Feature-aware Forecasting of Large-scale Time Series Data Sets

Claudio Hartmann, Lars Kegel, Wolfgang Lehner

Technische Universität Dresden
Database Systems Group
01062 Dresden, Germany
firstname.lastname@tu-dresden.de

The Internet of Things (IoT) sparks a revolution in time series forecasting. Traditional techniques forecast time series individually, which becomes unfeasible when the focus changes to thousands of time series exhibiting anomalies like noise and missing values. This work presents CSAR, a technique forecasting a set of time series with only one model, and a feature-aware partitioning applying CSAR on subsets of similar time series. These techniques provide accurate forecasts a hundred times faster than traditional techniques, preparing forecasting for the arising challenges of the IoT era.

1 Introduction

Time series forecasting is an essential part of today's business landscape and has been a well-established research subject for many decades (*1, 2*). The automation of factory processes led by the Internet of Things (IoT), decision-making and investment planning in large companies, and transmission grid balancing in the energy sector are just a few examples. Especially IoT is pushing the Big Data trend which nowadays is virtually present everywhere. Thousands of sensors,

customers, and energy sources are monitored and analyzed at fine temporal and structural granularity to improve analytic results and gain a better understanding of the respective domains. As a result, the Big Data trend also affects the process of time series forecasting (*3–5*). While accurate forecasts are still most important, the computation has to keep up with the increasingly fine temporal granularity.

Typically, time series are forecast one by one, i.e., an individual model for a single time series. This induces the creation of thousands of forecast models. Together with the fine temporal granularity, which produces lots and lots of training data, the model creation becomes too time-consuming to be of actual practical use. Furthermore, time series at fine granularity tend to be noisier and have a higher portion of missing values resulting in novel challenges rendering approaches from the related literature unable (*3, 5*).

The **C**ross-**S**ectional **A**uto**R**egression model (CSAR) is a novel forecast technique designed for large-scale time series data sets (*3*). So far, the model is applied on a data set level where it forecasts all time series from a data set together. In order to further improve the forecast accuracy, we apply feature-aware partitioning techniques to divide the data set into partitions of similar time series and forecast each partition with an individual CSAR model. The evaluation of this approach shows encouraging results as it significantly improves the forecast accuracy while maintaining shorter execution time than traditional forecast techniques. In detail, the contributions of this work are the following:

- We give an overview of the specifics of forecasting large-scale time series data sets and introduce the main goals of this work in Section 2.

- In Section 3, we revisit the CSAR model and highlight its ability to address the arising challenges on forecasting large-scale time series data sets.

- Subsequently, in Section 4, we present a feature-aware partitioning technique that we use

to divide a data set into groups of similar time series that are forecast by individual CSAR models.

- We conduct an experimental study to show that partitioning a data set significantly improves the forecast accuracy compared to a monolithic CSAR model and cross-sectional forecasting is faster and more accurate than traditional forecast techniques in Section 5.

We close this paper with a short summary and an outlook to possible future research directions in Section 6.

2 Preliminaries

In this section, we first formulate the actual problem of time series forecasting. Subsequently, we highlight the new challenges on forecasting caused by the main drivers of the Big Data trend like IoT, i.e., the collection of large amounts of data at a fine granularity. Finally, we outline the solutions we are going to present in this paper.

2.1 Time Series Forecasting

In many application domains, data is collected from multiple sources, resulting in a large number of time series that have to be analyzed, modeled, and forecast. For example, the energy consumption of each individual customer of an energy provider produces its own stream of information. The targets of the analysis range from individual time series, e.g., the energy consumption of specific consumers, to different aggregates, e.g., the energy consumption of a district or a whole city. We refer to such a collection of time series as a data set Y where all time series y^n – n is a unique time series identifier – are recorded at the same points in time 1, ..., t. Figure 1 shows an example time series with a corresponding forecast. The time series is marked with connected black crosses ($\times$), each representing a measure value and the point

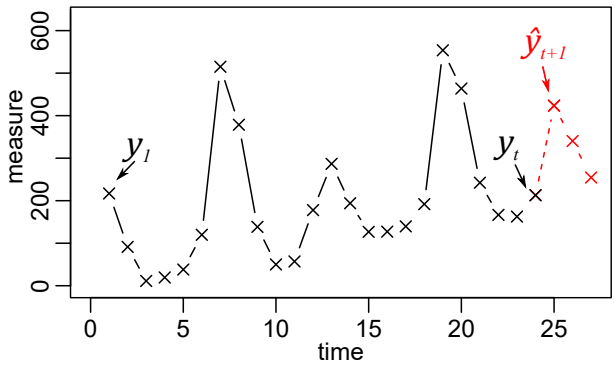


Figure 1: Example time series and forecast.

in time it was recorded. The forecast values y_{t+1}; ...; y_{t+h} are marked by red crosses ($\times$). The number of forecast values is called the forecast horizon and denoted by h, in this example $h = 3$. Traditional forecast techniques like ARIMA (*6*) or Exponential Smoothing (*7*) are trained to represent this time series as good as possible and then used to calculate future values.

2.2 The Challenges of Big Data

Changes in the way of data collection, towards more time series that are recorded automatically on increasingly fine structural and temporal granularities, lead to new requirements regarding the time series forecasting (*3, 8*). For example, in the energy domain, the energy consumption of individual customers is monitored in a 15 minutes granularity. At such a fine granularity, the operation times of domestic appliances, which are unique per household and vary on a daily basis, have to be taken into account (*9*). This makes accurate forecasts of those energy demands much more complicated. On a more coarse-grained granularity, e.g., daily or even weekly energy consumption, these fluctuations are hidden by aggregation effects. The energy consumption of domestic appliances is still part of daily energy consumption but it is not important when the energy was consumed. Based on many observations like in this example, we formulate three general challenges that are caused by the Big Data trend and present in many (if not all) domains.

Data Volume A continuously growing number of time series is monitored at very fine temporal granularity (*10*). Thus, a large number of time series has to be modeled and the creation of every individual model becomes more time consuming since more training data is available. This makes the application of models representing only one individual time series very difficult, especially since not all application domains can tolerate a time-consuming model creation.

