APPENDIX
PSEUDo: Interactive Pattern Search in Multivariate Time Series with Locality-Sensitive Hashing and Relevance Feedback

Y. Yu[1,2], D. Kruyff[1], J. Jiao[1,3], T. Becker[2] and M. Behrisch[1]

[1]Utrecht University, Netherlands
[2]IAV GmbH Ingenieurgesellschaft Auto und Verkehr, Germany
[3]Fraunhofer Institute for Systems and Innovation, Germany

Abstract

We present PSEUDo, a visual pattern retrieval tool for multivariate time series. It aims to overcome the uneconomic (re-)training with deep learning-based methods. Very high-dimensional time series emerge on an unprecedented scale due to increasing sensor usage and data storage. Visual pattern search is one of the most frequent tasks on such data. Automatic pattern retrieval methods often suffer from inefficient training, a lack of ground truth, and a discrepancy between the similarity perceived by the algorithm and the user. Our proposal is based on a query-aware locality-sensitive hashing technique to create a representation of multivariate time series windows. It features sub-linear training and inference time with respect to data dimensions. This performance gain allows an instantaneous relevance-feedback-driven adaption and converges to users' similarity notion. We are benchmarking PSEUDo in accuracy and speed with representative and state-of-the-art methods, evaluating its steerability through simulated user behavior, and designing expert studies to test PSEUDo's usability.

CCS Concepts

• ***Mathematics of computing*** → *Time series analysis;* • ***Information systems*** → *Users and interactive retrieval;*

Contents

Appendix A: User Interface

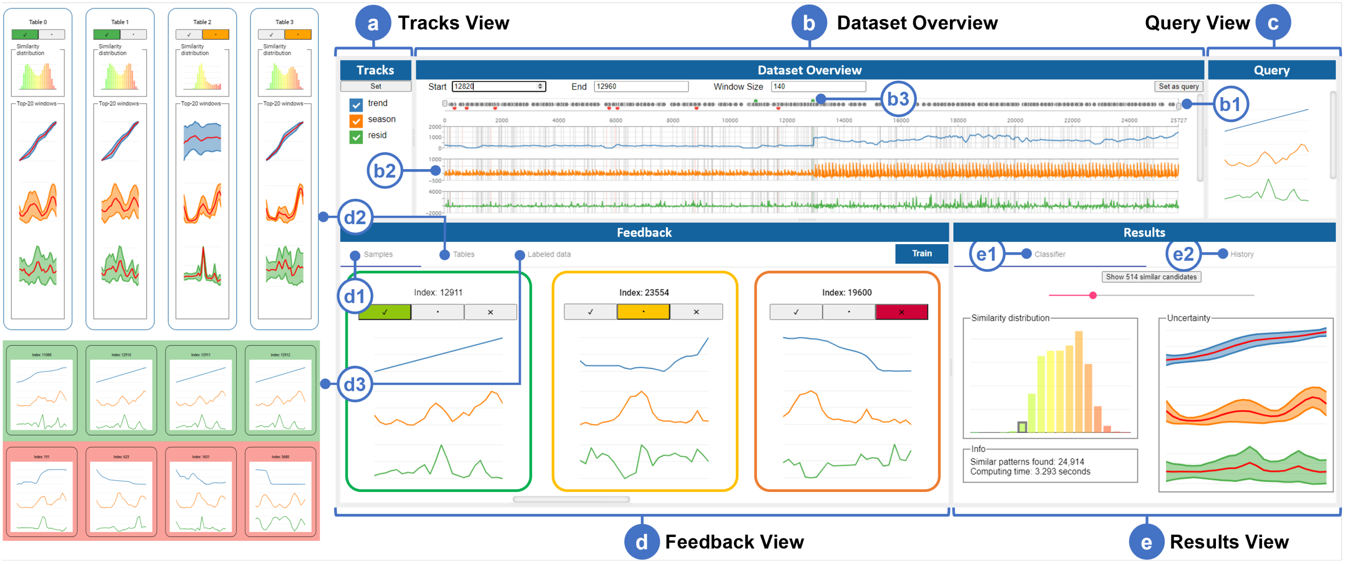

Figure A.1: ***Interface Overview:*** *We plot the selected tracks from the Tracks View ⓐ in the Dataset Overview ⓑ along with the user-defined query in ⓒ. The Feedback View ⓓ depicts classification result samples and information about the classifiers and is used to receive relevant feedback. The Results View ⓔ displays the result distribution and provides search history management.*

To facilitate data exploration, search process control, and result inspection, we designed and implemented a prototype of a visual interface depicted in Figure A.1, consisting of five interlinked views/components. The web-based user interface (mainly Angular) is connected to the backend algorithm (Python and C++) through a REST API. The code is available under `https://git.science.uu.nl/vig/sublinear-algorithms-for-va/pseudo`.

