Cohortney: the framework for clustering of event sequences

Taras Khakhulin [*][1] **Mikhail Pautov** [*][1]

Abstract

This report emphasises the results of the project on developing a framework for methods of clustering of event sequences. Here we formulate the problem statement and possible applications of our solution to it. Also in this report, we outline the description of methods, datasets, project structure and evaluate the results. At the end, we emphasise the contribution of team members and possible directions for improvement. Based on repository[1]

1. Problem statement

The problem of clustering of event sequences may be formulated as follows. Suppose we have a set of event sequences $S = \{s_n\}_{n=1}^N$, where $s_n = \{(t_i, c_i)\}_{i=1}^{M_n}$ contains a series of events $c_i \in C = \{1, ..., C\}$ and their time stamps $t_i \in [0, T_n]$. The goal is to cluster set S by splitting it onto an optimal (in some sense) number of non-intersecting subsets in which there are similar (in some sense) event sequences. It is supposed that the characterization of point processes is done via intensity function $\lambda_c(t) = \mathbb{E}\frac{d}{dt}N_c(t)|\mathcal{H}_t^C$, where $N_c(t)$ is the number of type-c events occurred before moment t and $\mathcal{H}_t^C = \{(t_i, c_i)|t_i < t, c_i \in C\}$.

2. Description of the project

Our project is the framework with methods solving problem of event sequences clustering. As a framework, it consists of different modules:

- *configs* – configuration files for all the codebase, callbacks, networks, data loader, trainer, etc;
- *src* – codebase of models, trainers, aux utils, etc;
- *requirements* – the list of necessary packages;
- *run, demo* – files for code execution and demonstration.

Overall scheme of the framework is depicted in the figure below.

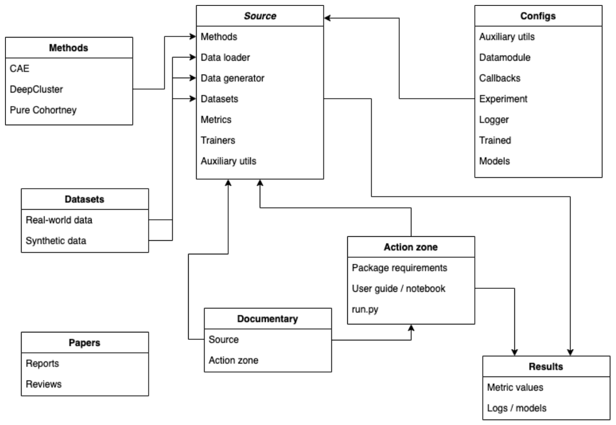


Figure 1. The scheme of the project.

2.1. Usage

The structure of our project is as follows: first user of the framework should specify train module (one of the predefined) and the dataset (synthetic or real). More formally, the following things have to be specified.

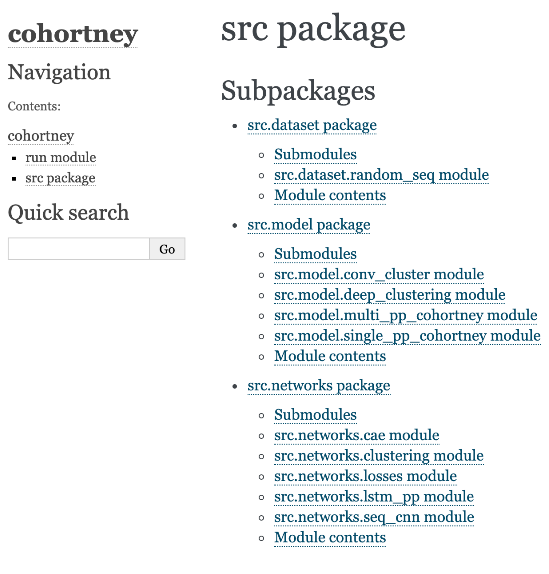
- parameter ”data_dir” in /configs/config.yaml responsible for the dataset to be downloaded and preprocessed (based on in /blob/main/src/utils/__init__.py);
- hyperparameters of corresponding method in /configs/aux_module/;
- training parameters (CPU/GPU) in /configs/trainer/default.yaml as well as NN-related training parameter (number of epochs).

