Neurocomputing 150 (2015) 250–264

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Recovery analysis for adaptive learning from non-stationary data streams: Experimental design and case study

Ammar Shaker *, Eyke Hüllermeier

Department of Computer Science, University of Paderborn, Germany

ARTICLE INFO

Article history:
Received 8 January 2014
Received in revised form
21 August 2014
Accepted 4 September 2014
Available online 18 October 2014

Keywords:
Machine learning
Data streams
Concept drift
Supervised learning
Regression
Classification

ABSTRACT

The extension of machine learning methods from static to dynamic environments has received increasing attention in recent years; in particular, a large number of algorithms for learning from so-called *data streams* has been developed. An important property of dynamic environments is *non-stationarity*, i.e., the assumption of an underlying data generating process that may change over time. Correspondingly, the ability to properly react to so-called *concept change* is considered as an important feature of learning algorithms. In this paper, we propose a new type of experimental analysis, called *recovery analysis*, which is aimed at assessing the ability of a learner to discover a concept change quickly, and to take appropriate measures to maintain the quality and generalization performance of the model. We develop recovery analysis for two types of supervised learning problems, namely classification and regression. Moreover, as a practical application, we make use of recovery analysis in order to compare model-based and instance-based approaches to learning on data streams.

1. Introduction

The development of methods for learning from so-called *data streams* has been a topic of active research in recent years [11,14]. Roughly speaking, the key idea is to have a system that learns incrementally, and maybe even in real-time, on a continuous and potentially unbounded stream of data, and which is able to properly adapt itself to changes of environmental conditions or properties of the data generating process. Systems with these properties have already been developed for different machine learning and data mining tasks, such as clustering and classification [13].

An extension of data mining and machine learning methods to the setting of data streams comes with a number of challenges. In particular, the standard "batch mode" of learning, in which the entire data as a whole is provided as an input to the learning algorithm (or "learner" for short), is no longer applicable. Correspondingly, the learner is not allowed to make several passes through the data set, which is commonly done by standard methods in statistics and machine learning. Instead, the data must be processed in a single pass, which implies an incremental mode of learning and model adaptation.

Domingos and Hulten [9] list a number of properties that an ideal stream mining system should exhibit, and suggest corresponding design decisions: the system uses only a limited amount of memory; the time to process a single record is short and ideally constant; the data is volatile and a single data record accessed only once; the model produced in an incremental way is equivalent to the model that would have been obtained through common batch learning (on all data records so far); the learning algorithm should react to concept change (i.e., any change of the underlying data generating process) in a proper way and maintain a model that always reflects the current concept.

This last property is often emphasized as a key feature of learning algorithms, since non-stationarity is arguably the most important difference between static and dynamic environments. Indeed, while the idea of an incremental learning is crucial in the setting of data streams, too, it is not entirely new and has been studied for learning from static data before. The ability of a learner to maintain the quality and generalization performance of the model in the presence of concept drift, on the other hand, is a property that becomes truly important when learning under changing environmental conditions.

In this paper, which is an extended version of the conference paper [28], we propose a new type of experimental analysis, called *recovery analysis*. With the help of recovery analysis, we aim at assessing a learner's ability to maintain its generalization performance in the presence of concept drift. Roughly speaking, recovery analysis suggests a specific experimental protocol and a graphical presentation of the learner's performance that provides an idea of how quickly a drift is recognized, to what extent it affects the

* Corresponding author.
E-mail addresses: ammar.shaker@upb.de (A. Shaker), eyke@upb.de (E. Hüllermeier).

http://dx.doi.org/10.1016/j.neucom.2014.09.076
0925-2312/

prediction performance, and how quickly the learner manages to adapt its model to the new condition. Our method makes use of real data, albeit in a modified and specifically prepared form, which is a main prerequisite for conducting controlled experiments under suitable conditions; therefore, it could be seen as a "semi-synthetic" approach.

Another contribution of the paper is an experimental study, which illustrate the usefulness of recovery analysis by comparing different types of learning methods with regard to their ability to handle concept drift. In particular, we turn our attention to the comparison of instance-based and model-based approaches to learning on data streams.

The remainder of the paper is organized as follows. By way of background, the next section recalls some important aspects of learning on data streams, with a specific emphasis on handling concept drift. Our method of recovery analysis is then introduced in Section 3. In Section 4, we contrast model-based with instance-based learning approaches and motivate their comparison, prior to describing the experiments and results in Section 5. We end the paper with some concluding remarks and an outlook on future work in Section 6. In Appendix A, we outline the learning methods that we included in our case study.

2. Learning under concept drift

We consider a setting in which an algorithm $\mathcal{A}$ is learning on a time-ordered stream of data $S=(\boldsymbol{z}_1,\boldsymbol{z}_2,\boldsymbol{z}_3,\ldots)$. Since we are mainly interested in *supervised learning*, we suppose that each data item $\boldsymbol{z}_t$ is a tuple $(\boldsymbol{x}_t,y_t)\in\mathbb{X}\times\mathbb{Y}$ consisting of an input $\boldsymbol{x}_t$ (typically represented as a vector) and an associated output y_t, which is the target for prediction. In classification, for example, the output space $\mathbb{Y}$ consists of a finite (and typically small) number of class labels, whereas in regression the output is a real number.

