HyperStar2: Easy Distribution Fitting of Correlated Data

Zhihao Shang, Tianhui Meng and Katinka Wolter
Department of Mathematics and Computer Science
Freie Universität Berlin
Takustr.9, Berlin, Germany
{zhihao.shang, tianhui.meng and katinka.wolter}@fu-berlin.de

ABSTRACT

In this paper, we present HyperStar2, a tool for fitting Markov Arrival Processes (MAPs) to empirical data. HyperStar2 uses a two-step approach, where the first step is cluster-based fitting of phase-type distributions and the second step is the construction of a correlation matrix. In the first step, we use the cluster-based algorithm for Hyper-Erlang distribution fitting from HyperStar [1]. Based on the Hyper-Erlang fitting result and the clusters of samples, in the second step we construct the correlation matrix.

The tool targets engineers and scientists who need distribution fitting for non-standard distributions but have little interest in the underlying theory. Therefore the tool has a GUI that offers graphical presentation of the data, the fitted distribution and the empirical as well as theoretical autocorrelation. We discuss the use of HyperStar2 in common application scenarios and the fitting algorithms behind it.

We provide some numerical examples, which show the abilities and limits of the fitting tool. We find that HyperStar2 can fit distributions very well but for some autocorrelation structures ProFiDo provides better results for autocorrelation.

CCS Concepts

•Mathematics of computing → Markov processes;

Keywords

probability distribution fitting, phase-type fitting, MAP fitting, tool description

1. INTRODUCTION

Markovian Arrival Processes (MAPs) play an important role in the field of performance evaluation. MAPs have a long history in stochastic modeling [2] and are powerful modelling tools. MAPs, in theory, allow the representation of almost all relevant stochastic behaviors that are observed in practice. MAPs were proposed in [3] and are widely used for probabilistic analysis of communication network traffic and stochastic modeling [4]. To capture empirical behavior by MAPs, the parameters of a MAP have to be fitted according to some trace resulting from observations. The most general approach is to find a MAP that maximizes the likelihood according to the available traces. The EM algorithm [5] can be used for this purpose and many specific variants of the algorithm for MAP fitting are available [6]. There are also some MAP fitting methods based on the two-phase approach [7], which suggested splitting the task into two phases: fitting of the inter-arrival times in the first phase by a PH fitting method and fitting the correlation in the second phase. However, it is still an open question what are the statistics that capture the correlation structure of the trace the best.

In recent years, several tools have been proposed [8, 9] that provide good fitting results. But only users with deep understanding of the underlying concepts of the distributions and the applied algorithms can use them well. The motivation of our work is to develop a user-friendly tool which allows for intuitive user interaction and provides direct user feed-back. Using HyperStar2 requires no knowledge of the underlying mathematics or theoretical foundations. The tool is used by working directly with the empirical data.

In this paper, we discuss our cluster-based approach to MAP fitting, implemented as HyperStar2, a tool with a graphical user interface where the user can mark density peaks of the empirical data to support the clustering algorithm and improve fitting results. The rest of this paper is structured as follows: We first introduce some basic concepts of PH distributions and MAPs in section 2. We describe our cluster-based fitting approach in section 3. Section 4 gives an overview of our implementation of the HyperStar2 tool. We evaluate the tool in section 5 by numerical experiments. Section 6 concludes the paper with an outlook on future work.

2. MATHEMATICAL BACKGROUND

A Hyper-Erlang distribution is a mixture of Erlang distributions. We consider a Hyper-Erlang distribution that consists of M independent Erlang distributions weighted with initial probabilities $\alpha_1, \alpha_2, ..., \alpha_M$, where $0 < \alpha_m \leq 1$. The number of phases in the mth Erlang distribution is denoted with r_m and the rate parameter of the mth Erlang distribution is denoted λ_m. The Hyper-Erlang distribution is commonly represented by a vector-matrix tuple $(\alpha, \mathbf{Q})$, where α is the vector of initial probabilities and $\mathbf{Q}$ is the generator

ICPE'17, April 22 - 26, 2017, L'Aquila, Italy
© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4404-3/17/04…$15.00
DOI: http://dx.doi.org/10.1145/3030207.3030243

matrix. They are presented in (1) and (2).

$$\alpha = (\alpha_1, \alpha_2, ..., \alpha_M) \quad (1)$$

$$\mathbf{Q} = \begin{pmatrix} \mathbf{Q}_1 & 0 & & \\ 0 & \mathbf{Q}_2 & \ddots & \\ & \ddots & \ddots & 0 \\ & & 0 & \mathbf{Q}_m \end{pmatrix} \quad (2)$$

where $\mathbf{Q}_m$ is the generator matrix of the mth Erlang distribution.

