so have witnessed large progress in machine learning and in its capability to handle real-world applications. Nevertheless, machine learning so far has mostly centered on one-shot data analysis from homogeneous and stationary data, and on centralized algorithms. Most of Machine Learning and Data Mining approaches assume that examples are independent, identically distributed and generated from a stationary distribution. Most of learning algorithms assume that computational resources are unlimited, that is data fits in main memory. In that context, standard data mining techniques use finite training sets and generate static models. Nowadays we are faced with tremendous amount of distributed data that could be generated from the ever increasing number of smart devices. In most cases, this data is transient, and may not be stored in permanent relations.

Examples of data mining applications that are faced with this scenario include sensor networks, social networks, user modelling, radio frequency identification, web mining, scientific data, financial data, etc. Data continuously flow possibly at high-speed, in a dynamic and time-changing environment. Data mining in these contexts require a continuous processing of the incoming data monitoring trends, and detecting changes. Traditional one-shot systems, memory based, trained from fixed training sets and generating static models are not prepared to process the high detailed data available, they are not able to continuously maintain a predictive model consistent with the actual state of the nature, nor are they ready to quickly react to changes.

Our ability to collect data is changing dramatically. Nowadays, computers and small devices send data to other computers. We are faced with the presence of distributed sources of detailed data. Data continuous flow eventually at high-speed generated from non-stationary processes. Most recent learning algorithms (Cormode et al., 2007; Babcock et al., 2003; Domingos & Hulten, 2000; Hulten et al., 2001; Gama et al., 2003; Ferrer-Troyano et al., 2004; Gama & Rodrigues, 2007) maintain a decision model that continuously evolve over time, taking into account that the environment is non-stationary and computational resources are limited.

P. Domingos and G. Hulten (Hulten & Domingos, 2001) identify desirable properties of learning systems for efficient mining continuous, high-volume, open-ended data streams: *i)* Require small constant time

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

per data example; *ii)* Use fix amount of main memory, irrespective to the total number of examples; *iii)* Built a decision model using a single scan over the training data; *iv)* Generating a any time model independent from the order of the examples; and *v)* Ability to deal with concept drift. For stationary data, ability to produce decision models that are nearly identical to the ones we would obtain using batch learner.

From this desiderata, we can identify 3 dimensions that influence the learning process: *space* – the available memory is fixed, *learning time* – process incoming examples at the rate they arrive, and *generalization power* – how effective the model is at capturing the true underlying concept. In this work we focus in the generalization power of the learning algorithm, although we recognize that the two first dimensions have direct impact in the generalization power of the learned model.

Data streams are open-ended. This could facilitate the evaluation methodologies, because we have training and test set as large as desired. Two aspects, in the emerging applications and learning algorithms that have strong impact in the evaluation methodologies are the continuous evolution of decision models and the non-stationary nature of data streams. In this work we discuss

2. Evaluation Issues

A key point in any intelligent system is the evaluation methodology. Learning systems generate compact representations of what is being observable. They should be able to improve with experience and continuously self-modify their internal state. Their representation of the world is approximate. How approximate is the representation of the world?

Evaluation is used in two contexts: inside the learning system to assess hypothesis, and as a wrapper over the learning system to estimate the applicability of a particular algorithm in a given problem. Three fundamental aspects are:

- What are the goals of the learning task?
- Which are the evaluation metrics?
- How to design the experiments to estimate the evaluation metrics?

2.1. Supervised Learning

In most supervised tasks for machine learning, for each new example an exact loss function may be computed with respect to a previously made prediction. This is especially useful to assess the quality of online predictive models. For predictive learning tasks (classification, and regression) the learning goal is to induce a function $\hat{y} = f(\vec{x})$. The most relevant dimension is the *generalization error*. It is an estimator of the difference between $\hat{f}$ and the unknown f, and an estimate of the loss that can be expect when applying the model to future examples. In online learners, this estimate can be used not only to assess the quality of the model, but also to tune the model's parameters before applying it to future examples.

Given the online setting of learning from data streams, the quality of a learning model is difficult to condense in a single value of loss or performance, since data is being produced with evolving concepts and the model itself is being continuously updated. In previous works, evaluation of online learners has been sidestepped, assuming either a hold-out test set or a fixed test set at the end of the learning process, computing average losses in time windows.

