Using Recurrent Boltzmann Machines in the Detection of Process Drifts Using Several Process Mining Perspectives

Alessandro Berti

SIAV
35030 Rubano PD
Email: alessandro.berti89@gmail.com

Abstract—**In this paper is proposed an algorithm that uses Recurrent Boltzmann Machines to detect concept drifts in Process Mining event logs. The method is able to use several perspectives (Control Flow, Resources, Data) in comparison to existing methods that are conceived to use a single perspective (Control Flow). The approach has been tested on some artificial event logs and on a real-life log.**

Keywords–Concept Drift; Process Mining; Boltzmann Machines.

I. Introduction and background

Process Mining [1] is a relatively new discipline related to the discovery and the analysis of business processes starting from event logs. These logs are organised in traces, that correspond to single executions of the process (a process instance). Each trace can contain several events, that can be described by some attributes, like the organizational resource that has performed the event, the activity that has been performed and the timestamp. In most cases, events are instantaneous (so, only the completion time of an activity is recorded), and a trace could be briefly described (the Control-Flow perspective) by a list of activities. A path is a succession of activities that can be observed in traces. For example, if a trace can be described by this list of activities: A,B,C,D,E; then the paths are AB, BC, CD, DE. The start timestamp of a trace is the minimum of the timestamps of its events; the end timestamp is the maximum of the timestamps.

Process Mining algorithms are often hampered by concept drifts in the underlying process, that are changes in the process during the analyzed time interval. Some papers (like [2], [3], [4]) have analyzed a way to cope with the drift in processes, but the analysis is restricted to the Control-Flow perspective, and ignore other information related to the process instance. The Control Flow perspective is the list of activities performed in order to complete a single business process instance. Other perspectives are the data perspective and the resource perspective (people that are involved in the completion of the instance). In this paper is described a way to detect the drift in the underlying process that takes into account also the other information related to business instances. The method is based on the estimation, for each of the business instances, of the probability that a concept drift has actually happened. The algorithm is based on Recurrent Boltzmann Machines [5] that are useful to model high dimensional sequences. Other methods (like NADE [6]) are known to be able to detect the probability to observe a given point; however, their application is usually done point-by-point and ignores the the history of the observed points (that are business instances). The proposed approach is a necessary extension of the methods that take in account only the Control Flow because they cannot detect changes happening in other perspectives. In the Background Section of this paper, Boltzmann Machines are presented. In the Method Section, the proposed approach to detect the concept drift is analysed. In the Results Section, some results related to artificial and real event logs are presented.

II. Background

Restricted Boltzmann Machines (RBM) are useful to learn a probability distribution over a set of inputs and are based on the concepts of visible and hidden binary units. Hidden units are activated by the RBM, that works taking in input a weighted sum of the visible units, applying a $[0,1]$-valued function and activating the hidden unit with a probability equal to the function value. An energy function can be associated with RBM, that is based on the visible and hidden units value; a low energy configuration is preferred as the definition of the energy function makes RBM useful for classification purposes [7]. A well-known method for RBM training is Contrastive Divergence [8]; basically, it is an iterative method for RBM weights discovery that tries to minimise the difference between the visible units and some temporary (binary) units whose value is found by the inverse application of the RBM (in this step, the hidden units become the visible units). RBM, however, do not handle sequence of points as the hidden units' activation depend only on the current iteration of visible units (that are single points), and do not handle history. Recurrent Boltzmann Machines are conceived to use the history of the sequence, as the hidden units activation do not depend only on the visible units but also on the previous states of the hidden units. However, other papers (like [5]) can be relevant for further information.

III. Method

The method is based on the construction of a sequence of binary points (each one corresponding to an event log trace) that is provided to a Recurrent Boltzmann Machine in order to learn a meaningful representation of the sequence. The number of hidden units has been set to be equal to the number of visible units. The trace is being described in both the Control-Flow perspective and the other perspectives:

1) The Control-Flow is described by the paths followed in the trace.
2) Other perspectives are described by recording all the different values for an attribute that can be seen in the various events of the trace.

A binary representation, whose length is equal to the sum of the number of different paths in all the traces of the log and the number of different values for the considered attributes in all the events of the log, can be obtained by giving value 1 in a position that describes a path / attribute value that is contained in the trace, and giving 0 otherwise. For example, if there are the following two traces: Trace 1 (events: A(Mike),B(Tom),C(Mike); paths: AB, BC; resources: Mike, Tom), Trace 2 (events: A(Alex),D(Maria),E(Maria); paths: AD, DE; resources: Alex, Maria); the binary representation could be as follow: position 1 is relative to the presence of the path AB, pos. 2 is relative to BC, pos. 3 is relative to AD, pos. 4 is relative to DE, pos. 5 is relative to the resource Mike, pos. 6 is relative to Tom, pos. 7 is relative to Alex, pos. 8 is relative to Maria; the eventual representation is (1,1,0,0,1,1,0,0) for Trace 1 and (0,0,1,1,0,0,1,1) for Trace 2. The traces are considered to be ordered by their start timestamp.

