A Framework for Trace Clustering and Concept-drift Detection in Event Streams

Sylvio Barbon Junior[1], Gabriel Marques Tavares[1], Paolo Ceravolo[2], and Ernesto Damiani[3]

[1] Londrina State University
barbon@uel.br | gtavares@uel.br
[2] Università degli Studi di Milano
paolo.ceravolo@unimi.it
[3] Khalifa University
ernesto.damiani@kustar.ac.ae

1 Introduction

Concept-drift is a well-known problem that affects data streams where the underlying relations between a recorded tuple $\boldsymbol{x}$ and a system response y change over time [1]. Ignoring concept-drift can lead to a deterioration in the quality of predictive analytics and its capacity to represent the most recent concepts. Nevertheless, implementing a concept-drift adaptation strategy is not a trivial task due to different types of concept-drift and different adaptations in response to them. In this work, we discuss the Concept-Drift in Event Stream Framework (CDESF) that addresses some of these challenges for Trace Clustering (TC) [2] in data streams. Instead of creating an additional level of complexity that isolates the final user from the deep behaviour of the system, our goal is to offer a simple instrument to supervise concept-drift and tracking the evolution of clusters over time.

2 Trace Clustering with Event Streams

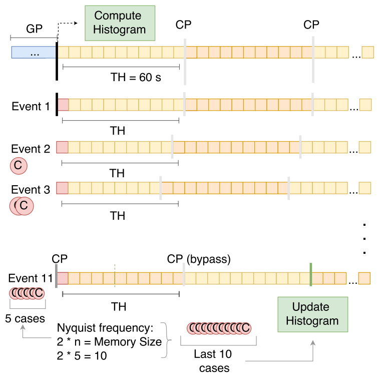
The learning task addressed is TC in event streams to run analytics on Business Processes. Traces represent sequences of activities or tasks executed to achieve a goal. Unlike other stream-related problems, in TC the single tuple imputing the learning procedure is obtained by grouping multiple events through time, with past events affecting future ones.

Model Update. Since ingestion of event data is asynchronous, we cannot rely on a training data set. For this reason, we introduced the idea of a Grace Period (GP) where there is no reference model and new events are used to feed the model construction. When the GP is declared over, the model creation is triggered. Subsequently, the model is updated based on the Nyquist sampling theorem, where the sampling frequency during data acquisition is forced to be at least twice the highest frequency observed before an update. A Time Horizon (TH), inputted by the user, specify how frequently the system has to check for a model update. However, if the conditions required by the Nyquist sampling theorem are not satisfied this TH is shifted over.

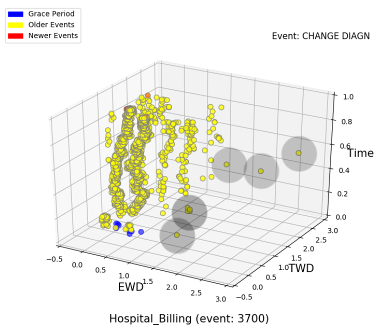
Evolution of Clusters. One of our motivating goals is to construct an human-friendly visual and inference system. For this reason, to control and observe the system behaviour over time, we selected two simple metrics, based on the histogram of events in a trace and on the histogram of events' timestamps.

When a new event is acquired it is attached to the respective trace that can be read as a string and compared to the histogram strings describing clusters. The histograms of events are compared using a Edit Weighted Distance (EWD). The histogram of timestamps is built from the same set of tuples but with some additional steps. First, for each tuple, a list of the differences between the events timestamps is created. Then, the list serves as input for quartiles calculation. Ultimately, the list values are placed into the quartile bins. Given a new event, its case timestamps are retrieved and binned. The bin is normalised and subtracted from the (also normalised) histogram. The result is the time-weighted distance (TWD) related to the interval between the events in a trace.

We can now use EWD and TWD, together with a global time, as parameters describing the evolution of traces and clusters in the feature space. Concept-drift is detected in the feature space (EWD, TWD and time) by looking at distribution sparsity. When a new sample falls outside the boundary of any existing cluster, it is marked as an anomaly and its density is monitored. An increase in the number of samples within the radius of an anomalous example indicates a concept-drift.



(a) Model Update



(b) Evolution of Clusters

3 Expected Results

We have observed the ability of our framework to effectively detect the anomalies and drift detection over the stream with different TH. Even more, an unfinished process case could be observed closely from the beginning leading to early identification of anomalous pattern. In this way, it is possible to mitigate a costly error, resist an attack before it reaches its goal, halt or migrate the fraudulent execution to a honeypot.

References

1. Gama, J., Rodrigues, P.P., Spinosa, E., Carvalho, A.: Knowledge Discovery from Data Streams. Web Intelligence and Security - Advances in Data and Text Mining Techniques for Detecting and Preventing Terrorist Activities on the Web pp. 125–138 (2010), `http://www.booksonline.iospress.nl/Content/View.aspx?piid=18418`
2. Song, M., Günther, C.W., Van der Aalst, W.M.: Trace clustering in process mining. In: International Conference on Business Process Management. pp. 109–120. Springer (2008)