The VFDT system (Domingos & Hulten, 2000) and the CVFDT system (Hulten et al., 2001), fast decision-tree learners without/with concept drift detection ability, were evaluated in synthetic data by keeping a hold-out test set of the current concepts, in which the system was tested every bunch of examples, being afterwards computed the averaged loss. When applying them to real data, while the first approach was measured in a test set at the end of the learning process, the second approach was evaluated by measuring the loss online, at each new example, updating the model with it afterwards. Final performance was the averaged accuracy, although a simple observation is also made regarding the evolution of the error. In (Gama et al., 2003), the VFDT system was enhanced with better predictors at the leaves and allowing the continuous update of the model with a single scan of data (a major requirement when learning from data streams). However The VFDTc system was evaluated likewise by the averaged loss on a hold-out test set.

Another usual technique applied to data streams is the induction of online neural networks. In (Gama & Rodrigues, 2007) online neural networks are learned from time series data, each example being fed only once to the model after the loss of predicting it has already been computed. The evaluation was carried out by computing the averaged loss in natural time windows, with respect to the real-world domain, but without any analysis of error evolution.

2.2. Unsupervised Learning

On unsupervised learning tasks, evaluating the learned model is not straightforward as no objective correct solution is usually known (Jain & Dubes, 1988). More than being hard to evaluate from the point of view of its application to real data, unsupervised learners are weak in self-evaluating their own learning process, which narrows their sustainability to learn from data streams.

Clustering is possibly the most popular unsupervised learning task in data mining. Given the application to data streams, the output of clustering systems is usually given by k centers, rather that an assignment of clusters to the data. Most common validity measures for clustering structures, based on compactness and separability measures, improve with the number of clusters found. This is one of the major issues in clustering validity. Not only the algorithm must find the best partition of the data into k clusters, but its performance will highly depend on the k number itself. This way, it should also be able to define k as an output of the system, in order to find the best partition of data into the best number of clusters possible. This is why most batch clustering procedures assume k is given by the user, or it is estimated using several runs on the same data, clearly not possible in the data stream framework, even in static environments. Furthermore, the performance of clustering procedures is usually highly dependent from the choice of the data set: most methods may give good results for a particular type of data but otherwise perform poorly (Beringer & Hüllermeier, 2006).

The previous facts have also a wide range of implications in the self-evaluation of online methods. On supervised learners, parameters are tuned according to the value of a given error measure with respect to the real data. On unsupervised learning, not only a error measure is hardly available, but also the definition of parameters may have an extremely high impact on the quality measure, hence the inapplicability as a parameter tuner.

Learning clustering structures from streaming examples has been addressed in several works, with different evaluating approaches (Aggarwal et al., 2003; Spiliopoulou et al., 2006; Cormode et al., 2007). Current works on learning clusters of streaming time series also presents different analysis of cluster validity (Beringer & Hüllermeier, 2006; Rodrigues et al., 2008). Most of them, however, lack the ability to track quality evolution in a meaningful way.

3. Design of Evaluation Methods

Data stream scenarios focus machine learning research in a precise problem: the design of evaluation methods for models that learn from data streams.

3.1. Supervised Learning

In batch learning using finite training sets, cross-validation and variants (leave-one-out, bootstrap) are the standard method to evaluate learning systems. Cross-validation is appropriate for restricted size datasets, generated by stationary distributions, and assuming examples are independent. In data streams context, where data is potentially infinite, the distribution generating examples and decision models evolve over time, cross-validation is not applicable, and research communities need other evaluation strategies.

Two viable alternatives are: *i)* Holdout an independent test set. Apply the current decision model to the test set, at regular time intervals (or set of examples); *ii)* Predictive Sequential: *Prequential* (Dawid, 1984), where the error of a model is computed from the sequence of examples. For each example the actual model makes a prediction based only on the example attribute-values. The prequential-error is a computed based on an accumulated sum of a loss function between the prediction and observed values.

Both methods provide a learning curve that monitor the evolution of learning as a process, the disadvantage is that both estimates are affected by the order of the examples. In the case of stationary data streams both might provide reliable error estimates. Nevertheless, the applicability of the holdout method in non-stationary streams is questionable.

3.2. Unsupervised Learning

In unsupervised learning, no *real truth* is available to the learning process in order to assess the quality of the resulting model. Moreover, most common clustering evaluation metrics require the entire set of examples to determine the quality of the clustering structure. This is obviously not possible in the data stream context, being a major problem in applying machine learning to data streams. Therefore, systems should being able to compress information while giving more relevance to recent examples.

