Online Unsupervised State Recognition in Sensor Data (Supplementary Materials)

Julien Eberle, Tri Kurniawan Wijaya, and Karl Aberer
School of Computer and Communication Sciences
École Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
Email: {julien.eberle, tri-kurniawan.wijaya, karl.aberer}@epfl.ch

***Abstract*—Smart sensors, such as smart meters or smart phones, are nowadays ubiquitous. To be "smart", however, they need to process their input data with limited storage and computational resources. In this paper, we convert the stream of sensor data into a stream of symbols, and further, to higher level symbols in such a way that common analytical tasks such as anomaly detection, forecasting or state recognition, can still be carried out on the transformed data with almost no loss of accuracy, and using far fewer resources. We identify states of a monitored system and convert them into symbols (thus, reducing data size), while keeping "interesting" events, such as anomalies or transition between states, as it is. Our algorithm is able to find states of various length in an online and unsupervised way, which is crucial since behavior of the system is not known beforehand. We show the effectiveness of our approach using real-world datasets and various application scenarios.**

This document contains the supplementary material of our paper presented at PerCom 2015 [1].

I. Reverting States

One of the goals of Spclust and StateFinder is to produce a symbolic time series to support higher level applications, without converting it back to sensor's original measurement values. Since symbolic time series is much shorter than its original version, this property is desirable, especially due to the limited sensor's storage and computational power. Thus, we do not discuess the process of reverting symbols back to its original valus in the main paper. However, for the sake of completeness, below we illustrate how one can revert symbolic time series to its original values.

Converting symbols level 0 generated by Spclust to its original values In this case, we could simply use the cluster centroids to approximate the original values of the symbols. Note that, using Spclust, we have one-to-one mapping between clusters and symbols.

Converting RLE compressed triples to non-compressed triples. Each symbol is repeated n times with $n = (t_e - t_s)/r$, where r is the sampling rate of the original data. As sensors may have a variable sampling rate or some gaps, this transformation could produce more/less triples that the original ones.

Converting symbols from level 1 or higher to one level lower. The main idea, is to use the Segment Transition Matrix To find the starting symbol in the case this one is not available, we use an heuristic that aims to find a symbol that has the lowest incoming transition probability and a positive outgoing transition probability. Formally, we take the symbol i such as the sum of the elements in the ith row, except the diagonal is greater than 0 and the sum of the elements on the ith column, except the diagonal, is minimal. Then according to the transition probability to the next symbol, we build a sequence. Even though we might produce a slightly different sequence from the original one, it eventually has the same symbol distribution.

II. State Forecasting

A. State Forecasting

The algorithm is inspired by the pattern-based forecasting in [2], [3], [4]. It consists of three steps: clustering, pattern similarity search, and prediction. Below we give an overview of the algorithm. Let us assume that we have a symbolic time series $\widehat{S} = \{\widehat{s}_1, \ldots, \widehat{s}_t\}$ available as the training set, a forecast horizon h, and window length w. Our goal is to forecast $\widehat{S}^* = \{\widehat{s}_{t+1}, \ldots, \widehat{s}_{t+h}\}$, i.e., sensor values up to h time periods following $\widehat{S}$.

Clustering Divide $\widehat{S}$ into a set of symbolic time series, $\mathcal{S} = [\widehat{S}_1, \ldots \widehat{S}_l]$, where each $\widehat{S}_i \in \mathcal{S}$ has length h, and if $i < j$, then $\widehat{S}_i$ is a series preceding $\widehat{S}_j$. Cluster the symbolic time series in $\mathcal{S}$, and let $c(\widehat{S}_i)$ be the cluster label of $\widehat{S}_i$.

Pattern similarity search Find all *matching* sequences of length w, $\{\widehat{S}_i, \ldots, \widehat{S}_{i+w-1}\}$, where $i \leq l - w$ and $[c(\widehat{S}_i), \ldots, c(\widehat{S}_{i+w-1})] = [c(\widehat{S}_{l-w+1}), \ldots, c(\widehat{S}_l)]$.

Prediction Let $\{\widehat{S}_{i_1+w}, \ldots, \widehat{S}_{i_k+w}\}$ be the *predictor set*, i.e., the set of series following the matching sequences. The predicted series, $\widehat{S}^*$, is obtained by aggregating the predictor set.

For the clustering step, we used the KMedoids algorithm. Note that, any other unsupervised learning techniques can also be used here. As in other unsupervised learning techniques, however, given different parameter settings, we are often uncertain which setting delivers the best cluster configuration. This holds even if the parameters are very intuitive, such as k in KMedoids, which is the number of clusters to be created. To select the best configuration, we use similar techniques to [3], [5]. We perform the clustering step several times using different configurations, i.e. different k, and evaluate the resulting configurations using the Silhouette, Dunn and Davies-Bouldin indices. The best cluster configuration is chosen by majority voting over the three indices.

For this experiment, we use GPS traces from the Nokia Lausanne Data Collection Campaign dataset [6]. The ten users

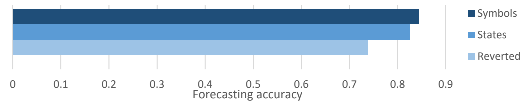

Fig. 1: Comparison of the 1-hour forecasting accuracy on top of both, symbols (level 0) and states.