An essential feature of Hyper-Erlang distribution is that an observation belongs to Erlang branch m always with probability α_m. This creates no correlation in the Hyper-Erlang distribution. To reflect correlation of samples, a MAP should be used instead of a Hyper-Erlang distribution.

A MAP of order n is usually defined by two $n \times n$ matrices $(\mathbf{D}_0, \mathbf{D}_1)$. Matrix $\mathbf{D}_0$ contains the rates of internal transition rates without an arrival and matrix $\mathbf{D}_1$ describes the transition rates with an arrival. And $\mathbf{D} = \mathbf{D}_0 + \mathbf{D}_1$ is an irreducible generator of the background n-state continuous-time Markov chain (CTMC). Let φ be the steady-state probability vector of the background CTMC, then φ is the solution of the linear system $\varphi\mathbf{D} = 0, \varphi\mathbf{1}^T = 1$, where $\mathbf{1}$ is a row vector of ones of the appropriate dimension. The pdf of a MAP is

$$f(t) = \pi e^{\mathbf{D}_0 t}(-\mathbf{D}_0\mathbf{1}) \quad (3)$$

and the lag-k correlation is computed as

$$\rho_k = \frac{\lambda^2\pi(-\mathbf{D}_0)^{-1}P^k(-\mathbf{D}_0)^{-1}\mathbf{1} - 1}{2\lambda^2\pi(-\mathbf{D}_0)^{-1}(-\mathbf{D}_0)^{-1}\mathbf{1} - 1} \quad (4)$$

3. THE FITTING ALGORITHM

In this section we discuss the theoretical aspects of the fitting algorithm. The main idea of the algorithm is to construct the $\mathbf{D}_0$ and $\mathbf{D}_1$ matrices separately. So there are two steps in our MAP fitting algorithm. The first step is using a cluster-based algorithm to fit the samples with a Hyper-Erlang distribution. We use the generator matrix of the Hyper-Erlang distribution as $\mathbf{D}_0$ matrix. After performing the cluster-based fitting, we have the Hyper-Erlang distribution and M clusters. The matrix $\mathbf{D}_1$ is constructed by analysing the sample transitions between clusters and assigning the relative switching frequency between two clusters to the respective transition probability.

3.1 Constructing the $\mathbf{D}_0$ matrix

The cluster-based fitting algorithm consists of clustering and refinement. In the clustering step, the samples are clustered using the k-means alorithm and each cluster is fitted with an Erlang distribution. We use a probabilistic re-assignment strategy for the cluster refinement. The refinement is repeated iteratively until either the parameters in each Erlang distribution do no longer change or a maximal number of iterations has been exceeded. More details can be found in our earlier work [1].

3.2 Constructing the $\mathbf{D}_1$ matrix

For a Hyper-Erlang distribution, the phase distribution after an arrival is always α. For a MAP, the phase distribution after an arrival depends on the phase immediately before that arrival and $\mathbf{D}_1$ describes the transitions rate between phases, which are states of a CTMC.

Our approach is to analyse the relative frequency of transitions between clusters, and constructing the $\mathbf{D}_1$ based on the transition frequencies. When an arrival occurs, let $\eta_{(i,j)}$ be the probability that the next phase is j given the current phase is i. We can obtain the element of $\mathbf{D}_1$ at row i column j by scaling $\eta_{(i,j)}$ with a proper factor because $\mathbf{D}_1$ and $\eta_{(i,j)}$ describe the same thing using different notation. In our MAP fitting algorithm, we construct $\mathbf{D}_1$ based on the relative frequency of transitions between clusters.

After Hyper-Erlang fitting, a generator matrix $\mathbf{Q}$ for the Hyper-Erlang distribution is obtained. If there are k_i phases in the Erlang distribution of cluster i it follows that $\mathbf{D}_0$ is a $K \times K$ matrix, where $K = \sum_{i=1}^{M} k_i$. Because the matrix $\mathbf{D}_0$ is associated with transitions without arrivals and we use a generator matrix of a Hyper-Erlang distribution in our MAP fitting algorithm, transitions with arrivals only happen between Erlang branches. Every Erlang branch is associated to a sample cluster and we analyze transitions between clusters to obtain the transitions between Erlang branches that are associated with arrivals. Let $\mathbf{CS}$ be a $K \times K$ zero matrix and $\mathbf{CS}_{(r,c)}$ be the element at row r column c. We use matrix $\mathbf{CS}$ to count the transitions between clusters.

