Scaling Forecasting Algorithms using Clustered Modeling

Izzeddin Gur[†] **Mehmet Guvercin**[†]
Hakan Ferhatosmanoglu[†]

BILKENT-TECHREPORT

Computer Science Department
Bilkent University
Ankara, Turkey

[†] Computer Science Department, Bilkent University, Ankara, Turkey

This work is supported in part by The Scientific and Technological Research Council of Turkey under Grant EEEAG-111E217 and The Turkish Academy of Sciences.

Keywords: Scalable forecasting, time-series models, dynamic maintenance, clustered modeling, streaming data, performance, accuracy

Abstract

Research on forecasting has traditionally focused on building more accurate statistical models for a given time-series. The models are mostly applied to limited data due to efficiency and scalability problems. However, many enterprise applications require scalable forecasting on large number of data series. For example, telecommunication companies need to forecast each of their customers' traffic load to understand their usage behavior, and to tailor targeted campaigns. Forecasting models are typically applied on aggregate data to estimate the total traffic volume for revenue estimation and resource planning. However, they cannot be easily applied to each user individually as building accurate models for large number of users would be time consuming. The problem is exacerbated when the forecasting process is continuous and the models need to be updated periodically. This paper addresses the problem of building and updating forecasting models continuously for multiple data series. We propose dynamic clustered modeling for forecasting by utilizing representative models as an analogy to cluster centers. We apply the models to each individual series through iterative nonlinear optimization. We develop two approaches: The Integrated Clustered Modeling (ICM) integrates clustering and modeling simultaneously, and the Sequential Clustered Modeling (SCM) applies them sequentially. Our findings indicate that modeling an individual's behavior using its segment can be more scalable and accurate than the individual model itself. The grouped models avoid over-fits and capture common motifs even on noisy data. Experimental results from a telco CRM application show the method is efficient and scalable, and also more accurate than having separate individual models.

1 Introduction

Statistical forecasting is an essential tool for enterprise planning and budgeting. The companies often make forecasts on an attribute of interest, such as the total revenue or network traffic by modeling an aggregate time-series. Such a collective analysis provides insights on common patterns but not on understanding the customers and their needs. Customer relationship (or experience) management applications require a customer centric view, and need scalable models on multiple evolving data series. In terms of accuracy, a separate model for each individual series could be expected to perform well, as each model can be tailored for the corresponding data. However, this approach is not scalable since common forecasting models, such as SARIMA (Seasonal Auto Regressive Integrated Moving Average) [24], take nontrivial time even for a single time-series. New methods are needed to scale the forecasting models to multiple and possibly correlated data series. The models should be updated in periods with the newly coming data. The process of fitting an incremental model for the updated data needs also be scalable.

We present the problem through a telco Business Intelligence (BI) application. Predicting the future network traffic load, such as 3G connections or call volumes, is valuable for resource planning and revenue estimation for telco companies. Fortunately, the forecasts are typically performed using well-formed statistical models such as Holtz Winter [23], exponential smoothing [23, 24], and SARIMA [4, 7, 23]. The models are applied to an aggregate time-series of the total traffic on the company network. While aggregating the data makes the analysis more feasible, an effective CRM (Customer Relationship Management) approach would be to understand each customer's usage individually. For example, if the companies can model and forecast a customer's traffic, they can design personalized campaigns to improve both the customer's experience and the company revenue.

Given the complexity of most statistical forecasting models, modeling and maintaining each time series individually would not be scalable for large CRM applications. Also, while the aggregate data may provide clear trends, each individual series includes noise and local outliers that reduce the accuracy of the models. We revisit the statistical forecasting in the context of dynamic large-scale analytics. The ideal method would be accurate in forecasting, capturing correlations for a large number of time series, efficient in building models, easy to update, and scalable in accuracy and speed.

Our approach scales the forecasting algorithms through a continuous clustered modeling (CM), i.e., forming groups of data based on their model similarities. We build common forecasting model parameters on the cluster centers, and apply the representative model to each series separately. A common model for a cluster eliminates the need for building a model for every individual. Applying the model parameters to each individual is significantly cheaper than building the model, yet it still keeps the individual focus and results in accurate fits as shown in the experimental results. Following this intuition, we develop two specific algorithms: the Integrated Clustered Modeling (ICM) integrates clustering and modeling simultaneously, and the Sequential Clustered Modeling (SCM) applies clustering and modeling sequentially. We focus on the SARIMA family of models, which performs comparable to more complex methods such as neural networks in forecasting the network traffic [16, 25]. The proposed method is independent of the underlying linear or nonlinear modeling approach.

The ICM method builds SARIMA based clusters on multiple series and updates them through iterative non-linear optimization. For a single time series, the best model is obtained by minimizing the AIC (Akaike Information Criterion) over the parameters. For multiple series, one can obtain the models by minimizing the AIC of each series hence the total AIC. Identically, for co-evolving data series, we minimize the total AIC in groups instead of individually. We seek through the space of SARIMA models to group correlated series into their segments according to their evolution pattern and find the ones minimizing the total AIC starting with initial SARIMA models. As we search to minimize the AIC, we also adjust the clusters to decrease the AIC further.

The SCM method applies clustering and modeling in a two-phase manner. We cluster the time series data using a choice of representation and build a model for each cluster representative. Forecasts are performed by applying the corresponding representative model to each time series in that cluster, individually. As data evolve, updates are applied only on the parameters of the representative models. We focus on Linear Prediction Cepstrum (LPC) coefficients as our experimental evaluations show that it has high forecast accuracies compared with several other representations.

The grouped models avoid over-fits and capture common motifs even on bursty data with local outliers. We use one model for all time-series in the cluster, however, the produced forecast for each time-series is tailored for itself. Our results suggest that it may be possible to achieve the two seemingly contradictory goals: more accurate and more efficient forecasts compared to modeling each time series individually.

We discuss the related work in Section 2 and present the background in Section 3. In Section 4, we explain the proposed methodology including the specific approaches following the two methods. We present the experimental study and results in Section 5. Finally, we conclude in Section 6.

2 Related Work

In this section, we give a brief survey on time series clustering methods and forecasting methods in the literature.

Time series clustering. Time series clustering methods can be broadly categorized into three groups in terms of representations they use: raw data, features extracted from time series, and models on time series [26]. Li and Prakash propose a clustering method to identify the category of the motion from given motion sequences [17]. They note that feature based clustering does not give appealing results for motion category identification since these methods fail to capture temporal dynamics and time shifts. Their method has interpretable features that eliminates time shifts and identifies joint dynamics across the sequences.

Corduas and Piccola use AR as a dissimilarity measure for time series classification and clustering [8]. They define AR distance from ARIMA processes, and derive an asymptotic distribution of the squared AR distance to compute time series dissimilarity. Their results suggest that AR is well defined for seasonal and non-seasonal, long and short, stationary and non-stationary time series.

In the management science community, Kumar and Patel use clustering for predictive analytics in retail merchandising [15]. The method finds the number of clusters using the trade-off between decreased variance and increased bias. To calculate the similarity of time series, they use the next period forecasts and the variance instead of using historical data. Alonso et al. propose a clustering approach that considers evolution time series [3]. For dissimilarity calculations, they use the full forecast densities instead of point forecasts and the squared Euclidean distance between full forecast densities. Authors also derive an approximation for the L_2 distance between the forecast densities.

Rodrigues proposes ODAC (Online Divisive-Agglomerative Clustering) for the time series clustering problem [22]. Clusters are on the leaves of a binary tree and updated incrementally. Each leaf can be split or aggregated after testing the confidence level, which is given by the Hoeffding bound. The computation of the dissimilarity matrix of variables in a leaf is necessary only if the confidence level of that leaf exceeds the Hoeffding bound. In this incremental hierarchical clustering, time and space requirements depend on the number of variables but they are constant with respect to the number of examples.

An application-oriented approach for the data stream clustering problem is presented in [2]. The stream clustering is divided into two sub-processes. In the first sub-process, which is called the online process, summary statistics of data streams are stored periodically. In the second one, the offline process, stored summary statistics are used to explore streams in different time horizons. Statistical properties of evolving data streams are captured effectively by means of pyramidal time window and micro-clustering in the online process.

An anytime iterative incremental clustering version of partitional clustering algorithms is introduced in [19]. The authors use Haar Wavelet decomposition of time series in their clustering algorithm and increase the level of decomposition. At each iteration, they run k-Means algorithm on the increased level representation of Haar Wavelet decomposition and use final centers as the initial clusters for the next iteration.