After building the sequence of binary points, the RBM could be trained. The results of the training are then used by “stopping before” the activation of the hidden units and recording the activation function (probability) values. So, a $[0, 1]$-valued vector can be obtained for each trace, that refers to the probability of activation of the hidden units. From these vectors, the maximum value is taken; in doing so, each trace is described by a single probability value. Traces with a lower value of probability are likely to show a concept drift in the underlying process. This is because the RBM tries to learn a representation that maximises the probability of a given sequence, and at least one hidden unit for trace should be activated with high probability. Considering also the previous states of the sequence, a trace that follows the previous schema shows usually a high value in the defined probability, while traces that show a different schema produce lower values of probability.

IV. Results

The following artificial logs have been used in order to evaluate the effectiveness of the method:

1) An event log that contains 1000 equal traces (events: A(Mike),B(Tom),C(Mike); paths: AB, BC; resources: Mike, Tom), and other 1000 equal traces (events: A(Alex),D(Maria),E(Maria); paths: AD, DE; resources: Alex, Maria).
2) An event log that contains 500 traces for each of the following schemas:
    - events: A(Mike),B(Tom),C(Mike); paths: AB, BC; resources: Mike, Tom
    - events: A(Alex),D(Maria),E(Maria); paths: AD, DE; resources: Alex, Maria
    - events: F(Billie),G(Louise),H(Billie); paths: FG, GH; resources: Billie, Louise
    - events: I(Barack),L(Francois),M(Barack); paths: IL, LM; resources: Barack, Francois
3) An event log that contains 500 traces for each of the following schemas:
    - events: A(Mike),B(Tom),C(Mike); paths: AB, BC; resources: Mike, Tom
    - events: A(Alex),B(Maria),C(Maria); paths: AB, BC; resources: Alex, Maria
    - events: F(Billie),G(Louise),H(Billie); paths: FG, GH; resources: Billie, Louise
    - events: F(Barack),G(Francois),H(Barack); paths: FG, GH; resources: Barack, Francois

The method is able to correctly identify concept drifts in all the cases. The first two logs are very simple, as there is a drift both in the Control Flow and the resources. The third log shows changes in the resource set; existing methods for concept drift in Process Mining (as [2], [3], [4]) would not have been able to identify changes in this log.

The method has been tested also on a real-life event log, that is “Receipt phase of an environmental permit application process” [9] containing an interesting shift in the process. Using Dotted Chart feature in ProM framework [10] you can identify a change in the underlying process after the timestamp 31/03/2011, when some activities (T06 and T10) became slightly less frequent. This shift has not been identified by the method described in [2] as the change in the Control Flow is not so great, but is identified by the proposed method taking into account the other perspectives. Also the methods described in [3] based on SVM, and [4] which is an improvement of [2], fail to identify this change.

V. Conclusion and Future Work

The proposed approach for the detection of concept drifts takes into account several Process Mining perspectives and correctly identifies process changes in the examined logs. Further work is needed to classify the change points, as there can be several drifts (reported in [2]): sudden drifts, gradual drifts, seasonal drifts; also, other types of hidden units (as [11]) might produce better results.

References

[1] W. Van Der Aalst, Process mining: discovery, conformance and enhancement of business processes. Springer Science & Business Media, 2011.

[2] R. J. C. Bose, W. M. van der Aalst, I. Žliobaitė, and M. Pechenizkiy, “Handling concept drift in process mining,” in Advanced Information Systems Engineering. Springer, 2011, pp. 391–405.

[3] R. Klinkenberg and T. Joachims, “Detecting concept drift with support vector machines.” in ICML, 2000, pp. 487–494.

[4] D. Luengo and M. Sepúlveda, “Applying clustering in process mining to find different versions of a business process that changes over time,” in Business Process Management Workshops. Springer, 2011, pp. 153–158.

[5] I. Sutskever, G. E. Hinton, and G. W. Taylor, “The recurrent temporal restricted boltzmann machine,” in Advances in Neural Information Processing Systems, 2009, pp. 1601–1608.

[6] H. Larochelle and I. Murray, “The neural autoregressive distribution estimator,” in International Conference on Artificial Intelligence and Statistics, 2011, pp. 29–37.

[7] H. Larochelle and Y. Bengio, “Classification using discriminative restricted boltzmann machines,” in Proceedings of the 25th international conference on Machine learning. ACM, 2008, pp. 536–543.

[8] M. A. Carreira-Perpinan and G. E. Hinton, “On contrastive divergence learning,” in Proceedings of the tenth international workshop on artificial intelligence and statistics. Citeseer, 2005, pp. 33–40.

[9] J. Buijs, “Receipt phase of an environmental permit application process (wabo), coselog project,” 2014. [Online]. Available: http://dx.doi.org/10.4121/uuid:a07386a5-7be3-4367-9535-70bc9e77dbe6

[10] B. F. van Dongen, A. K. A. de Medeiros, H. Verbeek, A. Weijters, and W. M. Van Der Aalst, “The prom framework: A new era in process mining tool support,” in Applications and Theory of Petri Nets 2005. Springer, 2005, pp. 444–454.

[11] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in Proceedings of the 27th International Conference on Machine Learning (ICML-10), 2010, pp. 807–814.