HAL
archives-ouvertes.fr

Day-ahead time series forecasting: application to capacity planning

Colin Leverger, Vincent Lemaire, Simon Malinowski, Thomas Guyet, Laurence Rozé

To cite this version:

Colin Leverger, Vincent Lemaire, Simon Malinowski, Thomas Guyet, Laurence Rozé. Day-ahead time series forecasting: application to capacity planning. AALTD'18 at ECML 2018, Sep 2018, Dublin, Ireland. <hal-01912002>

HAL Id: hal-01912002

https://hal.inria.fr/hal-01912002

Submitted on 5 Nov 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Day-ahead time series forecasting: application to capacity planning

Colin Leverger[1,2], Vincent Lemaire[1]
Simon Malinowski[2], Thomas Guyet[3], Laurence Rozé[4]

[1] Orange Labs, 35000 Rennes – France
[2] Univ. Rennes INRIA/CNRS IRISA, 35000 Rennes – France
[3] AGROCAMPUS-OUEST/IRISA - UMR 6074, 35000 Rennes – France,
[4] INSA Rennes, IRISA, 35000 Rennes – France
{colin.leverger}@orange.com

Abstract. In the context of capacity planning, forecasting the evolution of informatics servers usage enables companies to better manage their computational resources. We address this problem by collecting key indicator time series and propose to forecast their evolution a day-ahead. Our method assumes that data is structured by a daily seasonality, but also that there is typical evolution of indicators within a day. Then, it uses the combination of a clustering algorithm and Markov Models to produce day-ahead forecasts. Our experiments on real datasets show that the data satisfies our assumption and that, in the case study, our method outperforms classical approaches (AR, Holt-Winters).

Keywords: Time series, Capacity Planning, Clustering, Markov Models.

1 Introduction

Capacity planning (CP) is one major preoccupation for todays companies. This ensemble of practices characterises the way an enterprise manages, updates, adds or removes physical/virtualised servers from its infrastructure. A well-made CP helps to reduce operational costs, and improves the quality of the provided services: indeed, one of the major goals is to maintain the best quality of services for the end users. Most of the time, project owners, managers and experts apply CP guidelines to manage their infrastructure manually. This approach is cumbersome and overestimate needs to prevent from any business interruptions. As the manager will have to daily take decisions about its infrastructure, we are interested in forecasting the full time series of the day ahead. Contrary to a lot of time series forecasting techniques, the forecasting horizon is of several steps in this study. We could then rely on the circadian rhythm of the data to improve forecasts.

Bodik et *al.* [2] show that data analytics on datacenters key performance indicators (KPI) may be used to do CP (*e.g.*, CPU, number of users, RAM). Indeed, the data generated by servers or user activity could be valuable sources of

information. One particular data analytic task that may help resource managers in the daily activity is to forecast the evolution of the KPI in time. That is basically a time series forecasting task. The more accurate are forecasted the KPI, the more informed will be the management decisions regarding CP. In particular, we make the assumption, denoted ($\mathcal{A}$), that *KPI time series are driven by a two-order temporal process*. The circadian scale drives the daily evolution of the KPI but, at the second order, this daily behaviour is itself driven by some hidden rules. The evolution within a week could be considered. During the weekdays (from Monday to Friday) KPIs have a daily behaviour which is different from the weekend daily behaviour.

The method we propose is to capture the daily behaviour of the KPI by defining some "types of days" and thus to detect second-order behaviours by analysing sequences of typical days. For now and as a primary study, we did not take into account the position of a day in a week (*e.g.*, Mondays, Tuesdays, ...) and only consider types of days regarding shapes of the time series data only. The experiments compare the forecasting performance of our method with different baseline in order to validate our hypothesis $\mathcal{A}$ on a large collection of real data coming from a very large infrastructure (large time depth and great variability of KPI).