2.2. Documentation

Our project provides illustrative documentation for users regarding every implemented module:

[*]Equal contribution [1]Skolkovo Institute of Science and Technology, Moscow Region, Russia. Correspondence to: Taras, Misha <no-email-here>.

[1]https://github.com/adasegroup/cohortney/



cohortney

Navigation

Contents:

cohortney

- run module
- src package

Quick search

Go

src package

Subpackages

- src.dataset package
  - Submodules
  - src.dataset.random_seq module
  - Module contents
- src.model package
  - Submodules
  - src.model.conv_cluster module
  - src.model.deep_clustering module
  - src.model.multi_pp_cohortney module
  - src.model.single_pp_cohortney module
  - Module contents
- src.networks package
  - Submodules
  - src.networks.cae module
  - src.networks.clustering module
  - src.networks.losses module
  - src.networks.lstm_pp module
  - src.networks.seq_cnn module
  - Module contents

Figure 2. Documentation of the project.

2.3. Description of methods

Two main methods are based on the top of most simple approaches in unsupervised learning with deep networks. The third method see on the cohortney from the other side and performs clustering based on the estimation of the process parameters via the EM algorithm. In this subsection we give the overview of all methods.

CAE. The clustering of representation/sequence/featrues with convolutional autoencoders is well-known and widely used method offered in [Guo+17]. In the original paper [Guo+17] the clustering process was performed with 2d convolutions. Our case is different and here we apply 1d convolutions over cohortney estimated over event sequences. The original autoencoder minimize MSE distance between a decoded with g bottleneck representation of x and x.

The structure of the convolutional autoencoder is straightforward: it consists of 5 1D convolutions with Batch Normalization in between and borrowed according with the task from the original implementation[2] We used the default setting for torch convolutional layer: stride equals to 1 and padding equals 0. Kernel size was set to 3. In order to cluster the encoder output we used standard `sklearn` library with `kmeans` algorithm.

Deep Clustering. Given partitions obtained from Cohortney for a particular n we may proceed with clustering algorithms suitable for sequences of features of the same length. Therefore we use combination Cohortney with a deep learning approaches for object clustering, such method is called Deep Clustering [**deep-cluster**]. The idea of this approach is the following: we could use deep representation to create pseudo-labels and then use them for supervised loss, which update the representation networks itself. We denote by f_θ the convnet mapping, where θ is the set of corresponding parameters. We refer $f_\theta(x)$ to the vector obtained by applying this mapping to a sequence. Given a training set $X = \{x_1, x_2, \ldots, x_N\}$ of N sequences (in cohortney partition representation f_w), we want to find a parameter θ^* such that the mapping f_{θ^*} produces good general-purpose features. These parameters are traditionally learned with supervision, i.e. each sequence x_n is associated with a label y_n in $\{0,1\}^k$. This label represents the image's membership to one of k possible predefined classes. A parametrized classifier g_W predicts the correct labels on top of the features $f_\theta\left(x_n\right)$. The parameters W of the classifier and the parameter θ of the mapping are then jointly learned by optimizing MSE.

In our implementation the training consists of the following steps: each epoch of the training phase, deep features of sequences are computed with convolutional network, then we cluster it with KMeans algorithm. The next step is update network with supervised signal from the cross-entropy loss based on the pseudo-labels. We use `sklearn` implementation of KMeans algorithm with `kmeans++` initialization. We standardize features with `sklearn` realization of `StandardScaler` before clustering. For most dataset method converged without any problem.

Multiprocess COHORTNEY. This is the real-world method presented in work [Zhu+21]. The main idea of this method: is clustering based on the LSTM Model We have K clusters and N realizations of event sequences. Given these training sequences, we want to find the clusters among them and the corresponding intensity functions. Namely, the method works as follows:

- **Data structure:** given an example of sequence as sequence of pairs of $\{t, c\}$, where t is the time moment of event occurrence, and $c \in \{1, 2, ..., C\}$ is the type of this event, the data is represented as N examples of event sequences;
- **Data processing:** given the time length T_s of sequence s, it is divided onto n_{steps} intervals with $\Delta t = \frac{T_s}{n_{\text{steps}}}$;
- **Partitions:** for a sequence s, the so-called partition is obtained – the matrix $P \in \mathbb{R}_{0+}^{C\times n_{\text{steps}}}$ with entries corresponding to occurrences of events of all types in all the intervals;
- **LSTM step:** in order to obtain intensity function $\lambda_c = f_c(w_c^T h)$, where the activation function has the form $f_c(x) = b_c \log(1 + exp(x/b_c))$, LSTM [GSC99]