Incomplete Data The individual time series originate from a multitude of data sources, e.g., Smart Meters of individual households and enterprises. Technical malfunctions of these monitoring devices during data collection and transmission can lead to missing values and incomplete time series which the majority of forecast techniques is not able to work with (*11*). When the focus lies on only a few time series, in many cases it is possible to ensure complete data, e.g., by the application of imputation methods or compensation values from similar time series (*12*). However, regarding data sets with thousands of time series, this is associated with very high costs or might not be feasible at all from a statistical perspective.

Noisy Behavior Time series are monitored at increasingly fine structural and temporal granularity. Such fine-grained time series are prone to noisy behavior (*5*) that originates from external influences which can hardly be monitored and can only be observed at a fine granularity. On more coarse-grained monitoring granularities, these fluctuations are masked by aggregation effects. Time series with this kind of irregular behavior are especially hard to model and to forecast since they do not seem to be deterministic.

Some of the related literature already addresses these challenges. For example, **V**ector **A**uto**R**egression (VAR) models the influences of several time series on one another (*13*). In doing so, the model uses the information of other series to compensate for the noisy behavior of

individual ones and improves the forecast accuracy. Another noteworthy forecast technique is Croston's Method (*14*). This approach explicitly models the gaps in a time series and uses this information during the forecast calculation. However, none of the existing approaches covers all of the aforementioned challenges nor are they able to combine (*15*).

In order to overcome this lack of a scalable forecast technique for large-scale time series data sets, we developed CSAR, the **C**ross-**S**ectional **A**uto**R**egression model (*3*). This model assumes that time series that behave similarly can be represented by the same model. Thus, in contrast to other forecast techniques, CSAR represents many time series with the same model. This not only solves the problem of forecasting many time series in short time. It also improves the forecast accuracy of the often noisy and incomplete time series.

So far, CSAR has always been always applied on a data set level assuming all time series from the same domain follow a common behavior. However, experiments show this assumption only holds to a certain degree. There are always individuals or groups of time series showing a deviant behavior. Therefore, in this work, we use time series features to identify time series that exhibit the same statistical behavior and are thus best modeled together. We represent each of these groups with an individual CSAR model that represents the underlying time series better than a model for the whole data set.

In the following sections, we first briefly describe CSAR and how it addresses the aforementioned challenges on forecasting large-scale time series data sets (Section 3). Afterwards, we describe our feature-based partitioning approach that divides a data set into several partitions of similar time series each represented by an individual CSAR models (Section 4).

3 Cross-sectional AutoRegression Model (CSAR)

CSAR is a forecast technique specifically designed to address large-scale time series data sets (*3*). It is based on the concept of cross-sectional forecasting (*8*) assuming time series from the

same domain follow a common behavior and therefore models all time series of a data set with a single common model compensating the effects of incomplete and noisy data. Incomplete series are ignored during the model training but may be forecast with the model trained on all other series of the same data set. Outlier and noisy fluctuations of several time series neutralize one another such that a good model fit can be achieved. CSAR builds upon this paradigm and extends it by allowing a much wider training data selection. The model consists of three components: *Integration*, *Autoregression*, and *Error Terms*. The integration component is a preprocessing step making the time series of the data set stationary. Leaving Autoregression and Error Terms as the actual modeling components.

3.1 Integration

The integration component eliminates trend and seasonal effects from the time series and make them stationary. When integration is used, all time series of the data set are differentiated, subsequently forecast, and finally integrated. Each time series is differentiated individually, such that they do not influence one another. Differentiation does not involve any kind of model creation. Therefore, this approach is feasible for large data sets. Equation (1) shows the trend elimination using discrete differentiation:

$$y'_t = \frac{y_t - y_{t-d}}{d}. \tag{1}$$

The value y'_t of the differentiated time series is calculated by the difference of the time series values y_t and its most recent predecessor y_{t-d}. d denotes the time distance to the predecessor. If there are no missing values, the direct predecessor of y_t is used ($d = 1$). If values are missing, d is increased to use the most recent available predecessor. The division by d is necessary to represent the actual trend from one period to the next one. Seasonal differentiation is used to

eliminate reoccurring seasonal patterns, as shown in (2):

$$y'_t = y_t - y_{t-D \cdot s}. \tag{2}$$

The value y'_t of the differentiated time series is the difference of the time series values y_t and its most recent seasonal predecessor y_{t-s}. The value of s denotes the seasonality of the data set. D denotes the number of missing seasonal predecessors. Both equations show the first degree of differentiation. If a higher degree of differentiation should be used, the differentiation is applied iteratively.

3.2 Autoregression

The autoregressive component of CSAR is closely related to that of ARIMA and calculates forecasts based on historical time series values. The main difference is that CSAR is based on the concept of cross-sectional forecasting (*8*), applying the model on so-called cross-sections instead of individual time series as ARIMA. A cross-section $\vec{y}_t$ is a time slice containing the values of all time series of a data set at a certain point in time t. Again, there is a non-seasonal and a seasonal case. The non-seasonal autoregression is shown in (3):

$$\widehat{\vec{y}}_{t+1} = c + \sum_{i=1}^{p} \phi_i \cdot \vec{y}_{t-i+1}. \tag{3}$$

$\widehat{\vec{y}}_{t+1}$ denotes the forecast values for all time series of the data set, which are calculated as the sum of the historical cross-sections $\vec{y}_t, \ldots, \vec{y}_{t-p+1}$ weighted with the model parameters $\phi_1, \ldots, \phi_p$. The metaparameter p denotes the number of cross-sections used for the forecast calculation. The seasonal case shown in (4) has a similar structure:

$$\widehat{\vec{y}}_{t+1} = c + \sum_{j=1}^{P} \Phi_j \cdot \vec{y}_{t-j \cdot s+1}. \tag{4}$$

Again, forecasts are calculated based on historical cross-sections weighted with the model parameters $\Phi_1, ..., \Phi_P$. In the seasonal case, seasonal predecessor cross-sections are used for