2 Day-ahead time series forecasting

In this section, we present our approach to produce one-day-ahead time series forecasting. This approach is composed of three learning steps (i) data normalisation and split, (ii) clustering, (iii) next-day cluster estimation, and a forecasting step (iv) next day forecasting. The learning steps take as input a multivariate time series and build a forecasting model. In the forecasting step, the model is applied to the time series current day in order to forecast the next day. Fig. 1 depicts the learning steps of our approach, *i.e.* from (i) to (iii), together with the forecasting step (iv). The four steps are presented below.

Let $\boldsymbol{X} = \langle \boldsymbol{X}_1, \ldots, \boldsymbol{X}_n \rangle$ be a multivariate time series of length n. For all $i \in [1, n]$, $\boldsymbol{X}_i \in \mathbb{R}^p$, where p denotes the dimensionality of $\boldsymbol{X}$ (the number of monitored KPI in our application case). Let h be the forecasting horizon that corresponds to one day.

(i) Data normalisation and split: First, we normalise each dimension of $\boldsymbol{X}$ using a $\mathcal{N}(0, 1)$ normalization. Then, we construct the set $\mathcal{D} = \{\boldsymbol{D}_1, \ldots, \boldsymbol{D}_d\}$, where $\boldsymbol{D}_i$ represents the subseries of $\boldsymbol{X}$ corresponding to the ith day of measurements, and $d = \frac{n}{h}$. For sake of simplicity, we assume that n is a multiple of h.

(ii) Clustering: The elements of $\mathcal{D}$ are then given to a clustering algorithm. In our case, we use the multidimensional k-means based on the Euclidean distance. The centroids of the created clusters are denoted $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_k$. They correspond to typical day evolution. The choice of the number k of clusters will be discussed in Section 3.

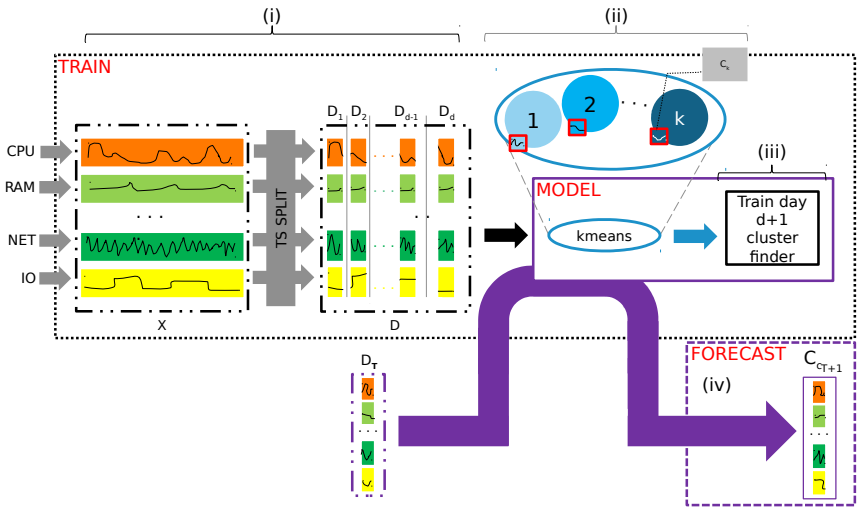

Fig. 1. Illustration of the main steps of the day-ahead time series forecasting: (i) multivariate time-series splitting in daily time series, (ii) time series clustering, (iii) day sequentiality modelling, (iv) forecast on a new daily time series (see details in the text).

(iii) Next-day cluster estimation: The set $\mathcal{D}$ is encoded into a sequence $\langle c_1, \ldots, c_d \rangle$, where $c_i \in [1, k]$ corresponds to the index of the cluster that contains the day $\boldsymbol{D}_i$. We model this sequence by a first-order Markov Model of transition matrix $\Pi = (\pi_{i,j})$, where $\pi_{i,j}$ is the probability that the next day belongs to cluster j given that the current day belongs to cluster i. These probabilities are estimated using the sequence $\langle c_1, \ldots, c_d \rangle$. In other words, this Markov Model enables us to estimate the most probable cluster to which will belong the next day of measurements.