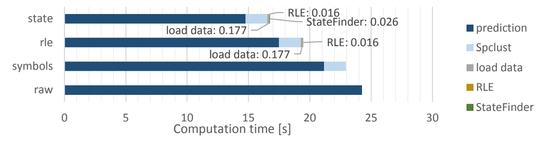

Fig. 2: Timing for the pattern-based forecasting algorithm, on top of different data, and the processing steps for generating those data.

having the largest number of location records during the first 6 months of the study were selected to minimize the gaps in the resulting time series and to avoid the users which were not carrying the phone with them all the time. In this experiment, we aim for next-hour forecasting, i.e. h is 1 hour. We compute the prediction result, $\widehat{S}^*$, as the medoid of the predictor set.

The symbolization is achieved through Spclust and results in 9 to 34 symbols, depending on the mobility of the users. The first level symbols mostly represents the places visited by the user, dividing the regions like a Voronoi diagram. On average, 24% of the symbols level 0 are converted to symbols level 1 (states), which can be seen as the routes taken by the users in their daily routines. Pattern-based forecasting is evaluated by taking the first 80% of the dataset as a training set (from 0 to t) and testing it on the next hour ($t+1$). Then we repeat the experiment with forecasting $t+2$ after including $t+1$ to the training set. We run the forecasting algorithm at different symbolization levels and for each of them, the forecasting accuracies are averaged over all users and iterations. Figure 1 shows the accuracy of forecasting before and after running StateFinder. Using the time series of states (symbols level 1) allows making predictions that are similarly good as with the symbols (level 0). showing that the states are consistently found and replaced by higher level symbols. As the reverting of the sates (using the Segment Transition Matrix) introduces also some error, the third bar illustrates the accuracy when comparing the reverted predicted values to the original time series, but this result highly depends on the reverting function which still has room for improvement.

The running time for the forecasting algorithm, shown on Figure 2, is strongly related to the data size, thus giving advantage to more compressed data representation (higher level symbols). Additionally, the time for computing RLE and the StateFinder are orders of magnitude lower, making them good candidates to be run on low power devices.

III. Running Time

To asses the impact of the time series size and show that StateFinder can be run on infinite stream. We measured the time needed to process time series of various lengths on a server with a CPU @ 2.30GHz. We used the same datasets as for the forecasting experiment (GPS traces) and the microwave power from the state identification experiment (in the main paper). The third one is a random time series where symbols (between 0 and 7) and their duration (between 1 and 1000) where chosen uniformly at random. The Figure 3 shows the processing time of StateFinder with regard to the size of the time series. Too short time series were repeated to match the length of the others. It appears clearly that the processing time grows linearly with the data size, thus having a constant processing time per element. The different slopes can be explained by the number of symbols and the randomness of the data. The more different symbols are present in the time series, the more time it will take to process, but the more predictable and repetitive, the faster StateFinder is. In this case, with only 3 different symbols and a lot of identifiable states the microwave power time series allows a very fast processing, below 0.1 seconds for 50'000 values. Whereas a completely random time series with 8 symbols is even slower than the GPS trace time series with 64 symbols.

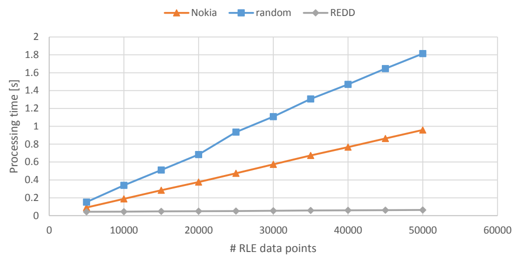

Fig. 3: Processing time for running StateFinder on different datasets and dataset sizes.

References

[1] J. Eberle, T. K. Wijaya, and K. Aberer, "Online unsupervised state recognition in sensor data," in *2015 IEEE International Conference on Pervasive Computing and Communications (PerCom) (PerCom 2015)*, St. Louis, USA, Mar. 2015.

[2] B. Motnikar, D. Čepar, P. Žunko, M. Ribarič, and B. Vovk, "Time series forecasting by imitation of preceding patterns," in *Operations Research'91*, P. Gritzmann, R. Hettich, R. Horst, and E. Sachs, Eds. Physica-Verlag HD, 1992, pp. 272–275.

[3] F. Martinez Alvarez, A. Troncoso, J. Riquelme, and J. Aguilar Ruiz, "Energy time series forecasting based on pattern sequence similarity," *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 8, pp. 1230–1243, 2011.

[4] W. Shen, V. Babushkin, Z. Aung, and W. L. Woon, "An ensemble model for day-ahead electricity demand time series forecasting," in *Proceedings of the fourth International Conference on Future Energy Systems*, ser. e-Energy '13. New York, USA: ACM, 2013, pp. 51–62.

[5] T. K. Wijaya, T. Ganu, D. Chakraborty, K. Aberer, and D. P. Seetharam, "Consumer segmentation and knowledge extraction from smart meter and survey data," in *SIAM International Conference on Data Mining (SDM14)*, 2014.

[6] J. K. Laurila, D. Gatica-Perez, I. Aad, B. J., O. Bornet, T.-M.-T. Do, O. Dousse, J. Eberle, and M. Miettinen, "The Mobile Data Challenge: Big Data for Mobile Computing Research," in *Pervasive Computing*, 2012.