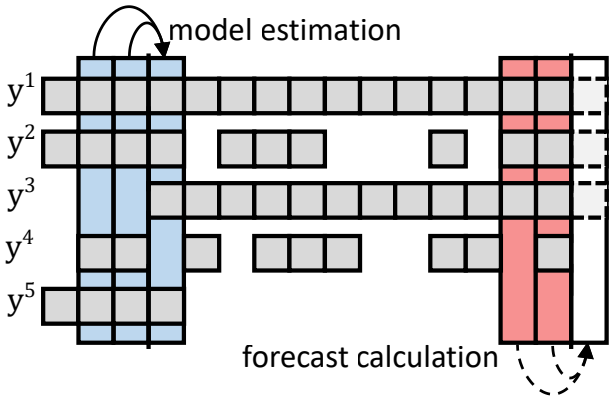


Figure 2: CSAR model with two non-seasonal AR components.

the forecast calculation. P denotes the number of seasonal cross-sections used. The model parameters are optimized on all available time series which have values in all required cross-sections. Thus, the model represents all time series of the data set. Additionally, there is a scalar constant c, which is also optimized during the model training.

The example in Figure 2 shows a simple CSAR model with two non-seasonal autoregressive model components. In this case, there are five time series $y_1, \ldots, y_5$ where each gray square represents a single time series value. The vertical colored rectangles represent the cross-sections. Blue cross-sections represent training data, the data used for the forecast calculation is highlighted in red. The rightmost cross-section is the target time slice that should be forecast. The data in this small example has a seasonality of twelve values, as it would be typical for monthly data. Therefore, the model training is located exactly one season before the target cross-section. Since the example shows an AR(2) model, two cross-sections are used as input for the model training. All time series with values in all three blue cross-sections contribute to the model training. All other time series are ignored, in this case y_3 and y_4. The trained model is applied to the most current data. For every time series with values in all red cross-sections a forecast value can be calculated, here y_1, y_2, and y_3.

3.3 Error Terms

The Error Terms are used to compensate systematic mispredictions for time series not following the behavior of the majority of time series in the data set. This is accomplished by forecasting some of the available historical cross-sections with the autoregressive model components and calculating the errors of these forecasts. Again, there is a non-seasonal and a seasonal case. In the non-seasonal case, the most recent cross-sections $\vec{y}_t, \vec{y}_{t-1}, \ldots$ are forecast. In the seasonal case, the seasonal predecessor cross-sections to $\vec{y}_{t+1-s}, \vec{y}_{t+1-2s}, \ldots$ are forecast.

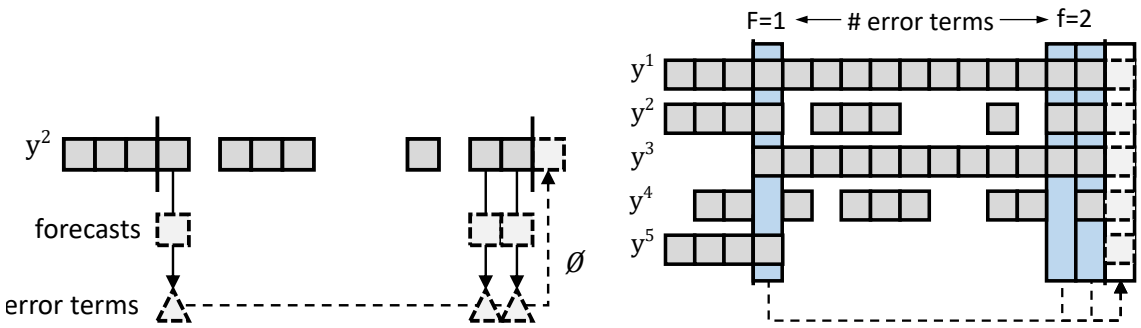
$$e_t = y_t - \widehat{y}_t \tag{5}$$

$$\overline{e^n_{t+1}} = \frac{1}{f+F}\left(\sum_{i=1}^{f} e^n_{t-i} + \sum_{j=1}^{F} e^n_{t-j\cdot s}\right) \tag{6}$$

$$\widehat{y}^n_{t+1} = c - \overline{e^n_{t+1}} \tag{7}$$

Equation (5) shows the calculation of the forecast error e_t at time t as the difference of the real time series value y_t and the corresponding forecast $\widehat{y}_t$. Equation (6) shows how the average error $\overline{e^n_{t+1}}$ for time series n at time $t+1$ is calculated. The first term within the parenthesis collects the non-seasonal forecast errors, the second term collects all seasonal forecast errors. f denotes the number of non-seasonal error terms and F the number of seasonal error terms. Finally, (7) shows how the error is incorporated into the forecast calculation by subtracting it from the constant c or the forecast calculated by a purely autoregressive CSAR model.

The example in Figure 3 shows a CSAR model with one seasonal and two non-seasonal error terms. The upper part, Figure 3(a), shows the application of the error terms on one example time series. Here the forecasts for historical time series values are calculated, represented by dashed squares under the actual time series. For these forecast values, the corresponding forecast errors are calculated, averaged, and used to correct the final forecast for the time series. Figure 3(b) shows the application of the error terms on the data set level. For each time series, the error terms are calculated individually but at the same points in time, highlighted with blue rectangles.



(a) Error terms on one series. (b) Error terms on a whole data set.

Figure 3: CSAR model with one seasonal and two non-seasonal error terms.

If a time series has missing values such that either the forecast or the error calculation is not possible, the affected values are ignored during the averaging step. In the example, this is the case for time series y_3 and y_4. Both are corrected based on only a single error term.

CSAR is specially designed to work with large-scale time series data sets based on the concept of cross-sectional forecasting in order to represent a set of similar time series with the same model. This approach allows CSAR to calculate forecast values for thousands of time series with very low overhead. Furthermore, using information of many time series helps to compensate missing values and noisy behavior of individual series. Incomplete time series are ignored during the model training but can nonetheless be forecast with the model created using the information of many other time series. Noisy series that would end up with a bad model fit when forecast individually are forecast much more accurately since the irregular fluctuations are smoothed by the information of other series.