Let B_i be the first phase of the ith Erlang distribution and E_i the last phase of the ith Erlang distribution. We can get

$$B_i = \sum_{a=1}^{i-1} k_a + 1 \qquad E_i = \sum_{a=1}^{i} k_a. \quad (5)$$

If a sample belongs to the ith cluster and the next sample belongs to the jth cluster, where i may be equal to j, we update the counter as below:

$$\mathbf{C}S_{(E_i,B_j)} = \mathbf{CS}_{(E_i,B_j)} + 1. \quad (6)$$

We can set $\mathbf{D}_1^{'}$ to the probability matrix for sample transitions.

$$\mathbf{D}_1^{'} = \frac{1}{n-1}\mathbf{CS}. \quad (7)$$

Let $\mathbf{D}_{1(r,c)}^{'}$ be the element of $\mathbf{D}_1^{'}$ at position (r, c). Hence $\mathbf{D}_{1(r,c)}^{'}$ is the probability that the sample generating process transits from state r to state c. We assume $\mathbf{D}_{1(r,c)}^{'}$ equals the probability that transitions occur between states in the MAP. To get $\mathbf{D}_1$, we must convert the probability in $\mathbf{D}_1^{'}$ to state transition rates by scaling every element in $\mathbf{D}_1^{'}$ with a suitable factor. Let $\mathbf{F}$ be the matrix of factors and $\mathbf{D}_1$ is the Hadamard product of $\mathbf{D}_1^{'}$ and $\mathbf{F}$.

$$\mathbf{D}_1 = \mathbf{D}_1^{'} \circ \mathbf{F}. \quad (8)$$

The matrix $\mathbf{D}_0 + \mathbf{D}_1$ is an irreducible generator of a CTMC, so

$$(\mathbf{D}_0 + \mathbf{D}_1)\mathbf{1} = \mathbf{0}, \quad (9)$$

where $\mathbf{0}$ is a row vector of zeros of the appropriate dimension.

Let $\mathbf{D}_{1(r,c)}$ be the element of $\mathbf{D}_1$ at row r column c. Combining (8) and (9), then we can get

$$\mathbf{D}_{1(r,c)} = \begin{cases} -\frac{\mathbf{D}_{0(r)}}{\mathbf{D}_{1(r)}^{'}}\mathbf{D}_{1(r,c)}^{'} & \text{if } \mathbf{D}_{1(r)}^{'} \neq 0 \\ 0 & \text{if } \mathbf{D}_{1(r)}^{'} = 0 \end{cases} \quad (10)$$

where

$$\mathbf{D}_{0(r)} = \sum_{c=1}^{K} \mathbf{D}_{0(r,c)} \qquad \mathbf{D}'_{1(r)} = \sum_{c=1}^{K} \mathbf{D}'_{1(r,c)} \tag{11}$$

4. IMPLEMENTATION

We implemented the fitting method in the tool HyperStar2. In earlier work [1] we found that the human user can very often detect clusters much better than a fully automatic algorithm by marking the peaks in the distribution as relevant values in the observations.

For most cases, users do not have to set the parameters and HyperStar2 can provide good fitting results with the default values after simply clicking the *Fit* button. If the user is not satisfied with the fitting result, the peaks on the histogram can be marked in the GUI. The density peaks are used as the initial centers of the K-means algorithm. There are also fitting parameters that can be set to improve the fitting result. Table 1 lists some important fitting parameters to control the behaviour of the fitting algorithm. The tool can export the result after completion of the fitting procedure. It can only export plain text for now. Among our the future work is the export of scripts that can generate samples from the fitted MAP in some simulators.

Table 1: Fitting parameters

Parameter	Description(default value)
fitter	distribution to be fitted (MAP)
branch	number of branch to be fitted (6)
reassignment	maximum number of iterations (20)
shuffles	number of reassignments in a iteration (2)

5. EVALUATION

In this section we demonstrate the fitting properties of HyperStar2 by analysing some examples. We fit distributions and evaluate the goodness of fit by comparing the empirical and the theoretical probability density function as well as the lag-k correlation. We also compare our result with the results from ProFiDo [8], which we found to be a very versatile fitting tool. Computation of error measures is also among our future work.