Dataset Overview, Tracks View, and Query View.

The *Dataset Overview*, *Tracks View*, and *Query View* collaborate closely. The *Dataset Overview* (Figure A.1.b) plots tracks selected in the *Tracks View* (Figure A.1.a) together with the window labels (Figure A.1.b3, dots in the range slider and area in the line chart). Besides direct panning and zooming, the range slider (Figure A.1.b1) above the track curves (Figure A.1.b2) serves as a mini-map for navigation and overview of predictions and labels.

The *Query View* (Figure A.1.c) shows the user-defined multi-track query. Our query is defined in a query-by-example manner by selecting a region in the *Dataset Overview*. The user is allowed to change the query on the fly at any time, but we have to repeat the hashing process whenever the *query size* changes. Typically users have to wait several seconds for Locality-Sensitive Hashing (LSH) parameter estimation.

Feedback View

The *Feedback View* (Figure A.1.c) shows representatives of the classified result, visualizes hash tables, and summaries labeled data. We differentiate three respective tabs for different purposes:

The *Samples Tab* (Figure A.1.d1) lists representatives or samples of the classified windows. The windows are surrounded by frames color-encoded from green to red, indicating decreasing predicted similarity. Right above the curves, the user is invited to label the window by clicking ✓ for similar, ● for indecisive and ✖ for dissimilar, in order to modify the sample relevance.

PSEUDo visualizes hash tables in the *Tables Tab* (Figure A.1.d2). The hash tables are visualized as histograms showing the similarity distribution perceived by each hash function. Each bin corresponds to a bucket in the corresponding hash table. The bars in the histograms are color-encoded from green to red, indicating decreasing similarity to the query. To better understand patterns perceived as similar by the hash functions, we plot the mean, minimum, and maximum values of each time step among the top-20 similar windows for each hash function. The mean values portray the pattern shape perceived as similar by the hash function, whereas the minimum and maximum values form a pair of lower and upper bound approximation of the pattern variance. The tightness of the band between the bounds implies the (un-)certainty or importance of the track during classification. Based on this visual encoding, the user can modify the hash functions' importance by clicking ✓ for important and ● for indecisive.

The labeled windows for the current round are kept in the *Labeled Data Tab* (Figure A.1.d3). Users can revise decisions before clicking the Train button. The labeled sample windows and hash tables will be considered in the next training round.

Results View

The *Results View* (Figure A.1.e) shows the outcome statistics and provides a git-like version management tool.

In the *Classifier Tab* (Figure A.1.e1), we use a result histogram to visualize the distribution of similarity between the query and all windows that survive LSH's pruning. Clicking a bin in the histogram shows the reconstructed pattern, which is analogous to the curves under the Tables Tab in the *Feedback View* (Figure A.1.d2). Rather than using the top-20, this pattern result view summarizes all windows in the chosen bin. The mean curves also show the average form within this bin, bounded by each time step's minimum and maximum values to illustrate the variance within tracks. The histogram and the reconstructed shape help the user better understand the classification result and provide the user guidance on the strictness during the labeling process Besides, users can set the number of top candidates and display them in the data overview.

Due to user mislabels or inappropriate parameter setting, the result quality may suffer. In this case, the user can go back to previous states through the tree plot under the History Tab (Figure A.1.e2 and Figure A.2).

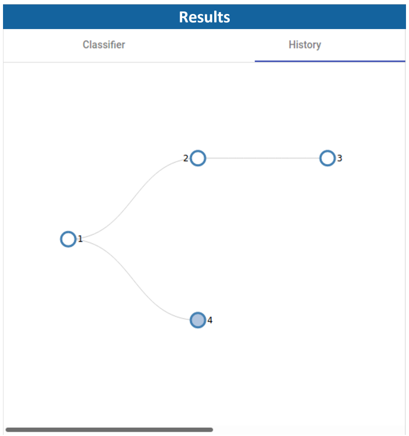

Figure A.2: *History Tab in the Result View.*

Appendix B: Resources

- Video: https://www.youtube.com/watch?v=oJfXoDyZRPY
- Code: https://git.science.uu.nl/vig/sublinear-algorithms-for-va/pseudo