4 Time Series Partitioning

The assumption that all time series of a data set exhibit the same behavior only holds to a certain degree. To fit a CSAR model to a subgroup of time series or *partition* can be beneficial for several reasons: (1) there are only a few models to compute, (2) they are fast to compute,

and (3) they may improve the forecast accuracy compared to one monolithic CSAR model.

In this section, we present a partitioning technique efficiently identifying partitions for CSAR models and taking the challenges of big data into account. First, we give an overview of partitioning techniques for time series, and secondly, we propose our technique for CSAR.

4.1 State of the Art

Time series partitioning is an unsupervised data-mining task. It assigns unlabeled time series to groups such that each group is homogeneous, i.e., the distance of time series within a group is minimal, while the distance of time series from different groups is maximal (*16*). Partitioning techniques are suitable for observations from a low-dimensional space consisting of a few values. However, time series are inherently high-dimensional due to their length. Their discriminative characteristics arise from many values which depend on each other. Therefore, it is necessary to prepare such data by accurately representing them in a low-dimensional space.

In the literature exists a wide body of partitioning techniques (*17*). Subsequently, we give an overview of these techniques and discuss them regarding the challenges of large-scale time series data and our use case.

Raw-data-based techniques take a time series as-is and do not represent them in a low-dimensional space. Instead, they focus on various distance measures such as lock-step distance measures, elastic distance measures, and correlation-based distance measures (*16, 17*). Thus, they focus on a similarity of time series *in shape*, i.e., on a similarity in local, time-dependent characteristics. Overall, raw-data-based techniques calculate the distance on a high-dimensional representation which is why they do not tackle the data volume challenge. As noted by (*18*), they are sensitive to missing or noisy data.

Shape-based techniques refer to representations which are reduced in the time domain, by selecting random or salient values, by aggregating values segment-wise, or by discretizing

values (*19*). They focus on a similarity in shape, too. Recently, a piecewise aggregate approximation, which represents values segment-wise by their mean value, has been used for partitioning (*20*). Overall, shape-based techniques can handle large data volumes and noise. However, they do not consider missing data.

Model-based techniques focus on the cause of a time series and express it as a generative time series model. After identifying the model and its metaparameters, model parameters are estimated and form the representation. These techniques focus on a global, time-independent similarity *in structure*, which is desirable for our use case. ARIMA models and Markov models are common examples (*17*). The class of model-based techniques works with noisy data, some techniques even handle missing data. Markov models are efficiently calculated but have large models. On the other hand, estimating model parameters for an ARIMA model can be time-consuming.

Feature-based techniques focus on certain properties of the analyzed data, which are often use case specific. A *feature* is a global time series property. It arises from the application of a method from time series analysis, which characterizes a time series as a single real value (*21, 22*). A *feature vector* gathers these characteristics and thus, represents a time series in a low-dimensional space and compares them regarding their similarity in structure. Most important features arise from the value distribution and correlation, time series components, stationarity, and the frequency domain (*21*). They all represent time series efficiently, as they only require a few passes over its values. Moreover, they are less sensitive to noise, and they are robust with respect to missing data (*18*).

A feature-based representation best tackles the challenges mentioned above and focuses on a similarity in structure which is desirable for CSAR modeling. Besides the representation technique, partitioning also depends on the selected distance measure as well as the selected

partitioning technique. In most cases, the Euclidean distance assesses the similarity of features from the categories mentioned above (*18, 23–25*). Therefore, we adopt this distance measure for our technique. Researchers have been assessing several partitioning techniques from different groups (*17*). We select a density-based technique based on our observation that it provides a useful partitioning for our feature-based representation.

4.2 Feature-based Partitioning for CSAR

A CSAR model consists of a set of metaparameters (p, P, q, Q, $constant$) and applies it for the model estimation. It assumes that the time series in a partition exhibit a common behavior, i.e., they have the same degree of autoregression and systematic misprediction.

Through the use of *correlation features*, we are able to partition a data set with respect to this behavior. Originally, correlation features are used for ARIMA model identification. Therefore, we first introduce the features based on this application. Second, we apply them to CSAR for partitioning.

4.2.1 ACF and PACF for ARIMA

ARIMA modeling needs an identification of metaparameters for each time series, using the *autocorrelation function* (ACF) and the *partial autocorrelation function* (PACF) for the determination of the order of the moving average as well as the autoregressive process (*6*).

The ACF expresses the linear dependence of a time series on itself, shifted by a sequence of lags. For example, a lag of 1 expresses the linear dependence between a time series with itself lagged by one period. It measures the linear predictability of a value y_t using the value y_{t-1} and is formally defined by:

$$acf_1 = \frac{\sum_{t=1}^{T-1}(y_{t+1} - m)(y_t - m)}{\sum_{t=1}^{T}(y_t - m)^2}, \tag{8}$$

where m is the sample mean value of the time series. Autocorrelation values range between

-1 and $+1$: $acf_1 = 0$ therefore reflects that there is no linear dependence between a value y_{t-1} and its successor y_t. If $acf_1 > 0$, then there is a positive linear dependence between the values. The maximum correlation, $acf_1 = 1$, implies a perfect predictability of y_{t-1} by y_t. If $acf_1 < 0$, there is a negative linear dependence between successive values: if y_{t-1} increases, y_t decreases. This also leads to perfect predictability, if $acf_1 = -1$. With ACF, ARIMA determines the order of a moving average process. The first q ACF lags of a moving-average process are nonzero while subsequent lags are zero.

The PACF expresses the partial autocorrelation of a time series on itself, shifted by a sequence of lags. In contrast to ACF, the dependence of periods between the lag is removed. For example, $pacf_2$ expresses the dependence of y_t from y_{t-2} which is corrected by the effects of y_{t-1}. The partial autocorrelation $pacf_1$ expresses the dependence of y_t from y_{t-1} which is equal to acf_1 since there are no periods between t and $t+1$. With PACF, ARIMA determines the order of an autoregressive process. The first p PACF lags of an autoregressive process are nonzero while subsequent lags are zero.