Kalpakis et al. study clustering of time series modeled with ARIMA models [9]. They use LPC coefficients as the features of time series and show that fewer number of LPC coefficients are needed to discriminate time series when compared to the traditional distance measures. They do not focus on forecasting accuracies as they work on the goodness by silhouette coefficient and sum-of-squares error. In SCM, we utilize ways to use LPC coefficients in the context of Clustered Modeling (CM) for forecasting. Our approach uses SARIMA models to obtain LPC coefficients, and does not need to extract AR coefficients manually.

Time series forecasting. Time series forecasting methods are generally built on historical data and a modeling schema. Li et al. [18] capture the essential characteristics of the collection of time series using Linear Dynamical System (LDS) and then extract features called *fingerprints*. The proposed method gives interpretable features that can be used to forecast motion capture, sensor, and network router traffic data. Hong et al. [12] study tracking volume of terms from text corpora of conference and computational linguistics papers. They incorporate the volumes of terms into the temporal dynamics of topics using state-space models by a supervised learning system. Their system is capable of forecasting the future volume of textual terms.

A Delay Coordinate Embedding based approach is proposed in [6]. The authors use an automated non-linear forecasting for periodic and chaotic time series generated by a common physical system over separate periods of time. They use intrinsic dimensionality of time series using fractals to estimate the lag length. The data is divided into training and holdout sets to find k in k-nearest neighbor estimation. Using the k-nearest neighbors, they interpolate the data using an SVD-based interpolation and achieve superior performance over prior approaches including auto-regression.

Recently Matsubara et al. [21] introduce TriMine to find three-way patterns in complex time-stamped events and can be used to forecast future events in a web text corpora. They use the concept of M-th order tensor with topic modeling to associate each actor-object with extracted hidden topics. The approach uses different levels of granularity to catch long term and short term fluctuations. Forecasting the next volume of clicks of a user on a certain URL is achieved using topic modeling and multi-level representation of data.

Xiong et al. [27] propose a mixture of ARMA models for clustering stationary time series, mimicing the EM algorithm for Gaussian mixtures. They focus on clustering rather than forecasting and use manually extracted orders for models. Using only likelihoods to obtain models may introduce an overfit where an information criterion is more appropriate. The improved EM algorithm models iteratively from scratch to find the number of clusters. We use a linear modeling approach relative to the optimal number of clusters which scales to large data sets.

Most of the current approaches have mainly focused on building more accurate forecasting with no particular consideration on collective and continuous models. We aim a methodology to scale the forecasting algorithms through a clustered modeling approach that exploits correlation between data series and that is optimized for forecasting. The solution is general and can be used to further improve both linear and non-linear IM approaches, such as the recent ones proposed by databases and data mining community [12, 18, 21].

3 Background

In this section, we provide a technical background including the definitions used throughout this paper. A data series or time series x is defined as an ordered list of real numbers indexed by positive integers. More formally a time series x is a vector in n dimension

$$x = (x_1, x_2, ..., x_n) \tag{1}$$

where $x_i \in \mathbf{R}^a$, and n is called the length of the time series x. If $a > 1$ then it is called a multivariate time series, in the other case it is called a univariate time series. In our context we use multiple univariate time series, and the words ”time series” and ”univariate time series” are used interchangeably. We note that a time series does not necessarily have a constant length. It may be dynamic thus left-bounded and right-unbounded, or static and bounded on both intervals.

The definition of a time series using an n-dimensional vector is the simplest form of its representation. There are different representations that are more eligible for different problems. We may categorize these representations as; the transformation based models: PCA [14], SVD [10], the spectral domain models: DFT [10], DWT [7], the time domain models: ARMA [4, 24], ARIMA [4, 24], SARIMA [4, 24], GARCH [4], and the state-space models: ARMAX [4, 24]. The use of these models may vary from problem to problem, but we focus on the multiplicative Seasonal Autoregressive Moving Average (SARIMA) family of models which includes SARMA, ARIMA, ARMA, SMA, SAR, AR, MA and more generally SARIMA which we explain in detail next.

SARIMA is a widely used time domain model that has desirable theoretical and asymptotic behaviors. SARIMA family of models exploits the fact that the value of a time point in a time series can be represented by the linear combination of its past time points and the linear combination of a white noise with indexes shifted through time. The linear combinations of the past time points and white noise are formed by both periodic and non-periodic components. Following the notation in [24] we give the definitions of the operators and obtain a more compact formula for SARIMA.

Definition 1. *(Operators) The operators*

$$\phi(B) \quad = 1 - \phi_1 B - ... - \phi_p B^p \tag{2}$$

$$\theta(B) \quad = 1 + \theta_1 B + ... + \theta_q B^q \tag{3}$$

$$\Phi_P(B^s) = 1 - \Phi_1 B^s - ... - \Phi_P B^{Ps} \tag{4}$$

and

$$\Theta_Q(B^s) = 1 + \Theta_1 B^s + ... + \Theta_Q B^{Qs} \tag{5}$$

are the autoregressive operator, moving average operator, seasonal autoregressive operator and seasonal moving average operator respectively with s being seasonal period where $B^u x_t = x_{t-u}$.

A time series can be classified as being stationary, thus having a time independent mean value and/or variance, or non-stationary, thus having a time dependent mean value and variance.

Definition 2. *(Stationarity of a Time Series) A time series x_t is called stationary if the mean value and autocovariance function of x_t, don't depend on time, and autocovariance function depends only on time difference,*

$$cov(x_u, x_t) = \gamma(u, t) = \gamma(|(u - t)|) \tag{6}$$

In case of the non-stationary time series, further processing is required to remove non-stationarity to make the time series suitable for SARMA. If the variance of the time series varies with time then a power transformation like Box-Cox family of transformations may be used,

$$y_t = \begin{Bmatrix} (x_t^\alpha - 1)/\alpha \\ log x_t \end{Bmatrix} \tag{7}$$

where α is called the power of the transformation. In the other case where a time series is not stationary because of its mean value, differencing may be used to remove the non-stationarity

$$y_t = \nabla^d x_t = (1 - B)^d x_t \tag{8}$$

where d is called the order of differencing.

SARIMA family of models without the integrated part deals with stationary time series and called SARMA. Now we turn our attention to the generic SARMA model using the operators:

Definition 3. *(SARMA) A SARMA model is defined as*

$$\Phi_P(B^s)\phi(B)x_t = \Theta_Q(B^s)\theta(B)w_t \tag{9}$$

where x_t is stationary, $w_t \sim N(0, \sigma_w)$ and called Gaussian white noise, $\phi(B)$, $\theta(B)$, $\Phi_P(B^s)$, and $\Theta_Q(B^s)$ are autoregressive operator, moving average operator and their seasonal counterparts respectively.

A SARMA model is denoted by $SARMA(p, q)x(P, Q)_s$, where p, q, P, Q are autoregressive, moving average, seasonal autoregressive and seasonal moving average orders respectively. Building a model on a data refers to choosing the right number of orders and estimating the model parameters.

Parameter Estimation. There are different ways of estimating the values of model parameters. One can use Maximum Likelihood Estimation (MLE), Sum of Squares Estimation (SSE), or Conditional Sum of Squares Estimation (CSSE). In case of invertible SARMA models, all these approaches lead to optimal estimators [24].

We start with the definition of the likelihood of a SARIMA model. As $\arg\max_x f(x) = \arg\max_x gof(x)$ if g is a monotonically increasing function, instead of raw likelihood of a SARMA model we use its log transform because of its analytical tractability. To find the optimal parameters, we can maximize the log-likelihood.

Definition 4. *(Log-likelihood) The log-likelihood of a $SARMA(p, q)x(P, Q)_s$ model built on x is defined as*

$$\ell(\beta; x) = -\frac{n}{2} ln(2\pi\sigma_w^2) - \frac{1}{2\sigma_w^2} \sum_{t=1}^{n} w_t^2 \tag{10}$$

where β is the parameter vector of the model, w_t is the white noise of the underlying SARMA model, σ_w^2 is the variance of the w_t.

We can also minimize unconditional or conditional sum of squares to find the optimal parameter values.

Definition 5. *(Unconditional Sum of Squares) The unconditional sum-of-squares of a $SARMA(p, q)x(P, Q)_s$ model built on x is defined as*

$$SS(\beta; x) = \sum_{t=-\infty}^{n} \hat{w}_t^2 \tag{11}$$

where β is the parameter vector of the model,
$\hat{w}_t = E(w_t | x_1, x_2, ..., x_n)$.

If the unconditional sum-of-squares is conditioned on the initial values of the white noise, then the sum is called the conditional sum-of-squares.

Definition 6. *(Conditional Sum of Squares) The conditional sum-of-squares of a $SARMA(p,q)x(P,Q)_s$ model built on x is defined as*

$$CSS(\beta;x) = \sum_{t=p+1}^{n} \hat{w}_t^2 \tag{12}$$

where β is the parameter vector of the model,
$\hat{w}_t = E(w_t|x_1, x_2, ..., x_n)$.