[2] https://github.com/rodrigorivera/mds20_cohortney/

network is applied. The processing block is the same for all elements. The input to this block for the t-th element consists of a hidden state, $h_{t-1} \in \mathbb{R}^d$, with memory of past events and new information, x_t from s, available at time t. The output of the LSTM block, $h_t = g(x_t, h_{t-1})$, is a new hidden state. After the LSTM block, 1D Batch Normalization, [IS15], is used. The matrix of intensities, $\Lambda \in \mathbb{R}_{0+}^{C \times n_{\text{steps}}}$ is obtained after sequential application of LSTM: $\Lambda = LSTM(P(s, \Delta t), h_0)$;

- **Clustering over LSTM:** given K clusters and N sequences, the goal is to find clusters and corresponding intensity functions. For each cluster k, the hidden states, h_{ik}, change during an event sequence with starting points corresponding to the initial cluster hidden state, h_{0k}. Given a mixture model with mixing probabilities for clusters $\pi_k \geq 0$, $\sum_{k=1}^{K} \pi_k = 1$ and $P_n = P(s_n, \Delta t)$, the probabilities of cluster labels are given by

$$p(z_k = 1|P_n) = \frac{\pi_k \mathbb{P}(P_n|h_{0k})}{\sum_{j=1}^{K} \pi_j \mathbb{P}(P_n|h_{0j})},$$

  where z is the latent variable corresponding to a particular sequence ($z_k = 1$ if a sequence corresponds to k-th cluster and 0 otherwise.)

- **EM step:** since the exact way to compute probabilities above is unstable, the EM algorithm, [DLR77], equipped with realisations of probabilities is used to obtain the expected values of probabilities.

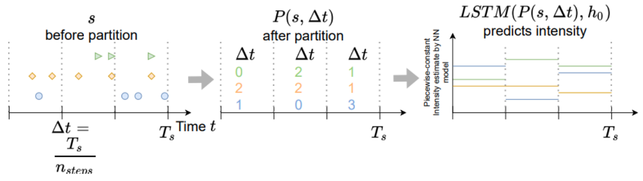


Figure 3. Image visualising the first 4 steps of Cohortney, [Zhu+21]

The most detailed description of the method is in the original paper, [Zhu+21].

2.4. Possible applications of the project

It is worth mentioning that our project as a tool may be used for testing and evaluation of different methods of event clustering. According to personal experience and advice in peer reviews, the methods themselves may be useful for but not limited to such tasks as:

- monitoring to preempt failure in business processes;
- bank transactions;
- websites use clustering;
- clustering visitors of different institutions;
- monitoring of appearances of medical manifestations of patients.

3. Experiments

3.1. Datasets

We use both synthetic and real-world datasets in the evaluation of our approach. We use pre-generated by original authors datasets. Every synthetic data sets with various clusters use sine-like impact functions and piecewise constant impact functions, respectively, and the other four with exponential kernel simulations. In each data set, we set the number of clusters between 2 to 5. Each cluster contains 400 event sequences, and each event sequence contains 5 event types. All datasets could be downloaded automatically using our framework. Some datasets based on the harmonical impact function, where piecewise constant impact function is a truncation corresponding sine-like impact function. Another part of random data have exponential impact function. All datasets precomputed by the original authors of the cohortney.

We also support real data: Age & LinkedIn datasets. According to the original description of Age dataset, it contains sequences of transaction records stemming from clients at financial institutions. For each client, we have a sequence of transactions. Each transaction is described with a discrete code, identifying the type of transaction, the Merchant Category Code, such as a bookstore, ATM, drugstore, and more. According to transactions' history, dataset is categorized clients into age categories by the authors. The data provides each client's age bin with four different bins in total.

The LinkedIn contains users' history of employers. There are 82 different companies, comprising different event types, and with 2439 users in total. Based on this data, we can categorize users into different professional areas. There is information about industry sectors for each user to compare those sectors with real ones. The event sequences in these three datasets have strong but structural triggering patterns, which we can model via different Hawkes processes.