We show two examples that demonstrate the advantages and disadvantages of our tool. We first generated samples from a given MAP and then study how well HyperStar2 is able to approximate this distribution. We use samples that contain obvious peaks in the first example. There are not many samples in the overlap area between clusters. Therefore, it is clear which cluster a sample falls into and the estimate of the relative frequency of transitions between clusters in the sample sequence is highly accurate. In this case the algorithm provides a good correlation fit. In the second example, the samples do not contain as obvious peaks. Hence, there are many samples in the overlap area between clusters. These samples may be assigned to different clusters. The assignment may not impact the density fitting very much, but we suspect that it impairs the autocorrelation fitting as the estimate of relative frequencies of changes between clusters in the sample sequence has much more uncertainty.

The *branch* parameter indicates into how many clusters the data is splitted. This is a very import parameter in our

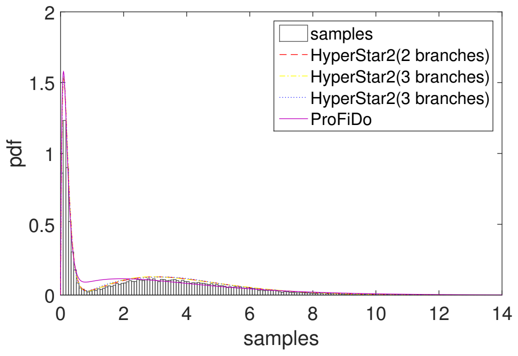

Figure 1: pdf of the first example

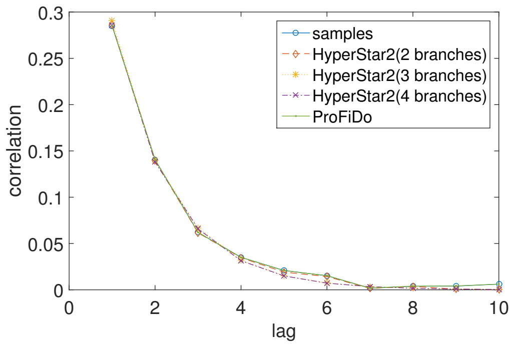

Figure 2: correlation of the first example

algorithm as it impacts the fitting result severely. So we show fitting results with different numbers of *branches* in the two examples.

The first example uses a 6-state MAP with the following matrices. Using this generator there are two states that contribute strongly to the dynamics of the model, while the remaining four states are less influential. This can also be seen in Figure 1, where the empirical density has two peaks.

$$\mathbf{D}_0 = \begin{pmatrix} -10 & 10 & 0 & 0 & 0 & 0 \\ 0 & -10 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix}$$

$$\mathbf{D}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 0 & 3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.3 & 0 & 0 & 0.7 & 0 & 0 \end{pmatrix}$$

The first data set consists of 25000 samples generated from this MAP. We fit the samples using HyperStar2 and ProFiDo. Setting the *branch* parameter to 2, which means the algorithm splits the data into 2 clusters and there are 2 Erlang branches in the Hyper-Erlang distribution. Fig. 1 shows the histogram of samples and the densities of the fitted distributions. The solid line represents the pdf of the result from ProFiDo. The other three lines represent the pdf of results from HyperStar2 using different parameter settings. The red line shows the 2-branch result from HyperStar2, the yellow and blue lines show the results for 3 branches and 4 branches respectively. From Fig. 1 we observe that HyperStar2 fits this data set very well with 2, 3 or 4 branches. ProFiDo provides a good fit of the density but is not able to capture the second peak and the gap between the peaks as precisely.

Fig. 2 shows the autocorrelation function of the resulting MAP. As expected, the autocorrelation function is fitted well by both tools. The figure indicates that the result with 2 Erlang branches is the best of the results we have computed using HyperStar2. It is as good as the result from ProFiDo.

But for different numbers of Erlang branches the results are not as good. The 3-branch and 4-branch results for the density are not as good as 2-branch result, but they are also very close to the samples' autocorrelation.

In the second example, we approximate the following MAP:

$$\mathbf{D}_0 = \begin{pmatrix} -0.2 & 0.2 & 0 & 0 & 0 & 0 \\ 0 & -0.2 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & -0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -0.5 & 0.5 & 0 \\ 0 & 0 & 0 & 0 & -0.5 & 0.5 \\ 0 & 0 & 0 & 0 & 0 & -0.5 \end{pmatrix}$$

$$\mathbf{D}_1 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.16 & 0 & 0 & 0.04 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0.02 & 0 & 0 & 0.03 & 0 & 0 \end{pmatrix}$$

We again generated 25000 samples from this MAP and fitted them with HyperStar2 and ProFiDo.