4.2.2 ACF and PACF for CSAR

ARIMA and CSAR are closely related. The autoregressive process of ARIMA is the blueprint for the autoregressive part of CSAR and the moving-average process serves the same purpose as the the error terms of CSAR. ACF and PACF advises ARIMA regarding these components. We hypothesize that these features are able to identify time series with similar behavior such these time series may be grouped into the same partition. Interestingly, ACF and PACF may be calculated efficiently and they provide reasonable results for incomplete time series which makes them suitable for large-scale time series data sets.

Let p_{max}, P_{max}, q_{max}, and Q_{max} be the maximum number of non-seasonal and seasonal cross-sections as well as non-seasonal and seasonal error terms that are assessed during model

Table 1: Correlation features for CSAR.

Component	Features	Indices
Non-seasonal cross-section	$pacf_k$	$1 \leq k \leq p_{max}$
Seasonal cross-section	$pacf_{k \cdot s}$	$1 \leq k \leq P_{max}$
Non-seasonal error term	acf_k	$1 \leq k \leq q_{max}$
Seasonal error term	$acf_{k \cdot s}$	$1 \leq k \leq Q_{max}$

validation. The correlation features as given in Table 1 build up a feature-based representation for neighboring time series with similar autoregression and misprediction characteristics.

The partitioning is carried out as follows. The feature-based representation is reduced into two dimensions by t-distributed stochastic neighbor embedding (t-SNE) (*26*). For each pair of embedded representations, the Euclidean distance is calculated and stored in a distance matrix. Based on this distance matrix, a density-based spatial clustering (DBSCAN) is carried out in order to discover partitions of the data set (*27*). With kNN-distplot, the DBSCAN parameters (minimum distance ϵ and minimum points $minPoints$) are appropriately configured.

5 Evaluation

CSAR is compared to commonly used forecast techniques in order to examine the following hypotheses: first, CSAR improves the forecast accuracy for a large number of time series, second, it has a lower calculation time for identifying metaparameters, and third, it has a lower calculation time for model estimation and providing forecasts. Moreover, these hypotheses are tested for feature-based partitioning.

5.1 Experimental Setting

This section details the experimental setting, i.e., the forecast techniques that are applied, the data sets that are selected, the output variables that are measured, and the configurations for representation, partitioning, and forecast techniques.

R (*28*) provides efficient built-in functions for model parameter estimation and traditional forecast techniques. We also rely on these techniques and evaluate our approach in R. The experiments are executed on a server machine with a Twelve-Core Intel(R) Xeon(R) Gold Processor 6136@3.0GHz and 64GB of RAM.

5.1.1 Data Sets

We use two real-world data sets from the energy and the economic domain.

Energy The Energy data set is the result of the Smart Metering Project initiated by the Irish Commission for Energy Regulation (*29*). It contains the electricity consumption of 6,433 households, small or medium-sized businesses, as well as unclassified entities in Ireland between July 2009 and December 2010 measured in kilowatt hour at a half-hour granularity. Compared to a complete data set, 5% of the data is missing. Each time series is aggregated to a 6-hour granularity, i.e., to a length of 2,144. All time series exhibit daily, weekly, and yearly seasonal components. We evaluate our techniques on the weekly season.

Payment The second data set is taken from the IJCAI-2017 Data Mining Contest (*30*). It consists of payment transactions of 2,000 distinct shops in daily granularity monitored over 494 days. The shops are classified into food stores, supermarkets, entertainment stores, health stores, beauty stores, and other shopping entities. None of the time series has a complete history which means that every time series is either not monitored right from the beginning or has missing values in its history. Compared to a complete data set, 40% of the data is missing. The data set exhibits a weekly season which we use for the evaluation.

5.1.2 Forecast Techniques

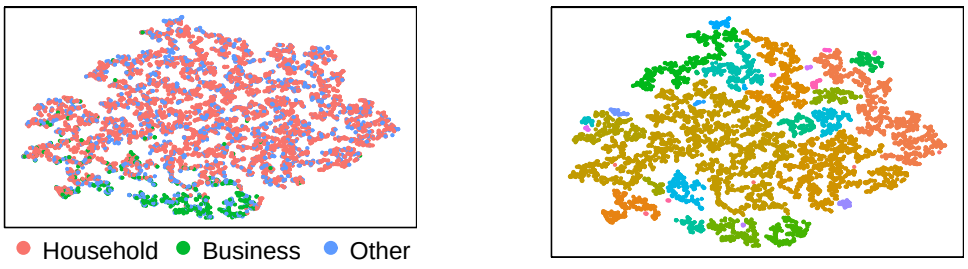
The following forecast techniques are compared with each other:

Nave (N) We include the nave forecast where every predicted period has the same value as its predecessor: $\hat{y}_t = y_{t-1}$. This forecast reflects the baseline of our evaluation: if a forecast technique performs worse than the nave forecast, it is obviously not suited for the given data set.

Nave Seasonal (NS) If a time series is seasonal, then a value is highly correlated to the value from the previous season. Based on the observation, we include the nave seasonal forecast as a second baseline: $\hat{y}_t = y_{t-s}$. Like the nave forecast, it should be a lower bound for more advanced forecast techniques.

ARIMA (A) As traditional forecast technique, we select an ARIMA model as implemented in R (*28*). We use *auto.arima* from the *forecast* package (*31*) for model identification and *arima* from the *stats* package for model estimation and forecasting. In compliance with CSAR, we set the maximum order to the same number of cross-sections and error terms (p_{max} and q_{max}). A season component is a assumed with a maximum order in compliance with the seasonal cross-sections and error terms of CSAR (P_{max} and Q_{max}). A (constant) mean value is assumed but not a trend drift. Missing values in a time series are handled by this implementation. If identification or estimation fails, the forecast for this period is missing.

CSAR with a Single Partition (1C) CSAR with "one global" partition only trains one single monolithic CSAR model for a complete data set which in turn, provides a forecast for all time series simultaneously.



(a) Class-based (b) Feature-based

Figure 4: Partitioning of Energy data set.