3.2. Evaluation

We use evaluation protocol designed in [XZ17]. The clustering consistency is the minimum proportion of preserved pairs over all trials:

$$\text{Consistency} = \min_{j\in\{1,\dots,J\}} \sum_{j'\neq j} \sum_{(n,n')\in\mathcal{M}_j} \frac{1\left\{k_n^{j'} = k_{n'}^{j'}\right\}}{(J-1)\left|\mathcal{M}_j\right|},$$

where $\mathcal{M}_j = \left\{(n,n') \mid k_n^j = k_{n'}^j\right\}$ is the set of sequence pairs within same cluster in the j -th trial, and k_n^j is the index of cluster of the n -th sequence in the j -th trial.

For the synthetic data with clustering labels, we use clustering purity to evaluate various methods:

$$\text{Purity} = \frac{1}{N}\sum_{k=1}^{K} \max_{j\in\{1,\dots,K'\}} \left|\mathcal{W}_k \cap \mathcal{C}_j\right|$$

where $\mathcal{W}_k$ is the learned index set of sequences belonging to the k -th cluster, $\mathcal{C}_j$ is the real index set of sequence belonging to the j -th class, and N is the total number of sequences.

In all experiments, Purity, [MRS08] is the target metric of our framework.

4. Results

In this section, we report some of results of considered methods given datasets. Here, the columns correspond to the methods and the rows correspond to the datasets and on the intersection we report the value of Purity metric of given method on given dataset in the form $mean \pm std$.

Table 1. Quantitative results of considered methods

Dataset / Method	CAE	DeepCluster
K2_C5	0.83 ± 0.09	0.71 ± 0.06
K5_C5	0.37 ± 0.04	0.32 ± 0.11
LinkedIn	0.20 ± 0.01	0.19 ± 0.02
sin_K2_C5	0.95 ± 0.02	0.91 ± 0.06

These results are mainly illustrative, still they may be used to outline several features:

- the clustering with convolutional autoencoders over Cohortney on average outperforms DeepCluster over Cohortney both on synthetic and real-world data;
- it is simpler to cluster synthetic data (due to procedure of generation) than real-life data;
- increasing the number of ground truth cluster affects performance of the models and implies fine-tuning with different hyperparameters;
- overall runs of the methods are hyperparameters-dependent.

5. Conclusion

We proposed a framework for event sequences clustering methods which have a lot of industrial applications. The adding new methods to the framework is straightforward – the user has to specify the train module and add corresponding configuration module(s). All the modules in the project are independent and configurable. Also, methods considered in the work were evaluated both on synthetically generated dataset and real-world data and the results show applicability of them in industrial settings. Since the main challenge in such projects is the difference in APIs of methods considered, one of possible areas of improvement of the project is adding more methods related to the problem unifying their APIs.

References

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. “Maximum Likelihood from Incomplete Data via the EM Algorithm”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* 39.1 (1977), pp. 1–38. ISSN: 00359246. URL: http://www.jstor.org/stable/2984875.

[GSC99] F.A. Gers, J. Schmidhuber, and F. Cummins. “Learning to forget: continual prediction with LSTM”. In: *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*. Vol. 2. 1999, 850–855 vol.2. DOI: 10.1049/cp:19991218.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. “Probabilistic information retrieval”. In: *Introduction to Information Retrieval*. Cambridge University Press, 2008, pp. 201–217. DOI: 10.1017/CBO9780511809071.012.

[IS15] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[Guo+17] Xifeng Guo et al. “Deep clustering with convolutional autoencoders”. In: *International conference on neural information processing*. Springer. 2017, pp. 373–382.

[XZ17] Hongteng Xu and Hongyuan Zha. “A dirichlet mixture model of hawkes processes for event sequence clustering”. In: *Advances in Neural Information Processing Systems*. 2017, pp. 1354–1363.

[Zhu+21] Vladislav Zhuzhel et al. *COHORTNEY: Deep Clustering for Heterogeneous Event Sequences*. 2021. arXiv: 2104.01440 [cs.LG].

A. Distribution of roles of participants

Finally, we briefly outline the roles of participants:

- Taras: realisation of methods, configuration, documentation paperwork (reports, reviews, etc.);
- Misha: data processing, usage guide, debugging, experiments, paperwork (reports, reviews, etc.)