Model selection. Although the parameter estimation methods seem to be enough for modeling, it is known that increasing the number of parameters always gives better models but introduce overfit. Model selection is a tradeoff between these two contradictary goals. Even though there is no best way to choose the right statistical model, Akaike Information Criterion (AIC), AIC Bias Corrected (AICc), and Bayesian Information Criteron (BIC) are well studied and widely used ways of choosing a statistical model from a set of candidate models [24]. Our algorithms are not specific to any model selection, but we will focus on AIC.

Definition 7. *(AIC) The AIC of a $SARMA(p,q)x(P,Q)_s$ model is defined as*

$$AIC(\beta;x) = -2\ell(\beta;x) + 2r \tag{13}$$

or

$$AIC(\beta;x) = n(1 + log(2\pi)) + nlog(CSS(\beta;x)) + 2r \tag{14}$$

where r is the total number of parameters present in the given SARMA model, $\ell(\beta;x)$ is the log-likelihood of x with respect to the parameter vector β, and $CSS(\beta;x)$ is the conditional sum-of-squares of the model on x with parameters β.

Given a time series x, the best model parameters are found by

$$\beta_x^{opt} = \arg\min_{\beta} AIC(\beta;x) \tag{15}$$

In Table 1, we present the symbols we used throughout the paper.

4 Proposed Methodology

We now present our method for scaling the forecasting models on co-evolving time series. The method uses a specific notion of similarity for clustering in the context of forecasting. We use a bisecting method to find the number of clusters in the data. For each cluster, we devise a representative model that minimizes the AIC values for each group of time series collectively. As the new data arrives, the representative models can be reclustered until the AIC does not decrease. Forecasts are performed by applying the representative models to each series independently, and parameters are updated incrementally as data evolve.

Real time series are noisy and react to the events resulting in local outliers. If the events are apriori known, they can be modeled easily. For example, during holidays there are more personal

Table 1: Symbols used throughout the paper

Symbol	Meaning
$\mathcal{X}$	all time series
$x = (x_1, ..., x_n)$	time series
n	size of x
a	dimension, $\mathcal{R}^a$
$\phi, \theta, \Phi, \Theta$	operators
β	parameter vector
B^u	differencing by u in time
s	seasonal period
u, t	time series indices
α	order for Box-Cox transformation
∇^d	$(1 - B)^d$
d	difference order
w_t	white noise
p, q, P, Q	orders
r	total number of parameters $p + q + P + Q$
σ_w^2	variance of white noise
$\ell(\beta; x)$	log-likelihood of β on x
$AIC(\beta; x)$	AIC of x using parameters β
$CSS(\beta; x)$	CSS of x using parameters β
$SS(\beta; x)$	SS of x using parameters β
β_x^{opt}	optimal β for x
$Error(\tau_Y; x_i, Y)$	error of x using τ_Y and Y
$\tau_{Y_i}^{opt}$	optimal parameters on $Y_i \subset \mathcal{X}$
Y_i	subset of $\mathcal{X}$
$C(F_i, Y_i)$	model cluster with model F_i, time series Y_i
e	current iteration in an algorithm
$model(Y_i, M, \beta_i)$	algorithm for optimal parameters
N	total number of time series
A_i	aggregate time series for cluster C_i
l, m	cluster indices
c_i	LPC coefficients

phone calls, and less calls made by commercial customers. Events that are ad hoc and lack a certain pattern introduce noise to an individual model. By identifying groups of time series with similar models, and fitting a common model for them, we aim to minimize their AICs by avoiding the local outliers and overfitting.

Following the above intuition, we first present our Integrated Clustered Modeling (ICM) in Section 4.1 that simultaneously builds and enhances the models while clustering. We then present Sequential Clustered Modeling (SCM) in Section 4.2 which separates the steps of developing representative models and time series clustering. For each time series, we apply the representative model by putting the parameter vector of the model and the time series to the appropriate positions in Eq. 9 and estimating the residuals. Next period forecasts for each series are derived by applying the corresponding model for each time series individually. We note that this approach of applying the model to a data series is negligible in time compared to building the model from scratch.

We analyze the behavior of multiple time series to understand whether minimizing the model errors of multiple time series individually is the best thing to do. More formally given N time series $\mathcal{X} = (x_1, x_2, ..., x_N)$, we seek whether the proposition below is true or not;

$$\sum_{x_i \in Y} Error(\tau_Y^{opt}; x_i, Y) > \sum_{x_i \in Y} Error(\beta_{x_i}^{opt}; x_i) \tag{16}$$

where $Error(\beta_{x_i}^{opt}; x_i)$ is the forecast error of time series x_i using the optimal model parameters $\beta_{x_i}^{opt}$ and
$Error(\tau_Y^{opt}; x_i, Y)$ is the forecast error of time series x_i using a common optimal model estimated on a subset Y of $\mathcal{X}$. The left hand side of the inequality represents a group of time-series modeled collectively where all the time series $x \in Y$ share the same model parameters τ_Y^{opt}. The right hand side represents the errors of individual models. As also evidenced in our experiments, instead of modeling every time series individually, modeling them in clusters decreases the total forecast error in contrast to sum of individual errors. This also decreases the time required to give each time series a successful model.

4.1 Integrated Clustered Modeling

We present a definition of model cluster as a basis of the clustering for forecasting.

Definition 8. *(Model Cluster) A model cluster $C(F, Y)$ is a set of time series Y, and a common forecasting model F with parameter vector τ_Y^{opt} where $\forall x \in Y, AIC(\tau_Y^{opt}; x)$ is minimum over all clusters.*

Based on this definition, model clustering of a set of time series $\mathcal{X} = (x_1, x_2, ..., x_N)$ is a partitioning
$\mathcal{Y} = (Y_1, Y_2, ..., Y_l)$ of these time series and a vector of forecasting models $\mathcal{F} = (F_1, F_2, ..., F_l)$ where the model F_i is a common model for all the time series in the set Y_i. The common model F_i includes model parameters and the corresponding orders. The variance of the white noise, specific to each time series, can be estimated by Eq. 9.

Analogous to a cluster center, a forecasting model F_i of a cluster is the best (closest) in minimizing the AIC. Considering a single series, we defined the best model as the one minimizing the corresponding AIC. In multiple series case, if the models are independent from each other then minimizing the total AIC will be equal to minimizing the AICs individually which would give us the best model for each series. Similar to this, our approach is to minimize the total AIC but using a set of grouped models instead of modeling these series individually. More formally τ_i^{opt} is the optimal parameters of the model F_i minimizing total AIC of the time series in the corresponding cluster, i.e.,

$$\tau_i^{opt} = \arg\min_{\tau} \sum_{x \in Y_i} AIC(\tau; x) \tag{17}$$

Also note that the optimization we introduced in Eq. 17 is not specific to SARIMA models.

Assuming that we are given a cluster $C(F, Y)$, we need to find the optimal parameter vector τ^{opt} of F for each cluster. We first find the orders using a modeling approach on aggregated time

```
Algorithm 1: $Integrated - Clustered - Modeling(\mathcal{Y}, k^{(0)}$, M)
$\{C_i^{(0)}\} \leftarrow Initialize(\mathcal{Y}, k^{(0)}, M)$
$\{C_i^{(1)}\} \leftarrow Form - Clusters(\mathcal{Y}, \{C_i^{(0)}\}_{i=1}^{k})$
$t \leftarrow 1$
$\mathcal{Y}^{(0)} \leftarrow \mathcal{Y}$
while new data points arrive do
    $\mathcal{Y}^{(e)} \leftarrow$ Extend $\mathcal{Y}^{(e-1)}$ with new data
    $\tau^{(e)} \leftarrow update - models(\mathcal{Y}^{(e)}, M, \tau^{(e-1)})$
    Forecast future data points
    $e \leftarrow e + 1$
```

series for each cluster. Given the orders, minimizing the total AIC with respect to the model parameter vector will be equal to minimizing the negative of the sum of the log-likelihoods, or the (un)conditional sum-of-squares in Eq. 17 and eventually the followings:

$$\psi(Y_i) = -\sum_{x \in Y_i} \ell(\tau_i^{opt}; x) \tag{18}$$

or

$$\psi(Y_i) = \sum_{x \in Y_i} CSS(\tau_i^{opt}; x) \tag{19}$$

To minimize the functions in Eq.(18, 19), we utilize an iterative nonlinear optimization algorithm. We use quasi-Newton method (BFGS) [5] as it is parameter free and relatively efficient. BFGS is based on function evaluation and the gradient of the corresponding function. Because gradient gives a relatively better direction to search towards, BFGS converges fast. Also the iterative nature of BFGS makes it easy to adapt existing models when new time points arrive. Using the existing models and data series extended with new points, we can update the model for each cluster while preserving the accuracy. Algorithm 1 shows the general outline of ICM.