CSAR with Class-based Partition (CC) The selected data sets provide class labels that we exploit for creating partitions. Each partition identifies and models an individual CSAR model.

CSAR with Feature-based Partition (FC) We use our feature-based representation and assume a maximum order of 2 for the ACF and PACF features. A DBSCAN partitioning with $\epsilon = 1.5$ and $minPts = 3$ is applied. Each partition identifies and models an individual CSAR model.

Figure 4 displays the feature-based representation. The class-based partitioning (Figure 4(a)) shows that households (red points) and small- or medium businesses (green points) have a different behavior which is why they are not overlapping. However, the unclassified entities (blue points) are spread across the feature space. The feature-based partitioning (Figure 4(b)) leads to 33 partitions, each represented by a different color, plus one partition for outliers. On the Payment data set (not shown), the feature-based partitioning leads to 37 partitions plus one partition for outliers.

For a comparison with other traditional forecast techniques, i.e., Triple Exponential Smoothing, Vector Autoregression, Croston's Method, and Hierarchical Forecasting, please refer to the evaluation of (*15*) where the authors show that a monolithic CSAR model already achieves a higher forecast accuracy than these techniques.

Table 2: Metaparameter intervals.

	p	P	q	Q	constant
Metering	[0, 2]	[0, 2]	[0, 2]	[0, 2]	[False, True]
Payment	[0, 2]	[0, 2]	[0, 2]	[0, 2]	[False, True]

Table 3: Intervals for training, validation, and use.

	Training	Validation	Use
Metering	[1, 2088]	[2089, 2116]	[2117, 2144]
Payment	[1, 438]	[439, 466]	[467, 494]

5.1.3 Model Validation and Use

Model validation is carried out for CSAR in order to determine the best metaparameters for each model. For all partitions, all metaparameter combinations (Table 2), and all validation periods (Table 3), a model is estimated returning a one-step ahead forecast for each time series in its partition $\hat{y}_t, T+1 \leq t \leq T+V$, where T and V are the lengths of the training and validation intervals, respectively. The metaparameter combination providing the best forecast accuracy for its cluster is selected for model use. ARIMA selects the best metaparameters based on the recommendation of *auto.arima* on the training and validation interval $y_t, 1 \leq t \leq T+V$.

Model use is carried out in order to assess the forecast accuracy of all forecast techniques. For each periods, a model is estimated based on the selected metaparameter combination. It results in a vector of one-step ahead forecasts $\hat{y}_t, T+V+1 \leq t \leq T+V+U$, where U is the length of the model use interval (Table 3).

5.1.4 Output Variables

The forecast accuracy is assessed with a relative SAPE error measure (Symmetric Absolute Percentage Error):

$$SAPE(y_t, \hat{y}_t) = \frac{|y_t - \hat{y}_t|}{(|y_t| + |\hat{y}_t|)/2} \cdot 100\%, \tag{9}$$

where y_t is the real value and $\hat{y}_t$ the corresponding forecast. A relative error measure is easier to interpret and to compare than an absolute error measure. Moreover, SAPE has a defined range (0% to 200%). If a y_t or $\hat{y}_t$ are not defined, SAPE returns 200%. Averaging several SAPE errors results in the Symmetric Mean Absolute Percentage Error (SMAPE):

$$SMAPE(y, \hat{y}) = \frac{1}{T} \sum\nolimits_t SAPE(y_t, \hat{y}_t) \tag{10}$$

The forecast accuracy is assessed on the base and top aggregation levels. On the base level, every time series is assessed individually. On the top level, all time series are aggregated for each period: $\sum_n y_t^n$. Sticking to the example from the energy domain, the base level represents the energy consumption of each individual user while the top aggregation level represents the overall energy consumption of all customers of an energy provider. Both are relevant applications in the energy sector.

Model validation takes these output variables into account, i.e., for an assessment of an SMAPE accuracy on the top level, metaparameters are selected providing the best accuracy on the top level in the validation interval.

The calculation time is given in seconds. For all forecast techniques, we only report the time of the single-threaded execution. Since ARIMA is very time-consuming, we measure the calculation time for a representative sample of time series and extrapolate it for the complete data set. The parallel execution carried out on all time series for assessing the forecast accuracy confirms that this extrapolation is accurate.

5.2 Results and Discussion

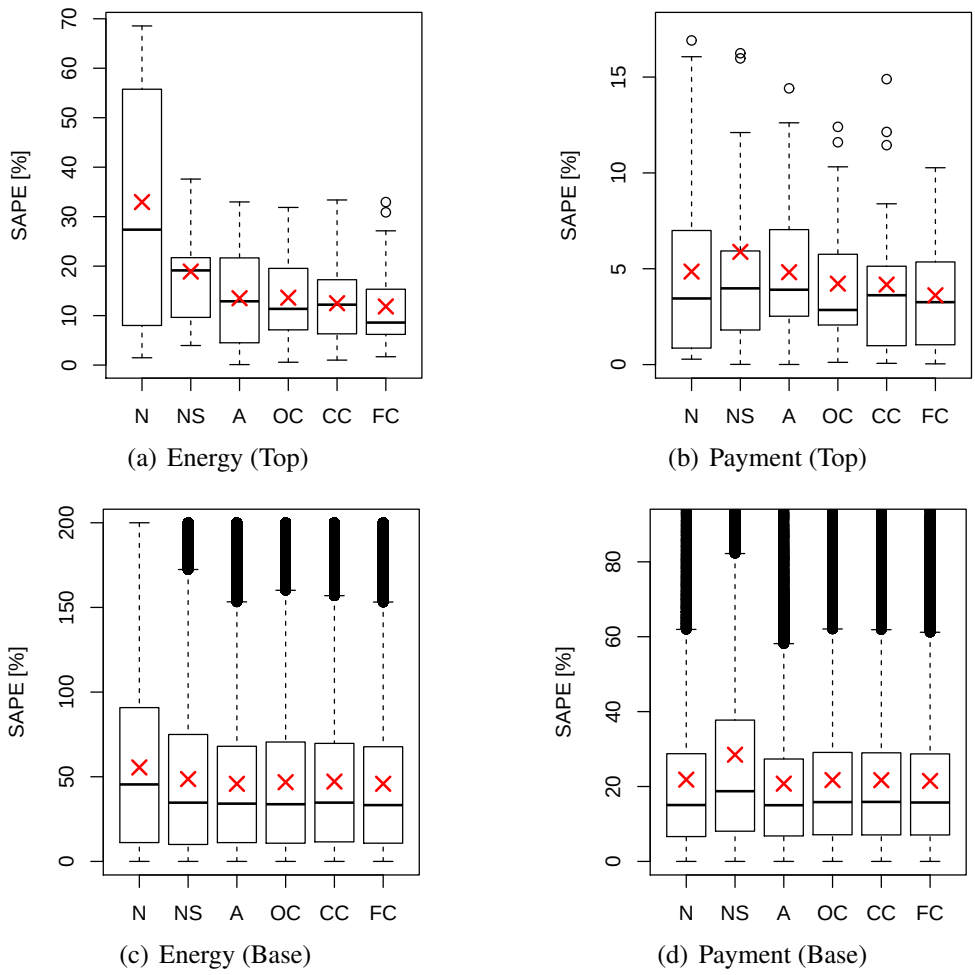
The results of our evaluation are presented and discussed in the order of our hypotheses: the forecast accuracy is assessed, followed by the evaluation of the calculation time for model validation and use.