(iv) Next-day forecasting: This forecasting step makes use of the model learned above to predict the next day measurements ($\boldsymbol{D}_{T+1}$) given the current day measurements ($\boldsymbol{D}_T$). First, the index of the closest cluster to $\boldsymbol{D}_T$ is computed, and denoted c_T. The most probable cluster for the next day is estimated using the transition matrix Π : $c_{T+1} = \arg\max_{j \in 1,\ldots,k} \pi_{c_T,j}$. Finally, the forecasting of the next day is given by the centroid $\boldsymbol{C}_{c_{T+1}}$.

3 Experiments and Results

In this section, we briefly introduce our real capacity planning dataset. Before the results, we present the evaluation protocol of our approach and its underlying assumption stating time series with two-orders temporal scales.

3.1 Data

This work has been endorsed by a growing project at Orange: Orange Money (OM)[5]. This latest was established in late 2008. It aims in providing an easy access to bank transfers for African customers. The bank exchange system is hosts by a large Orange infrastructure (162 servers) and is used by more than 31 million customers across all of Africa and Europe.

The datasets are collected from this infrastructure using Nagios supervisor. It collects technical and operational metrics every five minutes:

- *Technical data*: data concerning the servers performance such as percentages of CPU/memory use across all servers,
- *Operational data*: data concerning users browsing the service; number of financial transactions every minute, number of people on a website, etc.

3.2 Protocol

This study has been conducted in two major steps:

(i) We have first experimented using univariate time series, where the day $T+1$ is forecasted using only past data from current time series. Our multivariate time series have therefore first being exploited as univariate ones (we took each features independently).

(ii) We have also experimented using multivariate time series, where the past of several series is used to forecast future of one KPI. We were curious about the possible improvement of the accuracy using multivariate time series. Indeed, we can assume that having more information describing the datacenter behaviour may help improve forecasts.

The performance of our method is compared to four alternative approaches. Two baseline methods are used to evaluate our assumption $\mathcal{A}$ on the time series structure in the dataset:

- *Mean day* is a simple mean day calculation: all days present in learning ensemble $\mathcal{D}$ are used to compute the average day. This latest is then given as a forecast result for day $T+1$. Intuitively, it should give the worst forecasts if there are actually very different types of days.
- *Omniscient algorithm* is an adaptation of the day-ahead forecasting method (with same clusters) with an omniscient prediction of c_{T+1} instead of the Markov model. This baseline is used to evaluate the Markov Model independently of the clustering step. It assumes that we know to which cluster belong day $T+1$. The forecasts given by this method cannot be worse (in terms of prediction error) than the ones given by our approach.

The relative performance of our approach with regard to these two baselines gives indication about the compliance of data with our assumption $\mathcal{A}$. Moreover, two classical forecasting techniques are used to compare the quality of the forecasts given by our approach:

[5] `https://orangemoney.orange.fr/`

Table 1. Errors and ranks for univariate forecasting results.

Algorithm	Mean error ± std	Mean rank ± std
Omniscient algorithm	$\mathbf{0.45 \pm 0.83}$	$\mathbf{1.51 \pm 0.79}$
Day-ahead forecasting	0.52 ± 0.87	2.62 ± 0.91
AR model	0.73 ± 0.92	3.30 ± 1.38
HW model	*314 ± 1775*	*3.91 ± 1.37*
Mean day	0.71 ± 1.03	3.66 ± 0.95

1. *Auto-Regressive (AR)* model [1], which is a representation of a random process that can be used to describe some time series,
2. *Holt Winters (HW)* Triple Exponential Smoothing [10], which is a rule of thumb technique for smoothing time series data. It extends the Holts method to capture seasonality.

These two methods have been preferred to more complex models such as ARIMA models [6] or LSTM [7] whose parameters are uneasy to configure.

We quantify the prediction error by computing the Mean Square Error (MSE) between the forecasted values and the real values. The dataset is split chronologically into three parts: 70% of the data as training data, 15% as validation data and 15% as test data. The validation dataset is used to select the best number of clusters k (ranging from 2 to 200) based on the MSE.