The range of recent clustering algorithms that operate online over data streams is wide. A common connecting feature is the definition of unit cells or representative points, from which clustering can be obtained with less computational costs (O'Callaghan et al., 2002;

Zhang et al., 1996). Afterwards, two alternatives are viable for evaluating hypotheses: *i)* apply incremental validity measures, which are defined by previous value and new data; *ii)* assessing quality by validity measures computed using only the representatives of the complete data set.

Although resulting in exact values for the clustering quality, given the usual definitions for validity indices which are based on compactness and separability, the first alternative is extremely hard to use. In data streams it is often allowed to present approximate solutions for the problems, since we never get to see or process the whole data. Given the huge number of clustering algorithms that use the notion of representative points to compress data, the second alternative presents a clear method to assess the quality of the clustering structure, even if only approximately.

4. Evaluation Methodology in Non-Stationary Environments

Any evaluation method should measure how the current model fits the current data.

4.1. Supervised Learning

An additional problem of the hold-out method comes from the non-stationary properties of data streams. The holdout-method using a fixed test set cannot provide reliable estimates for non-stationary streams. The main advantage of the *Prequential* becomes clear in these environments.

The prequential-error is a computed based on an accumulated sum of a loss function. We can consider two alternatives to compute the accumulated sum: *i)* from the starting of the learn process; *ii)* in a sliding window of fixed size. This latter option, seems to be more appropriate for non-stationary environments. The sliding window *forgets* past information, eventually from older contexts. The aggregated error estimate better reflects the current *fit* between the actual model and the state of the nature.

Other criteria relevant for change detection methods include:

1. Resilience to noise. That is, ability to not detect drift when there is no change in the target concept. We designate this type of errors as Type 1 error.
2. Type 2 error: does not detect drift when drift exist.
3. The number of examples required to detect a change after the occurrence of a change.

4.2. Unsupervised Learning

As previously stated, the main problem in applying machine learning to data streams is that systems should consider data evolution and adapt to new concepts. In non-stationary data streams, clustering algorithms should keep track of several aspects of the learning process:

1. What is the best partition of clusters (or the best centers) that fit the current data?
2. What is the validity of the resulting structure with respect to the recent data?
3. How did clusters and the clustering structure evolve since the last definition?

Evaluating these aspects is an overwhelming task. We have shown how clustering validity may be highly dependent of the correct definition of parameters such as the number k of clusters to find. Additionally, in dynamic environments, the real number of clusters may evolve with time, rising the difficulty to define k. Moreover, even if the number of clusters is constant, clusters' configuration may evolve with time. Methods that aim at finding clustering structures from data streams must include techniques for detection of structural drift (Rodrigues et al., 2008) or cluster change (Spiliopoulou et al., 2006).

Hierarchical techniques, which divide or aggregate clusters, are useful for structural drift detection, as they automatically define the number of clusters based on specifications or generalizations of previous clusters. Another advantage of these algorithms is that the clustering validity measure can be easily estimated from previous structure. However, they lack the ability to detect cluster evolution and transitions, which sometimes lead to the fusion of non-related clusters.

Cluster transitions are commonly measured by overlap ratios which are not possible in strict hierarchical structures. However, when considering data streams, if only representatives of real data are kept, how can overlap be accurately measured? This problem includes another issue in learning from data streams, as different granularity's can be used in the compact representation of the data points.

As in supervised learning, the use of sliding windows to assess fitness of the clustering structure to the most recent data is a viable approach if we measure validity using real data within the window. If a system

is designed to use representatives, computing validity in the sliding window offers the system with more robust estimation given the maintenance of some old representatives along with recent real data. However, cluster evolution may become harder to define.

Overall, if a specific validity index, defined as a loss function, is able to include the fitness of the clustering definition to real data, given the number of clusters found and the overlap ratio, the system could use the *Predictive Sequential* technique to this loss function. If we keep track of the values for the validity over a recent set of sliding windows, we could present a robust method to estimate not only the quality of the resulting structure, but also the evolution of the clustering structure, triggering changes in the process (e.g. division/aggregation of clusters, granularity of representatives, update or reset of the whole structure, etc.).

5. Conclusions

The main problem in the evaluation methods when learning from data streams consists of monitoring the evolution of the learning process. In this work we defend the use of *Predictive Sequential* error estimates over a sliding window to assess performance of learning algorithms that learn from open-ended data streams in non-stationary environments.