5.2.1 Forecast Accuracy

Figure 5 shows the forecast accuracy of the data sets on the top level and on the base level, respectively. The boxplot shows the distribution of forecast errors as SAPE, while the red cross displays the mean error as SMAPE.

On the top level of the Energy data set (Figure 5(a)), the nave forecasts (N and NS) have an SMAPE of 32.9% and 18.9%. With 13.5%, ARIMA (A) is more accurate. The monolithic CSAR (1C) is as accurate as ARIMA with 13.6%. Partitioning significantly improves the accuracy. Class-based partitioning (CC) has an error of 12.5%. The feature-based partitioning (FC) provides an even better partitioning which reaches 11.8%. This behavior is confirmed on the Payment data set (Figure 5(b)). The nave forecasts provide an SMAPE of 4.9% (N) and 5.9% (NS), respectively. Thus, the value of the previous season is not a good guess for this data set and the most recent value provides a smaller error. ARIMA (A) is only slightly better with 4.8%. CSAR modeling revealed an overall improvement compared to these forecast techniques. Monolithic CSAR (1C) provides an error of 4.2%. There is no significant difference with respect to class-based partitioning (CC) that also provides 4.2%. Interestingly, feature-based partitioning (FC) provides a strong improvement with 3.6%, underlining that class labels do not necessarily reflect the characteristics for accurate partitioning.

The evaluation on the base level shows different results. The forecast errors are generally higher since individual time series are much more noisy than aggregates and therefore harder to predict. Those noisy fluctuations eliminate each other during aggregation, therefore predictable behavior is much easier to identify and model. Moreover, there is no significant difference between the traditional forecast techniques and CSAR on the base level. The error on the Energy data (Figure 5(c)) decreases from 55.5% and 48.6% (N and NS) to 45.8% (A). 1C and CC provide results that are slightly less accurate than ARIMA (46.8% and 47.2%). The feature-based partitioning (FC) has an accuracy similar to ARIMA (45.9%). The Payment data



(a) Energy (Top) (b) Payment (Top)

(c) Energy (Base) (d) Payment (Base)

Figure 5: Forecast accuracy.

set confirms this behavior (Figure 5(d)). While the nave forecasts have an SMAPE of 21.8% (N) and 28.5% (NS), reconfirming the observation of the top level, ARIMA (A) is slightly more accurate with 20.8%. The evaluation does not reveal an improvement of CSAR over ARIMA, with an SMAPE ranging between 21.7% (1C and CC) and 21.4% (FC).

Overall, our experiments show that CSAR is at least as accurate as ARIMA on the base level and even more accurate on the top level. Furthermore, the forecasts are calculated much faster as the next experiment will show.

Table 4: Calculation time per period in seconds.

	Energy		Payment	
	Validation	Use	Validation	Use
A	163,668	141,968	3,792	582
1C	69 ± 4	0.29 ± 0.03	47 ± 4	0.10 ± 0.03
CC	108 ± 3	0.62 ± 0.03	168 ± 3	0.59 ± 0.09
FC	846 ± 7	3.63 ± 0.14	$1{,}186 \pm 8$	3.40 ± 0.27

5.2.2 Model Validation Time

To identify ARIMA metaparameters, *auto.arima* needs on average 25 seconds per time series. This sums up to 163,668 seconds for all Energy time series (Table 4). For CSAR, we assess all metaparameter combinations based on their one-step ahead forecast in the validation interval. For a single period and all time series, the validation of 1C takes on average 69 seconds with a standard deviation of 4 seconds. Splitting a data set into partitions leads to a higher validation time of 108 seconds (CC) and 846 seconds (FC) with a similar standard deviation. CC splits the dataset into three partitions but the validation time does not triple compared to 1C. Therefore, smaller partitions need less validation time but partitioning itself incurs some overhead compared to 1C. This behavior is confirmed on the Payment data set. ARIMA (A) only needs 1.9 seconds per time series, thus, 3,792 seconds for all time series and one period. It is faster than the validation time on the Energy data set due to the shorter time series length. Monolithic CSAR, class-, and feature-based partitioning (1C, CC, FC) only need 47 to 1,186 seconds because they validate all time series in a partition at once. Overall, identifying metaparameters for CSAR, is one order of magnitude faster than ARIMA, even on long validation intervals.