The problems in how to construct and maintain proper clusters are to i) find an initial representative model for each cluster, ii) appropriately assign a time series to one of the clusters given, iii) enhance representative models for every cluster, iv) find the optimal number of clusters, v) update the representative models as new data arrive, and vi) forecast the future data points.

4.1.1 Finding Initial Representatives

We first find the initial clusters, and select a model for each cluster. To search for an initial model of a cluster, we need the number of parameters and a modeling schema. We first group time-series into k clusters using a clustering scheme (e.g., random, PAM, KMeans). We then estimate an aggregate time series for each cluster, e.g., median and mean time series. For each cluster, we build an initial model by fitting an optimal SARIMA model on the aggregate time-series minimizing its AIC. We use these models as a start and further improve the parameter values for each cluster by minimizing Eq. 17. Algorithm 2 gives the details of the initial model selection. The *Bisecting* approach (Algorithm 4) is used to find the optimal number of clusters.

Algorithm 2: $Initialize(\mathcal{Y}$,k, M, $\epsilon)$

```
Group time-series into k clusters $C_i(F_i, Y_i)$ for $i = 1, 2, ..., k$
Estimate aggregates $A_i$ for $i = 1, 2, ..., k$
$\beta_i^{opt} = \arg\min_\beta AIC(\beta; A_i)$
$\tau_i^{opt} \leftarrow model(Y_i^{(0)}, M, \beta_i^{opt})$ for $i = 1, 2, ..., k$
$\{C_i^{(0)}\}_{i=1}^{k'} \leftarrow Bisecting(C_i(F_i, Y_i)_{i=1}^{i=k}, \epsilon))$
return $\{C_i^{(0)}\}$
```

$model(Y_i^{(0)}, M, \beta^{opt})$ estimates the best model parameters minimizing Eq. 17 given a minimization schema M, time-series of the corresponding cluster $Y_i^{(0)}$, and initial models β^{opt}.

4.1.2 Forming the Model Clusters

The *Initialize* algorithm returns the optimal models for each cluster. However, as the models are updated, some of the time series may be modeled better by other cluster models than its current one. We need to assign each time series to the right cluster, and then update the representative models.

Given a set of clusters $C_1, C_2, ..., C_k$, the best approach to select the appropriate cluster for a time series x_i would be to assign a time series $x_i \in Y_l$ from C_l to C_m and then update the model parameters of C_l and C_m after the assignment. The cluster that leads to the most total AIC reduction is the new cluster of time series x_i. This approach needs to update model parameters at every consideration of every series. Assigning a new object to a cluster will change its model but this may cause an increase in the AIC of some of time series while reducing the AIC of the others. We need a fast assignment scheme that also guarantees the decrease in the overall AIC. For each time series, we search for the cluster having the minimum AIC value and reassign the time series to the new cluster found. The only requirement of our assignment is to estimate the AIC for each cluster. This approach guarantees a decrease in the total AIC. We formally prove this in the following theorem.

Theorem 1. *Given a minimization algorithm M and two clusters $C_l(F_l, Y_l)$ and $C_m(F_m, Y_m)$ having model parameters β_l and β_m respectively with $x \in C_l$, if*

$$AIC(\beta_l; x) > AIC(\beta_m; x) \tag{20}$$

assigning x from C_l to C_m always decreases the total AIC.

Proof. Let $C_l^{'}$and C'_m be two clusters where $Y_l^{'} = Y_l \backslash \{x\}$, $Y_m^{'} = Y_m \cup \{x\}$ and $\beta_l^{'}$ and $\beta_m^{'}$ be the parameter vectors of clusters respectively, adjusted by M after the assignment of x from C_l to C_m. Assigning x from C_l to C_m is the best approach if

$$\sum_{y \in Y_l^{'}} AIC(\beta_l^{'}; y) + \sum_{y \in Y_m^{'}} AIC(\beta_m^{'}; y)$$

$$
\begin{aligned}
&< \sum_{y \in Y_l} AIC(\beta_l^{'}; y) + \sum_{y \in Y_m} AIC(\beta_m^{'}; y) \\
&= \sum_{y \in Y_l^{'}} AIC(\beta_l; y) + \sum_{y \in Y_m^{'}} AIC(\beta_m; y) \\
&\quad + AIC(\beta_l; x) - AIC(\beta_m; x)
\end{aligned}
$$

As M will update the model parameter vectors as long as the total AIC decreases, it will never increase total AIC of $C_l^{'}$ and $C_m^{'}$ after assignment of x from C_l to C_m. Based on the relations we obtained, if (20) is satisfied then

$$
\begin{aligned}
& AIC(\beta_l; X_t) - AIC(\beta_m; X_t) \\
&+ \sum_{y \in Y_l^{'}} AIC(\beta_l; y) + \sum_{y \in Y_m^{'}} AIC(\beta_m; y) \\
&> \sum_{y \in Y_l^{'}} AIC(\beta_l^{'}; y) + \sum_{y \in Y_m^{'}} AIC(\beta_m^{'}; y) \\
& \sum_{y \in Y_l} AIC(\beta_l; y) + \sum_{y \in Y_m} AIC(\beta_m; y) \\
&> \sum_{y \in Y_l^{'}} AIC(\beta_l^{'}; y) + \sum_{y \in Y_m^{'}} AIC(\beta_m^{'}; y)
\end{aligned}
$$

Thus, if 20 is satisfied, assigning x from C_l to C_m will always minimize the total AIC. □

As we find the best cluster for each time series, the models of the clusters need to be updated to acquire the minimum AIC values. So, we update the models for the altered clusters using Eq. 17. By initializing M with the previous models, we decrease the convergence time of our algorithm substantially. We continue the ”cluster reassignment” and ”model update” steps successively until the overall average AIC can no further be improved. Forming the model clusters is given in Algorithm 3.

4.1.3 Finding Optimal Number of Clusters

Finding the right number of clusters is a common problem in any clustering based approach. One can repeat the ICM algorithm with different number of clusters to find a tradeoff between scalability and accuracy. Instead of this, we opt in to a linear hill-climbing strategy to find the right number of clusters. Given the initial number of clusters to start with, we use a *bisecting* method to split a cluster into two, and continue this process until the splitting does not improve the models.

Let’s assume that the initial clusters are set using the *Initialize* algorithm. We estimate the average AIC of each of the clusters to find the cluster model that gives the highest average AIC for the corresponding time series. We aim to find a better model for the time series with high AIC value for the corresponding cluster model. We split the cluster into two according to the average

Algorithm 3: $Form - Clusters(\mathcal{Y}, \{C_i^{(0)}\}_{i=1}^{k}, \epsilon, M)$

```
$\triangle AIC^{(0)} \leftarrow \infty$
$e \leftarrow 0$
while $\triangle AIC^{(e)} > \epsilon$ do
    $l_i = \arg\min_j AIC(\tau_j^{(e)}; x_i)$ for $i = 1, 2, ..., N$
    $Y_{l_i}^{(e+1)} \leftarrow Y_{l_i}^{(e+1)} \cup x_i$ for $i = 1, 2, ..., N$
    $|\tau_i^{(e+1)}| \leftarrow update(Y_i^{(e+1)}, M, \tau_i^{(e)})$
    $AIC^{(e+1)} \leftarrow 1/N \sum_{j=1}^{k} \sum_{x \in Y_j^{(e+1)}} AIC(\tau_j^{(e+1)}; x)$
    $\triangle AIC^{(e+1)} \leftarrow (AIC^{(e)} - AIC^{(e+1)})/AIC^{(e)}$
    $e \leftarrow e + 1$
return $\{C_i^{(e-1)}\}_{i=1}^{k}$
```

AIC value of the cluster. We create a new cluster and put the time series with higher AIC value than the average AIC into the new cluster. We estimate an aggregated time series for the new cluster as we did in the initialization and fit a SARIMA model on this time series as the cluster representative. Then, we update the model parameters of these two clusters and reassign time series to clusters having minimum AIC value as in our form-clusters algorithm. We continue to split clusters until the improvement on the total AIC of the new clusters is within a small bound of the total AIC of the previous clusters. Algorithm 4 gives the details of this *bisecting* strategy.

Intuitively, we expect the new cluster model $\tau_{k'+1}^{(e+1)}$ to be a better fit for its time series $Y_{k'+1}$ than $\tau_{k'}^{(e)}$. Furthermore, we update the previous model as well which will give a better fit for the remaining time series $Y_{k'}$. Thus, the two new models will give better fits for time series $Y_l^{(e)}$ compared to other clusters. This is why we only update the models of these two clusters.