At every time point t, the algorithm $\mathcal{A}$ is supposed to offer a predictive model $\mathcal{M}_t:\mathbb{X}\to\mathbb{Y}$ that has been learned on the data seen so far, i.e., on the sequence $S_t=(\boldsymbol{z}_1,\boldsymbol{z}_2,\ldots,\boldsymbol{z}_t)$. Given a query input $\boldsymbol{x}\in\mathbb{X}$, this model can be used to produce a prediction

$$\hat{y}=\mathcal{M}_t(\boldsymbol{x})\in\mathbb{Y}$$

of the associated output. The accuracy of this prediction can be measured in terms of a loss function $\ell:\mathbb{Y}\times\mathbb{Y}\to\mathbb{R}$, such as the 0/1 loss in the case of classification or the squared error loss in regression. Then, the prediction performance of $\mathcal{M}_t$ is defined in terms of the expected loss, where the expectation is taken with respect to an underlying probability measure $\mathbf{P}$ on $\mathbb{Z}=\mathbb{X}\times\mathbb{Y}$. This probability measure formally specifies the data generating process.

If the algorithm $\mathcal{A}$ is truly incremental, it will produce $\mathcal{M}_t$ solely on the basis of $\mathcal{M}_{t-1}$ and $\boldsymbol{z}_t$, that is, $\mathcal{M}_t=\mathcal{A}(\mathcal{M}_{t-1},\boldsymbol{z}_t)$. In other words, it does not store the entire sequence of previous observations $\boldsymbol{z}_1,\ldots,\boldsymbol{z}_{t-1}$. Many algorithms, however, store at least a few of the previous data points, typically the most recent ones, which can then also be used for model adaptation. In any case, the number of observations that can be stored is normally assumed to be finite, which excludes the possibility of memorizing the entire stream. A *batch learner* $\mathcal{A}_B$, on the other hand, would produce the model $\mathcal{M}_t$ on the basis of the complete set of data $\{\boldsymbol{z}_1,\ldots,\boldsymbol{z}_t\}$. Note that, although $\mathcal{A}$ and $\mathcal{A}_B$ have seen the same data, $\mathcal{A}_B$ can exploit this data in a more flexible way. Therefore, the models produced by $\mathcal{A}$ and $\mathcal{A}_B$ will not necessarily be the same.

As mentioned before, the data generating process is characterized by the probability measure $\mathbf{P}$ on $\mathbb{Z}=\mathbb{X}\times\mathbb{Y}$. Under the assumptions of stationarity and independence, each new observation $\boldsymbol{z}_t$ is generated at random according to $\mathbf{P}$, i.e., the probability to observe a specific $\boldsymbol{z}\in\mathbb{Z}$ is given by[1]

$$\mathbf{P}(\boldsymbol{z})=\mathbf{P}(\boldsymbol{x},y)=\mathbf{P}(\boldsymbol{x})\cdot\mathbf{P}(y|\boldsymbol{x}).$$

Giving up the assumption of stationarity (while keeping the one of independence), the probability measure $\mathbf{P}$ generating the next observation may possibly change over time. Formally, we are thus dealing, not with a single measure $\mathbf{P}$, but with a sequence of measures $(\mathbf{P}_1,\mathbf{P}_2,\mathbf{P}_3,\ldots)$, assuming that $\boldsymbol{z}_t$ is generated by $\mathbf{P}_t$. One speaks of a *concept change* if these measures are not all equal [19].

In the literature, a distinction is made between different causes and types of concept change [12]. The first type refers to a sudden, abrupt change of the underlying concept to be learned and is often called *concept shift* ($\mathbf{P}_t$ is very different from $\mathbf{P}_{t-1}$). Roughly speaking, in the case of a concept shift, any knowledge about the old concept may become obsolete and the new concept has to be learned from scratch. The second type refers to a gradual evolution of the concept over time. In this scenario, old data might still be relevant, at least to some extent. Finally, one often speaks about *virtual* concept drift if the change only concerns $\mathbf{P}(\boldsymbol{x})$, i.e., the distribution of the inputs, while the concept itself, i.e., the conditional distribution $\mathbf{P}(y|\boldsymbol{x})$, remains unchanged [31]. To guarantee optimal predictive performance, an adaptation of the model might also be necessary in such cases. In practice, virtual and real concept drift will often occur simultaneously.

Learning algorithms can handle concept change in a direct or indirect way. In the indirect approach, the learner does not explicitly attempt to detect a concept drift. Instead, the use of outdated or irrelevant data is avoided from the outset. This is typically accomplished by considering only the most recent data while ignoring older observations, e.g., by sliding a window of fixed size over a data stream. To handle concept change in a more direct way, appropriate techniques for discovering the drift or shift are first of all required, for example based on statistical tests.

3. Recovery analysis

In practical studies, data streams are of course never truly infinite. Instead, a "stream" is simply a large data set in the form of a long yet finite sequence $S=(\boldsymbol{z}_1,\boldsymbol{z}_2,\ldots,\boldsymbol{z}_T)$. In experimental studies, such streams are commonly used to produce a performance curve showing the generalization performance of a model sequence $(\mathcal{M}_t)_{t=1}^{T}$ over time. Although many of these studies are interested in analyzing the ability of a learner to deal with concept drift, such an analysis is hampered by at least two problems:

- Ignorance about drift: First, for a real data stream S, it is normally not known whether it contains any concept drift, let alone when such a drift occurs.
- Missing baseline: Second, even if a concept drift is known to occur, it is often difficult to assess the performance of a learner or to judge how well it recovers after the drift, simply because a proper *baseline* is missing: The performance that could in principle be reached, or at least be expected, is not known.

Obviously, these problems are less of an issue if data is generated synthetically. In fact, for synthetic data, the "ground truth" is always known. Moreover, synthetic data has the big advantage of enabling *controlled* experiments. For example, one might be interested in how an algorithm reacts to a drift depending on certain characteristics of the drift, such as its strength and duration. While real data will (at most) contain a single drift,

[1] We slightly abuse notation by using the same symbol for the joint probability and its marginals.