5.2.3 Model Use Time

For model use, the best configuration per time series (A) and per partition (1C, CC, FC) is selected for model estimation. Estimating an ARIMA model (A) for each Energy time series requires on average 22 seconds and adds up to 141,968 seconds ($\approx$40h) (Table 4). Estimating

a model for CSAR for a single period only requires 0.29 seconds (1C). Class- and Feature-aware partitioning incurs some overhead since one CSAR model per partition is estimated. However, even with partitioning, CSAR outperforms ARIMA model estimation by two orders of magnitude. This behavior is confirmed by the Payment data set where modeling an ARIMA model (A) takes 0.29 seconds which adds up to 582 seconds for one data set. This is faster than the calculation time on the Energy data set due to the shorter time series length. Again, monolithic CSAR, class-, and feature-based partititioning (1C, CC, FC) are faster by two orders of magnitude. They are less affected by the time series length, but by the amount of partitions to model.

6 Conclusion and Future Work

We revisited the CSAR approach together with a feature-aware partitioning and showed its capability to forecast large time series data sets more accurate and by two orders of magnitude faster than traditional approaches. These findings are experimentally confirmed on two real-world data sets. Moreover, the feature-aware partitioning groups similar time series whose CSAR models improve the overall forecast accuracy even more while maintaining a low calculation time. Overall, we have found novel techniques to tackle the challenges of large-scale time series forecasting. Future work will concentrate on partitioning schemes based on learned features to utilize the accuracy from forecast validation as feedback for assessing the partitioning quality.

Acknowledgments

This work is partly funded (1) by the European Regional Development Fund (ERDF) under co-financing by the Free State of Saxony (100320127) and Systema GmbH, and (2) within the European Union's Horizon 2020 research and innovation program under grant agreement No 731232.

References

1. J G De Gooijer and R J Hyndman. 25 Years of Time Series Forecasting. *Int J Forecast*, 22(3):443–473, 2006.

2. T M McCarthy, D F Davis, S L Golicic, and J T Mentzer. The Evolution of Sales Forecasting Management. *J Forecast*, 25(5):303–324, 2006.

3. C Hartmann, F Ressel, M Hahmann, D Habich, and W Lehner. CSAR: the cross-sectional autoregression model for short and long-range forecasting. *Int J Data Sci Anal*, 2019.

4. K Kambatla, G Kollias, V Kumar, and A Grama. Trends in big data analytics. *J Parallel Distrib Comput*, 74(7):2561–2573, 2014.

5. H Hassani and E S Silva. Forecasting with Big Data: A Review. *AODS*, 2(1):5–19, 2015.

6. G E P Box, G M Jenkins, and G C Reinsel. *Time series analysis forecasting and control*. Wiley, 2008.

7. C C Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *Int J Forecast*, 20(1):5–10, 2004.

8. C Hartmann, M Hahmann, W Lehner, and F Rosenthal. Exploiting big data in time series forecasting: A cross-sectional approach. In *Proc of DSAA*, 2015.

9. B Neupane, T B Pedersen, and B Thiesson. Towards Flexibility Detection in Device-Level Energy Consumption. In *Workshop Proc ECML PKDD*, pages 1–16, 2014.

10. Y Sakurai, Y Matsubara, and C Faloutsos. Mining and Forecasting of Big Time-series Data. In *Proc of SIGMOD*, pages 919–922, 2015.

11. J-H Bse, V Flunkert, J Gasthaus, T Januschowski, D Lange, D Salinas, S Schelter, M Seeger, and Y Wang. Probabilistic demand forecasting at scale. In *Proc of VLDB*, volume 10, pages 1694–1705, 2017.

12. VDE Verband der Elektrotechnik Elektronik Informationstechnik e.V. Messwesen Strom (Metering Code); VDE-AR-N 4400, 2011.

13. G C Tiao and G E P Box. Modeling Multiple Time Series with Applications. *J Am Stat Assoc*, 76(376):802–816, 1981.

14. J D Croston. Forecasting and Stock Control for Intermittent Demands. *J Oper Res Soc*, 23(3):289–303, 1972.

15. C Hartmann. *Forecasting Large-scale Time Series Data*. PhD thesis, Technische Universität Dresden, 2018.

16. T Warren Liao. Clustering of time series data - a survey. *Pattern Recognit*, 38:1857–1874, 2005.

17. S Aghabozorgi, A Seyed Shirkhorshidi, and T Ying Wah. Time-series clustering – A decade review. *Inf Syst*, 53:16–38, 2015.

18. X Wang, K Smith, and R Hyndman. Characteristic-Based Clustering for Time Series Data. *Data Min Knowl Discov*, 13(3):335–364, 2006.

19. Tak-Chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

20. Rui Ding, Qiang Wang, Yingnong Dang, Qiang Fu, Haidong Zhang, and Dongmei Zhang. YADING: Fast Clustering of Large-Scale Time Series Data. *Proc of VLDB*, 8(5):473–484, 2015.

21. B D Fulcher, M A Little, and N S Jones. Highly comparative time-series analysis: the empirical structure of time series and their methods. *J R Soc Interface*, 10(83), 2013.

22. B D Fulcher and N S Jones. Highly Comparative Feature-Based Time-Series Classification. *IEEE Trans Knowl Data Eng*, 26(12):3026–3037, 2014.

23. R Agrawal, C Faloutsos, and A Swami. Efficient Similarity Search In Sequence Databases. In *Proc of FODO*, volume 730, pages 69–84, 1993.

24. K-P Chan and A W-C Fu. Efficient Time Series Matching by Wavelets. In *Proc of ICDE*, pages 126–133, 1999.

25. C Goutte, L K Hansen, M G Liptrot, and E Rostrup. Feature-Space Clustering for fMRI Meta-Analysis. *Hum Brain Mapp*, 13:165–183, 2001.

26. L J P van der Maaten and G E Hinton. Visualizing Data Using t-SNE. *J Mach Learn Res*, (9):2579–2605, 2018.

27. M Ester, H-P Kriegel, J Sander, and X Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *Proc of SIGKDD*, pages 226–231, 1996.

28. R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2018.

29. The Commission for Energy Regulation. CER Smart Metering Project, 2015.

30. International Joint Conference on Artificial Intelligence. *IJCAI 2017 - Data Mining Contest*, 08.02.2017. `https://tianchi.aliyun.com/competition`.

31. R J Hyndman and Y Khandakar. Automatic Time Series Forecasting: The Forecast Package for R. *J Stat Softw*, 27(3):1–22, 2008.