4.1.4 Updating Model Parameters with New Data

As new data points $\mathcal{Y}$ arrive, we update the models in each cluster using the

$\tau_i^{(e)} = update(\mathcal{Y}, M, \tau^{(e-1)})$ function that uses a minimization algorithm M initialized with $\tau^{(e-1)}$, and estimates next parameters $\tau^{(e)}$ minimizing the total AIC of $\mathcal{Y}$. We initialize the minimization algorithm M with previous parameters $\tau^{(e-1)}$ because this hastens the convergence as the clusters also stabilize in time.

4.1.5 Forecasting Future Data Points

Let's assume that given a time series x, h-step ahead forecasts are achieved by using the function f where $\hat{x}_{t+h} = f(x; \beta, h)$ using parameters β. Then our clustered forecasts are performed by using the desired forecasting function f with the model parameters of the corresponding cluster. More formally if the time series x is clustered in $C(F, P)$ with the corresponding parameter vector

Algorithm 4: $Bisecting(\mathcal{Y},\{C_i^{(0)}\}_{i=1}^{k}, \epsilon)$

```
$\triangle AIC^{(0)} \leftarrow \infty$
$e \leftarrow 0$
$k' \leftarrow k$
while $\triangle AIC^{(e)} > \epsilon$ do
    $AIC_j^{(e)} \leftarrow 1/\left\|Y_j\right\| \sum_{x \in Y_j^{(e)}} AIC(\tau_j^{(e)}; x)$
    $l \leftarrow \arg\min_j AIC_j^{(e)}$
    $Y_{k'+1} \leftarrow \{x | x \in Y_l^{(e)} \wedge AIC(\tau^{(e)}; x) > AIC_j^{(e)}\}$
    $Y_{k'} \leftarrow Y_{k'}^{(e)} - Y_{k'+1}$
    $|\tau_j^{(e+1)}| \leftarrow update(Y_j, M, \tau_j^{(e)})$ for $j \in \{k', k'+1\}$
    $l_i = \arg\min_j AIC(\tau_j^{(e+1)}; x_i)$ for $i = 1, ..., N$
    $Y_{l_i}^{(e+1)} \leftarrow Y_{l_i}^{(e+1)} \cup x$ for $i = 1, ..., N$
    $AIC^{(e+1)} \leftarrow 1/N \sum_{j=1}^{k'} \sum_{x \in Y_j^{e+1}} AIC(\tau_j^{(e+1)}; x)$
    $\triangle AIC^{(e+1)} \leftarrow (AIC^{(e)} - AIC^{(e+1)})/AIC^{(e)}$
    $e \leftarrow e + 1$
    $k' \leftarrow k' + 1$
return $\{C_i^{(e-1)}\}_{i=1}^{k'-1}$
```

τ then we give the h-step ahead forecasts of x using the following formula

$$\tilde{x}_{t+h} = f(x; \tau, h) \tag{21}$$

One can also use a single forecast as an estimator of the cluster. But, this will lack the effect of the individual time series values. As a result, using a single model on the time series individually is better than giving a single forecast for the cluster.

4.2 Sequential Clustered Modeling

The ICM approach clusters and models time series concurrently. We now present Sequential Clustered Modeling (SCM), where these two steps are applied sequentially. Although most time-series representations are not specifically designed for forecasting, we investigate how to utilize them. Our intuition here is same: applying a suitable common model is more efficient and can be more accurate than building separate models. Using a time series representation, we cluster data and assign the model built on each cluster center as the corresponding representative model. Forecasts for each individual time series are obtained using the model of the corresponding cluster representative.

More formally, we initially partition time series into k clusters $C_i(A_i, Y_i)$, $i = 1, ..., k$ using a representation and a distance measure where A_i is the cluster center, and Y_i is the time series in cluster C_i. Next we fit a SARMA model F_i to each of the cluster centers A_i. We construct our model clusters by transforming previous clusters to $C_i(F_i, Y_i)$ where F_i gives the SARMA model for Y_i by minimizing the corresponding AIC.

While any representation and clustering approach can be utilized, we adapt LPC (Linear Prediction Cepstrum) coefficients which is commonly used in speech and image processing [11]. Our experimental results confirm that LPC perform significantly better than the traditional representations of PCA, DWT and DFT for our purposes.

LPC is the cepstral representation of the Linear Prediction Coefficients. It was shown to be effective for model based time series clustering in terms of silhouette coefficient [9]. One can use the invertibility of a SARMA model and construct LPC coefficients using AR representation of every SARMA model. Linear Prediction Coefficients are the AR representation of the time series and can be specified by all-pole model in frequency domain [20]. Although a time series can be modeled by an AR model explicitly, an invertible SARIMA model can be converted to an equal infinite order AR model [24]. Thus based on our SARMA definition with the integration, AR representation of the corresponding SARIMA model can be found by solving

$$\phi^{'}(B) = \sum_{i=0}^{\infty} \phi^{'}_i x_{t-i} = \frac{\Phi_P B^S \phi(B)(1-B)^d}{\Theta_P B^S \theta(B)} x_t = w_t \tag{22}$$

Given an AR representation $\phi^{'}(B)$, LPC coefficients can be defined as follows [11]

$$c_i = \begin{cases} -\phi'_1 & \text{if } i = 1 \\ -\phi'_i - \sum_{j=1}^{i-1} (1 - \frac{j}{i})\phi'_j c_{i-j} & \text{if } 1 < i \leq p \\ -\sum_{j=1}^{i-1} (1 - \frac{j}{i})\phi'_j c_{i-j} & \text{if } p < i \end{cases} \tag{23}$$

Utilizing LPC for forecasting has several challenges, such as how to find a model for each LPC cluster and how to make next period forecasts. We set the cluster centers as the medians of the LPC coefficients of the respective time series. As the LPC coefficients are extracted using AR coefficients, the obvious approach would be to use the cluster center and obtain a common AR coefficient for that cluster. One can use this approach to model the clusters and obtain the next period forecasts. However, we experimentally observe that this approach has a high MAPE to be used in practice. Instead, we construct the median time series for each cluster and build a SARIMA model on it. Another decision is whether to generate only one next period forecast for all the time series in a cluster, or use the common model in the cluster for each time series separately and obtain time series specific forecasts. We use the later approach as it involves data itself and the forecasts use historical data for each time series.

We show various trade-offs and insights in using SCM, ICM, and IM approaches in the next section. For example, the execution time of LPC based SCM is comparable to IM, and is significantly slower than ICM. It requires models of each time series in advance, which is not that case in ICM.

5 Performance Evaluation

We demonstrate the efficiency, accuracy, and scalability of the proposed approach compared to the individual forecasts. We used three real data sets and six synthetic data sets in our experiments.

The first real data is a set of telco traffic time-series of 2497 enterprise customers each with 867 time points of their usage minutes. The time series shows highly seasonal behavior and is sensitive to the special events, e.g., holidays, unexpected events, campaigns. The other real data sets are publicly available data used in [9]. The first one is the Population data, which is a set of time series representing population estimates from 1900-1999 in 20 states of the US. The second data set is Personal Income, which is a set of time series representing the per capita personal income from 1929-1999 in 25 states of USA. We also generated six synthetic data sets based on these real data through aggregations, introducing random local outliers, and enhancing their sizes following the methodology presented in [9]. Using the available time series with periodicity 7, we fit SARIMA models to all of the time series. We uniformly selected AR, SAR, MA, and SMA coefficients from the intervals $[\phi_i - \sigma, \phi_i + \sigma]$, $[\Phi_i - \sigma, \Phi_i + \sigma]$, $[\theta_i - \sigma, \theta_i + \sigma]$ and $[\Theta_i - \sigma, \Theta_i + \sigma]$ respectively. In our experiments we used $\sigma = 0.05$ for the first real data and $\sigma = 0.01$ for the rest, which preserves the invertibility and causality of the generated SARMA model. We generated six data sets each having 100,000 time series. The CM approach intrinsically handles stationary issues by differencing and Box-Cox transformations. For modeling, we utilize the R Project that involves several statistical packages useful in our analyses [1]. To fit the best model minimizing 18 and 19 of a single time series, we use the R Package [13].