3.3 Results

Table 1 presents the results obtained for 458 univariate time series (both operational and technical coming from 162 servers) of the OM projects. To obtain more significant results, we use all day present on the test ensemble to produce forecasts (*i.e.* if the test ensemble is composed of 6 days, we produce 5 forecasts (excluding the first day). The mean errors are computed as the average MSE per univariate time series. For each forecast, the methods are ranked by decreasing error, and we compute the mean rank.

Let us first analyse the results to assess our assumption $\mathcal{A}$ about the underlying data structure. We observe that forecasting the centroid of the known cluster for day $T + 1$ (omniscient method) outperforms significantly the daily mean of the whole time series. This means that our assumption about the two temporal scales is satisfied by our dataset. The average number of clusters selected on the validation sets is 16 for 365 days of data. This number is sufficiently low to conclude that there are actually clusters of typical days.

Unsurprisingly, the omniscient method outperforms our method as the former always make the best choice. It also shows that the crucial step of forecasting the type of the day ahead could be improved. In fact, Markov models are very simple models and more advanced approaches could probably help improve our overall approach. But we mainly notice that our algorithm has better performance than AR and HW that is often chosen for seasonal time series forecasting. It can be explained by a weak adequacy of these methods to our task. AR and HW are

Table 2. Errors and ranks for multivariate forecasting results.never had the lower MSE not presented.

Algorithm	Mean error ± std	Mean rank ± std
Multivariate mean	0.1 ± 0.14	$\mathbf{2.07 \pm 1.48}$
Multivariate day-ahead forecasting	0.044 ± 0.1	2.24 ± 0.86
Omniscient multivariate algorithm	$\mathbf{0.041 \pm 0.09}$	2.44 ± 0.70
Omniscient univariate algorithm	0.5 ± 0.6	4.03 ± 0.85
Univariate day-ahead forecasting	0.57 ± 0.63	4.84 ± 0.87
AR model	0.76 ± 0.7	5.38 ± 1.38

more used to forecast next few points of the time series while our task requires to forecast the time series for the entire day (96 points). The mean error is higher for AR than the mean-day forecast, but its ranking is better, meaning that AR is better than a mean day on most of the time series, but when it fails, it fails with higher errors.

One noticeable thing is the pretty bad *performances of the HW model*. This could be explained because HW is less efficient in forecasting several points in the future, and often base its forecasts following the last tendency observed.

The previous experiments evaluate our algorithm on univariate time series. Table 2 presents results on multivariate time series with two CPUs from two different servers.

The multivariate mean algorithm is slightly better than our algorithm in terms of rank but not in terms of MSE. It indicates that having more features involved in the process increases the chance that the mean day is more representative for the forecast, and thus better.

One possible explanation is that the number of clusters is too small to extract meaningful groups of days. With multivariate time series, the number of types of day increases but we did not increase the maximum number of cluster (tested during on the validation set). Finally, one noticeable performance of our multivariate algorithms is its low mean error. Nonetheless, experiments on more time series are required to conclude, and but it shows that the method is suitable regarding technical CPU time series.

3.4 Discussion

Experimental results show that our approach performs well on our real dataset, but we also know that it is an early proposal that has some limitations and weaknesses.

In the first stage of our model, daily time series are clustered. At the time, we only experimented the k-means algorithm with a Euclidean distance. In practice, if detecting data peaks between 2 PM and 4 PM is something of interest, this distance is to be privileged. But if the goal is to detect peaks in a day without precise information about their timings, DTW [8] is a better candidate. In our mind, there is not a unique good choice but better some choices that fit the data

characteristics like Douzal-Chouakria et Amblard [4] suggest for a classification task. In addition, the clustering strategy could also be evaluated. One of the weaknesses of the current approach using a k-means algorithm is the critical choice of k. In this study, we find the optimal number of clusters using the validation set. We test various k-means sizes on the training dataset, and select the one which helps our algorithm in having a lower MSE. Less empirical techniques such as David et Bouldin criteria [5] or even Silhouette [9] could help to select a priori the best number of clusters with a lower computational cost.