In this work we propose the prequential method as a general methodology to evaluate learning algorithms in a streaming scenario. In some applications, where feedback is available later, it can be implemented inside the learning algorithm. This open interesting opportunities: the system would be capable of monitoring the evolution of the learning process itself and *self-diagnosis* the evolution of the learning process.

In this work we only discuss *loss* as performance criteria. Nevertheless, other criteria, imposed by data streams characteristics, must be taken into account. Memory is one of the most important constrains. Learning algorithms run in fixed memory. They need to manage the available memory, eventually discarding parts of the required statistics or parts of the decision model. We need to evaluate the memory usage over time, and the impact in accuracy when using the available memory. Another aspect is that algorithms must process the examples as fast if not faster than they arrive. Whenever the rate of arrival is faster than the processing speed, same sort of sampling is required. The number of examples processed per second and the impact of sampling in performance are other evaluation criteria.

Acknowledgments

Thanks to the financial support given by the FEDER, the Plurianual support attributed to LIAAD, and project ALES II (POSC/EIA/55340/2004). The work of P.P. Rodrigues is supported by the Portuguese Foundation for Science and Technology (FCT) under the PhD Grant SFRH/BD/29219/2006.

References

Aggarwal, C. C., Han, J., Wang, J., & Yu, P. S. (2003). A framework for clustering evolving data streams. *VLDB 2003, Proceedings of 29th International Conference on Very Large Data Bases* (pp. 81–92). Morgan Kaufmann.

Babcock, B., Datar, M., Motwani, R., & O'Callaghan, L. (2003). Maintaining variance and k-medians over data stream windows. *Proc. of the 22nd Symposium on Principles of Database Systems*. ACM Press.

Beringer, J., & Hüllermeier, E. (2006). Online clustering of parallel data streams. *Data and Knowledge Engineering*, *58*, 180–204.

Cormode, G., Muthukrishnan, S., & Zhuang, W. (2007). Conquering the divide: Continuous clustering of distributed data streams. *ICDE* (pp. 1036–1045).

Dawid, A. P. (1984). Statistical theory: The prequential approach. *Journal of the Royal Statistical Society-A*, *147*, 278–292.

Domingos, P., & Hulten, G. (2000). Mining High-Speed Data Streams. *Proceedings of the ACM Sixth International Conference on Knowledge Discovery and Data Mining* (pp. 71–80). ACM Press.

Ferrer-Troyano, F., Aguilar-Ruiz, J. S., & Riquelme, J. C. (2004). Discovering decision rules from numerical data streams. *Proceedings of the 2004 ACM symposium on Applied computing* (pp. 649–653). Nicosia, Cyprus: ACM Press.

Gama, J., & Rodrigues, P. P. (2007). Stream-based electricity load forecast. *Proceedings of the 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD 2007)* (pp. 446–453). Warsaw, Poland: Springer Verlag.

Gama, J., Rocha, R., & P.Medas (2003). Accurate decision trees for mining high-speed data streams. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 523–528). Washington, D.C.: ACM Press.

Hulten, G., & Domingos, P. (2001). Catching up with the data: research issues in mining data streams. *Proc. of Workshop on Research issues in Data Mining and Knowledge Discovery*.

Hulten, G., Spencer, L., & Domingos, P. (2001). Mining time-changing data streams. *Proceedings of the 7th ACM SIGKDD International conference on Knowledge discovery and data mining* (pp. 97–106). San Francisco, California: ACM Press.

Jain, A. K., & Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall.

O'Callaghan, L., Meyerson, A., Motwani, R., Mishra, N., & Guha, S. (2002). Streaming-data algorithms for high-quality clustering. *Proceedings of the Eighteenth Annual IEEE International Conference on Data Engineering* (pp. 685–696). IEEE Computer Society.

Rodrigues, P. P., Gama, J., & Pedroso, J. P. (2008). Hierarchical clustering of time-series data streams. *IEEE Transactions on Knowledge and Data Engineering*, *20*, 615–627.

Spiliopoulou, M., Ntoutsi, I., Theodoridis, Y., & Schult, R. (2006). Monic: modeling and monitoring cluster transitions. *KDD* (pp. 706–711).

Zhang, T., Ramakrishnan, R., & Livny, M. (1996). BIRCH: An efficient data clustering method for very large databases. *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data* (pp. 103–114). ACM Press.