For evaluation, we took the first 839 time points for building the model and made forecasts for the later 28 points. To evaluate the dynamic update of ICM, we took the first 832 and dynamically add 7 points to each time series. We provide accuracy results for weekly (4 weeks) forecasts. We compare our accuracy results with the individual forecasts using Mean Absolute Percentage Error (MAPE),

$$MAPE = \frac{1}{h}\left(\Sigma_{i=1}^{h} | \frac{x_{n+i} - f_i}{x_{n+i}} | \right) \tag{24}$$

where h is the forecasting period, x_{n+i} is the i-th future time point, and f_i is the i-th forecast. In our experiments we address several questions, including:

- How does the accuracy results change compared to the individual forecasts?
- How does the speed of CM change compared to the speed of the individual fits?
- How does the accuracy and speed of CM change compared to other clustering algorithms?
- How does the number of clusters affect the speed and accuracy results?
- How does the accuracy and speed of the dynamic update change?
- Is the proposed approach scalable?

We give results to answer the questions above, then we compare ICM to SCM and IM. We finally experiment for the scalability of the algorithms.

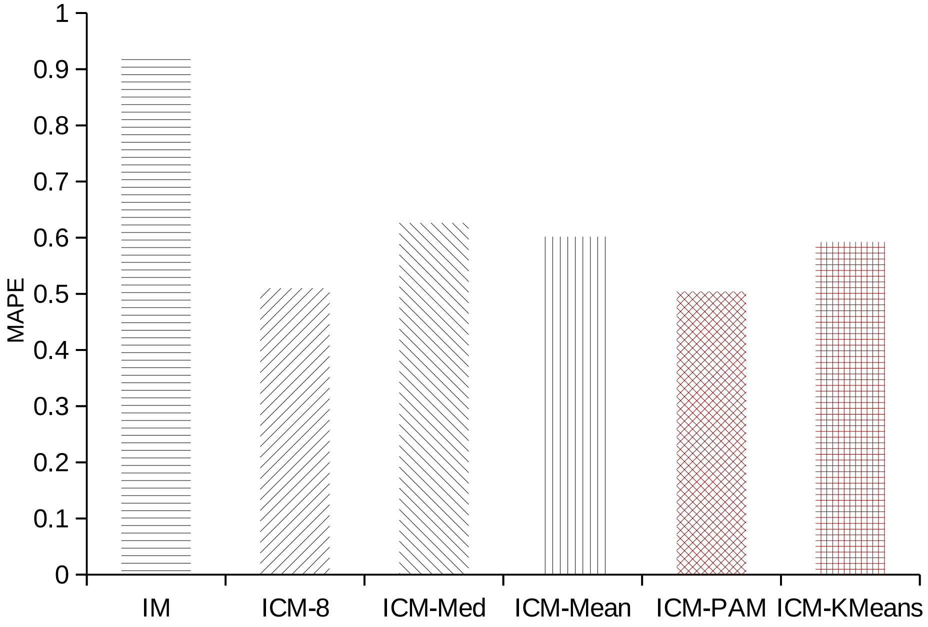

Figure 1: MAPE results of ICM and IM

5.1 Efficiency and Accuracy of Initialization

For each of the methods we used in initialization, we ran 10 experiments. We present the average results of these experiments. ICM-8 is the result of ICM with 8 clusters, using random initialization and median as the aggregate time series. The other methods use the presented bisecting method to find the optimal number of clusters. Considering our first question, Fig. 1 shows the average MAPE results over all time series. We take the average of weekly forecast errors. The error of ICM is 0.59, and for IM it is 0.93. On average ICM provides 37% improvement on weekly MAPE over IM. The difference between ICM-8 and the other ICM algorithms is small which shows that the bisecting strategy is effective in finding the number of clusters.

Fig. 2 shows the time requirement of ICM and IM. While IM takes hours to fit models, ICM is significantly faster by providing this in minutes. On average ICM takes around 9 minutes, while it takes 173 minutes to model each time series individually.

We observe that using a clustering algorithm in initialization (PAM, KMeans) has a significant time overhead for clustering and convergence of ICM. PAM and KMeans both use Euclidean distance and the outliers are assigned non-uniformly. The *Bisecting* algorithm has a small overhead with random initialization, 3.4 and 2.2 minutes respectively. A random initialization strategy with bisection has a high accuracy with a negligible overhead.

To evaluate the effect of randomness in different initialization strategies, we estimate the standard deviation (std) in MAPE and time. On average, the std in MAPE is 0.077, 0.046, 0.098, and 0.13, for median, mean, PAM, and KMeans, respectively. The error introduced by randomness in initialization has a relatively negligible effect in accuracy. On average, the std in time is 6, 2.4, 14, 11 minutes, respectively. While it is small for random initialization, it is relatively high for PAM

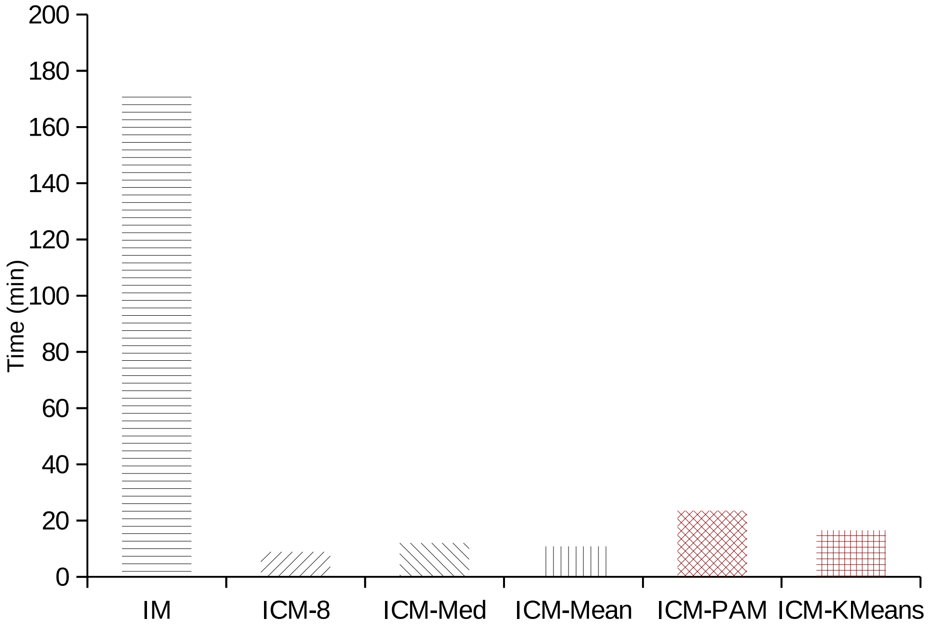

Figure 2: ICM and IM time

and KMeans. Overall, the randomness in initialization has a small overhead on the ICM algorithm in practice.

We evaluate the effect of seasonality by using an additive seasonal decomposition where a time series has 3 components, i.e., $x_t = seasonal_t + trend_t + random_t$. We ran our ICM model with random initialization and median aggregate on the $trend_t$ time series and add seasonal components to the resulting forecasts, i.e., $\tilde{x}_{t+h} = f(x; \tau, h) + seasonal_{t+h}$. The average MAPE and std is 0.41 and 0.0016, which shows that removing seasonality improves the accuracy and robustness of ICM. ICM is more valuable when there are local outliers in time-series. Removing the seasonality helps with these local events and noise and reduces the MAPE. The running time of the seasonal decomposition is 5 minutes for the first telco traffic data set. The total running time of ICM with seasonal decomposition is 9.3 minutes, which is 18.6 times faster than IM. ICM also converges faster with seasonal decomposition. On average it takes 4.3 minutes without seasonality while it takes 8.6 minutes with seasonality.

We also vary the number of clusters and do not observe a clear pattern for the relationship between the number of clusters and MAPE. There is a linear relationship between the number of clusters and the time that ICM requires.