An interesting thing would be to know which of the steps described on the Figure 1 is lowering forecast performances. This could help us in improving the chain by tuning very particular and identified parameters. The omniscient algorithm that knows for sure the day $T+1$ is useful as it reveals that having an almost perfect clustering algorithm could greatly enhance precision of forecasts (see Tables 1 and 2 which show that omniscient technique often outputs forecasts with lower mean error). Enhancing the quality of clustering could then be a key to better results.

In the second stage of our model, the experiment shows that there is possible improvements of the prediction of the type of the next day. A more accurate solution than Markov models can bridge the gap with the omniscient approach. At the time, a Markov model takes only into account the day before the one to predict. This simple model has been preferred to higher order Markov models because of the required quantity of training data. With 7 different days, transition matrix of size 49 are to learn in our case, but of size 343 for a 2-order Markov model. This requires long sequences of days to accurately estimate them. We currently collected 365 days of data history and thus prefer to focus on a simple but sound approach. A study of state-of-the-art approaches of sequence prediction with sparse data will help us to identify good candidates to replace the Markov model.

Finally, as mentioned in the introduction, our model does not take into account the position of the weekday to make forecasts. However, some observations let us believe that it could improve the prediction accuracy. In fact, we noticed that the clustering of daily time series set up with 2 clusters will extract two types of behaviours: A high activity profile (HA) from Mondays to Fridays and a low activity profile (LA) on Saturdays and Sundays. As a consequence, the trained Markov Model will more likely predict a HA day after another HA day (with probability 0.8). This means that every Friday, it will wrongly predict the day with a HA profile. A model with the weekday information will split the next day prediction rule in two different rules: High activity from Monday to Thursday leads to a high activity the next day while high activity on Friday leads to a low activity. It remains interesting to have information about the types of the days as, in real data, there are several different of profiles of days that are not necessarily correlated to the weekday information.

4 Conclusion

In this work, we presented a general method to address a specific problem of capacity planning, *i.e.* the forecast of the evolution of KPI indicators a day ahead. The method we propose is a time series forecasting method that is founded on the assumption that the time series are implicitly structured as a sequence of typical days. Our experiments comparing baseline approaches and classical time series forecasting methods to our method show that this assumption is fitted by most of the time series of our dataset. We now have to explore more deeply the characteristics of the data that does not fit it. to improve the overall approach. We observe an interestingly low mean square error for our algorithm on multivariate CPU and find these preliminary results promising. But the proposed method may also benefit from improvements in the two main stages: the clustering of time series and the sequential prediction. We strongly believe that Markov model may be improved by some more recent works on sequence prediction [3].

References

1. Akaike, H.: Fitting autoregressive models for prediction. Annals of the institute of Statistical Mathematics **21**(1), 243–247 (1969)
2. Bodik, P., Griffith, R., Sutton, C., Fox, A., Jordan, M.I., Patterson, D.A.: Automatic exploration of datacenter performance regimes. In: Proceedings of the 1st workshop on Automated control for datacenters and clouds. pp. 1–6. ACM (2009)
3. Budhathoki, K., Vreeken, J.: Causal inference on event sequences. In: Proceedings of the 2018 SIAM International Conference on Data Mining. pp. 55–63. SIAM (2018)
4. Chouakria, A.D., Amblard, C.: Classification trees for time series. Pattern Recognition **45**(3), 1076–1091 (2012)
5. Davies, D.L., Bouldin, D.W.: A cluster separation measure. IEEE transactions on pattern analysis and machine intelligence (2), 224–227 (1979)
6. Hillmer, S.C., Tiao, G.C.: An ARIMA-model-based approach to seasonal adjustment. Journal of the American Statistical Association **77**(377), 63–70 (1982)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
8. Niels, R.: Dynamic time warping. Artificial Intelligence (2004)
9. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics **20**, 53–65 (1987)
10. Winters, P.R.: Forecasting sales by exponentially weighted moving averages. Management science **6**(3), 324–342 (1960)