We used 2, 4 and 6 *branches*. It is not easy to see the peaks in this, so we did not mark any peak on the histogram. Fig. 3 and Fig. 4 show the fitting results.

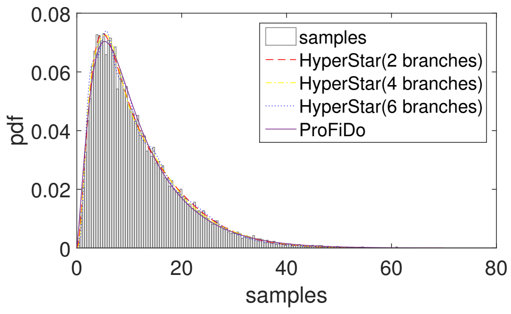

Figure 3: Probability density of the second example

Fig. 3 shows that both ProFiDo and HyperStar2 can fit the distribution very well. Although there are 2 Hyper-Erlang branches in the original MAP, HyperStar2 can still provide a good fit with 4, or 6 branches.

From Fig. 4, we can see that ProFiDo fits the correlation better than HyperStar2. But the distribution fit of HyperStar2 is still better than that of ProFiDo. We assume that HyperStar2 is less suited to capture relatively small autocorrelation, which means that clusters have a strong overlap. High autocorrelation typically shows as strongly distinct clusters and those are captured much better by HyperStar2 than less distinct clusters.

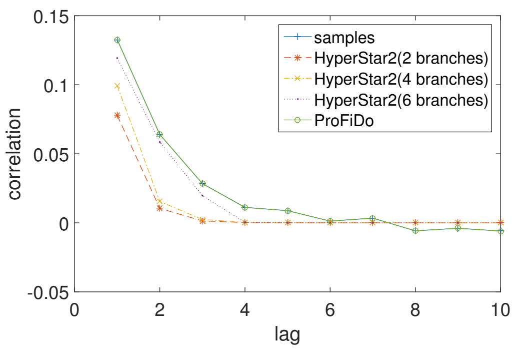

Figure 4: Correlation of the second example

6. CONCLUSION AND FUTURE WORK

In this paper we have presented HyperStar2, a fitting tool for correlated data and the algorithms behind it. We have shown that HyperStar2 can capture the shape of a distribution very well, but autocorrelation is not in all cases represented as well as done by ProFiDo. Numerical examples show that the tool can fit empirical data with MAPs very well. In the future, we will improve the accuracy of autocorrelation fitting and provide the user with metrics for the goodness-of-fit.

7. REFERENCES

[1] Philipp Reinecke, Tilman Krauß, and Katinka Wolter. Cluster-based fitting of phase-type distributions to empirical data. *Computers & Mathematics with Applications*, 64(12):3840–3851, 2012.

[2] Marcel F Neuts. Matrix-geometric solutions in stochastic models, volume 2 of johns hopkins series in the mathematical sciences, 1981.

[3] David M Lucantoni, Kathleen S Meier-Hellstern, and Marcel F Neuts. A single-server queue with server vacations and a class of non-renewal arrival processes. *Advances in Applied Probability*, pages 676–705, 1990.

[4] Alma Riska, Mark Squillante, Shun-Zheng Yu, Zhen Liu, and L Zhang. Matrix-analytic analysis of a map/ph/1 queue fitted to web server data. *Matrix-Analytic Methods; Theory and Applications*, pages 333–356, 2002.

[5] Peter Buchholz. An em-algorithm for map fitting from real traffic data. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 218–236. Springer, 2003.

[6] Lothar Breuer. An em algorithm for batch markovian arrival processes and its comparison to a simpler estimation procedure. *Annals of Operations Research*, 112(1-4):123–138, 2002.

[7] Peter Buchholz and Andriy Panchenko. A two-step em algorithm for map fitting. In *International Symposium on Computer and Information Sciences*, pages 217–227. Springer, 2004.

[8] Falko Bause, Peter Buchholz, and Jan Kriege. Profido-the processes fitting toolkit dortmund. In *Quantitative Evaluation of Systems (QEST), 2010 Seventh International Conference on the*, pages 87–96. IEEE, 2010.

[9] Giuliano Casale, Eddy Z Zhang, and Evgenia Smirni. Kpc-toolbox: Simple yet effective trace fitting using markovian arrival processes. In *Quantitative Evaluation of Systems, 2008. QEST'08. Fifth International Conference on*, pages 83–92. IEEE, 2008.