5.2 Results on Time Series Clustering Approaches

We perform experiments for SCM using 5 different representations, LPC, DFT, DWT, PCA, and raw data. We use the resulting clusters and raw time series to forecast the future points. With median and mean of the time series belonging to each cluster, we end up having 10 different

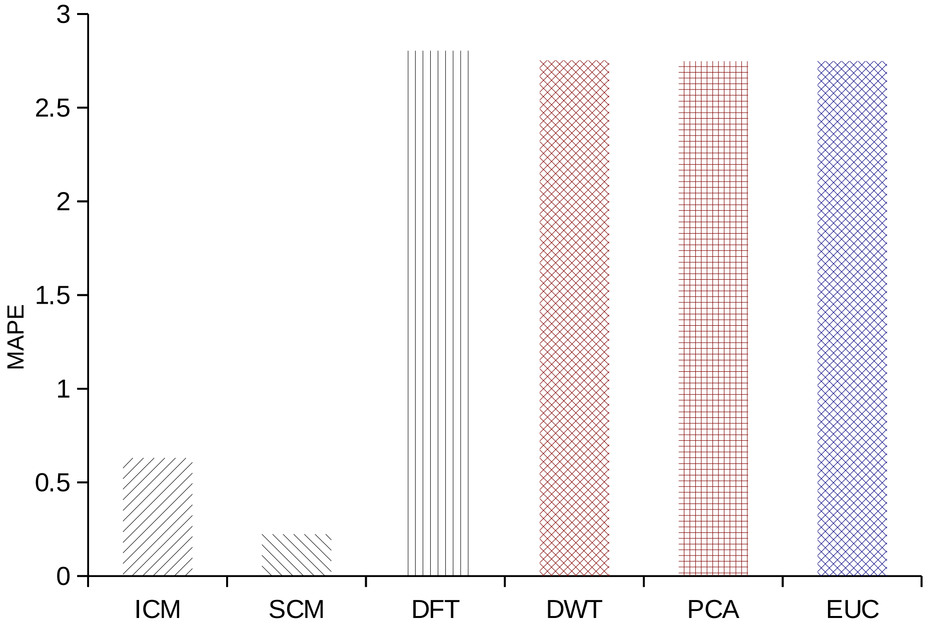

Figure 3: MAPE results of ICM vs SCM

approaches. We use the top 10 features extracted using each of the representations DFT, DWT, PCA, and LPC with PAM clustering for DWT, PCA, LPC and raw data and k-means clustering for DFT with Euclidean distance. Using 10 features was shown to be generally descriptive enough for these approaches [9].

Fig. 3 shows the comparison of modeling using representations with Euclidean distance. We present the results using the median as the aggregate representation, but we observe similar results for the mean as well. While DFT, DWT, PCA and raw data do not perform well on accuracy, LPC based SCM achieves a MAPE of 0.22. LPC is more accurate than ICM, however, it requires SARMA models to be available to construct the cepstral coefficients. Thus it is computationally much more expensive than ICM. DFT, DWT, PCA and raw data are efficient but not accurate. LPC is comparable to IM with around 190 minutes on average. The execution time of each algorithm is presented in Fig. 4.

These results suggest that LPC is suitable for small-scale as it significantly improves the accuracy. For large-scale data, ICM is preferable over both IM and LPC, as both have efficiency and scalability problems. As new time points come, the cepstral coefficients need to be updated as well as the common models for each cluster. The other representations are reasonably faster but they lack of the necessary accuracy to be used in real life applications. Considering both speed and accuracy, ICM is both fast and accurate and hence more practical.

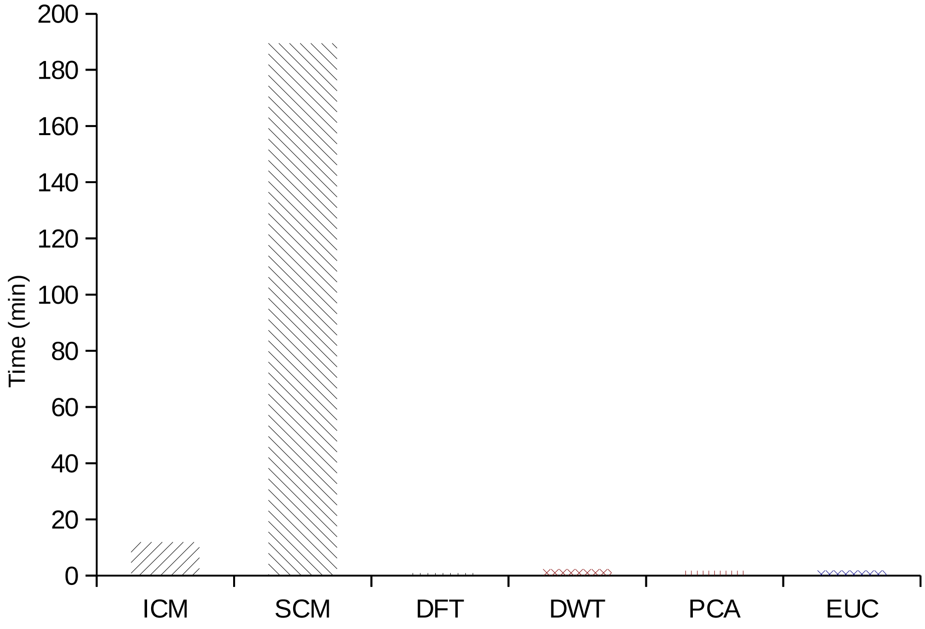

Figure 4: Comparison of the running time of ICM with SCM

5.3 Dynamic Update of Model Parameters

As our optimization algorithms are iterative in nature, we can update the models as new time points arrive. We first merge the existing time series with newly arriving points. Then we use the model parameters estimated in the initialization phase as the initial inputs to the optimization algorithm and run using the new data. To compare, we re-run our ICM algorithm from scratch and show that ICM-u (Clustered Modeling Update) takes less time than the re-run, thus transitively it takes considerably less time than individually fitting data as new points come. Fig. 5 shows the comparison of MAPE results of ICM-u as new points arrive with re-run of ICM and IM. ICM-u and re-run are comparable with each other, and both are more accurate than the IM. ICM-u takes 1.77 minutes which is faster than both re-run and IM. The reason is that clusters stabilize at the end of ICM, and ICM-u converges fast as we initialize it with the resulting clusters of ICM. This is summarized in Fig. 6.

To show how the accuracy and speed changes if data size increases, we choose 500 time series randomly and at each iteration we add 500 more distinct time series. Fig. 7 shows that as the size of data increases, the accuracy of the same subset remains nearly the same. This result shows that as the data increases, the clusters may change but the accuracy is preserved. Fig. 8 exhibits a linear relationship between the size of the data and the time it takes. As the data size doubles, ICM takes approximately double time but with a very small slope compared to IM.

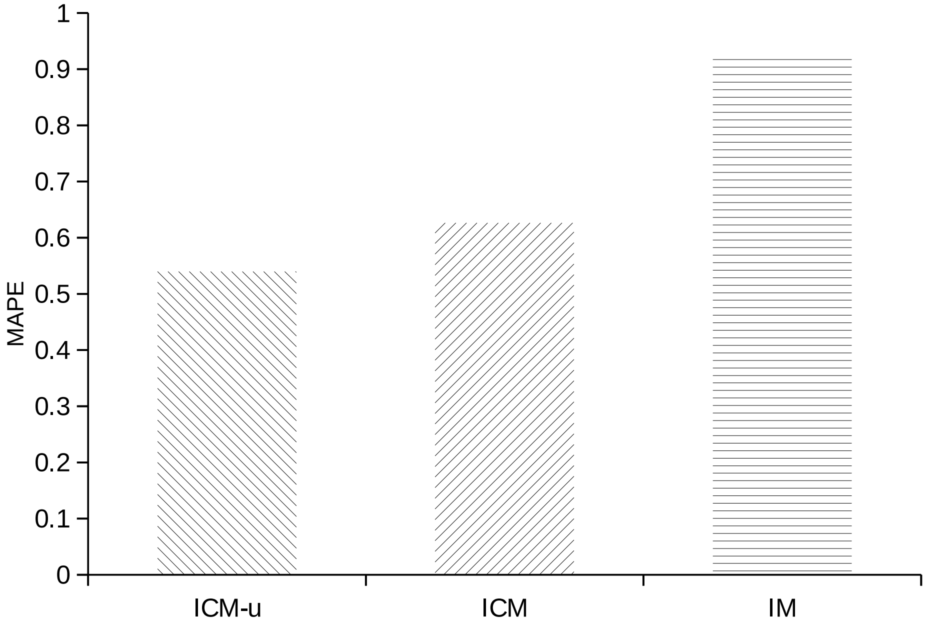

Figure 5: MAPE comparison of ICM-u with re- run and IM

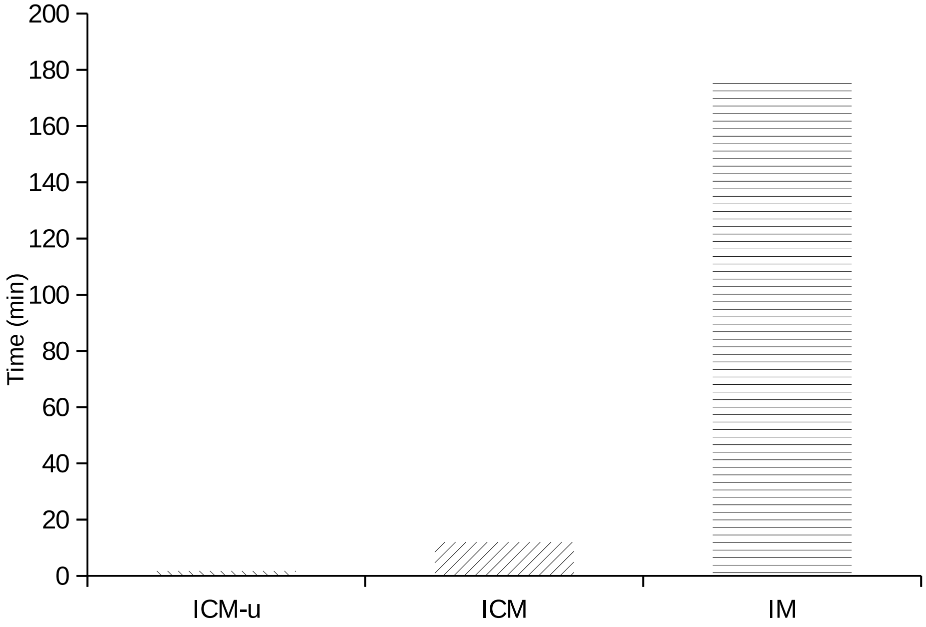

Figure 6: The time of ICM-u algorithm and re-run vs. IM

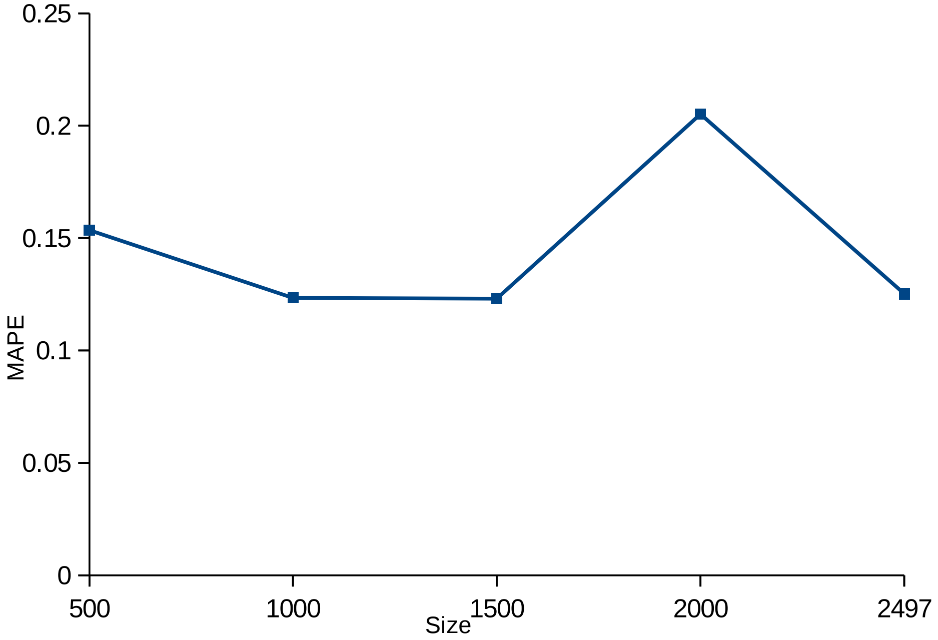

Figure 7: Accuracy as the data size increases

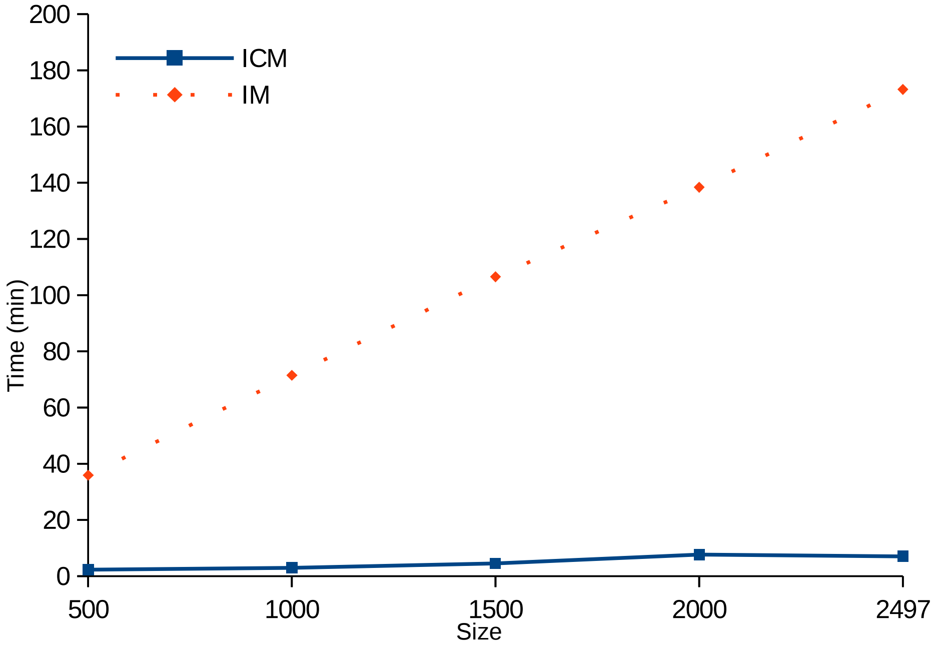

Figure 8: The time of ICM vs. IM as the data size increases

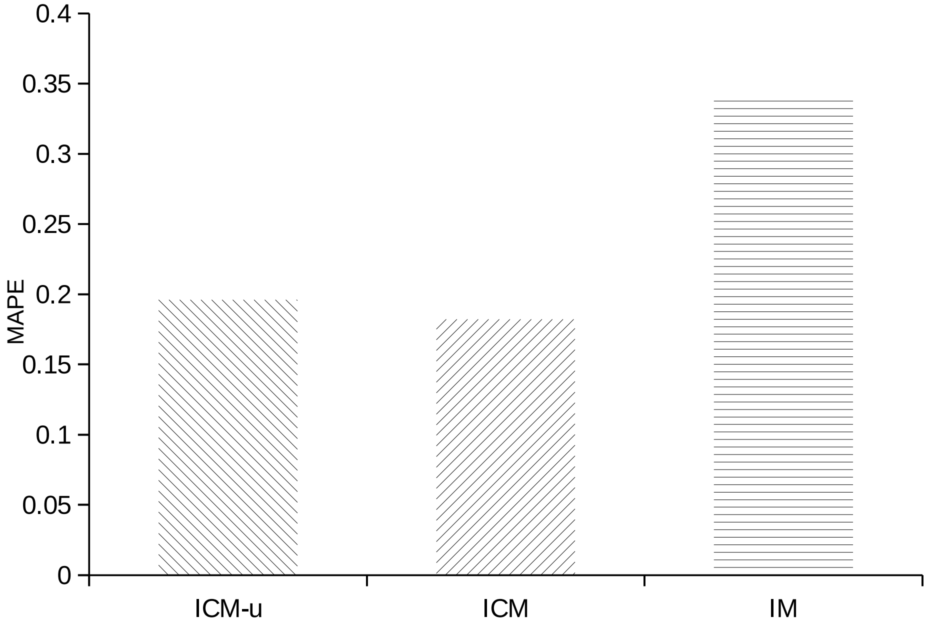

Figure 9: MAPE results of ICM-u, re-run and IM for large data set

5.4 Experiments for the Larger Dataset

We first compare ICM-u modeling with ICM and IM. We then compare the accuracy and running time performance of ICM and IM as the data size increases on the synthetic data sets.

Fig. 9 illustrates the accuracy comparison of the proposed approach with IM. We vary the data set size from 2500 to 100,000 and present the average results. ICM and ICM-u have lower MAPE than IM, and ICM-u competes with ICM. This suggests that, instead of building ICM from start, we can use ICM-u to obtain a similar accuracy.

Fig. 10 shows the time that ICM, ICM-u, and IM take. It takes 43 minutes and 6.5 hours for ICM-u and ICM respectively, while it takes 126 hours for IM. On average, ICM is 19 times faster than IM. If we have available clusters, we can further improve the results using ICM-u which is around 170 times faster than IM.

5.5 Evaluations for Scalability

We run ICM over six synthetic data sets to evaluate the scalability of ICM. Fig. 11 shows the time requirements for ICM and IM over the six data sets. On all data sets, ICM is more scalable than IM. On average, ICM takes 3.5, 10, 12.5, 7.5, 2.6, and 2.1 hours to build models while it takes 126, 146, 163, 153, 13, and 12 hours for IM, respectively.

When the data have local outliers, the improvement by ICM becomes more apparent, as it has an aggregate effect to remove outliers. If the data does not have any outliers, than ICM is comparable with IM. The MAPE results of IM in our synthetic data sets are 0.34, 0.15, 0.13, 0.13, 0.009, and 0.007 respectively. The first data set has a relatively higher error and ICM manages to improve

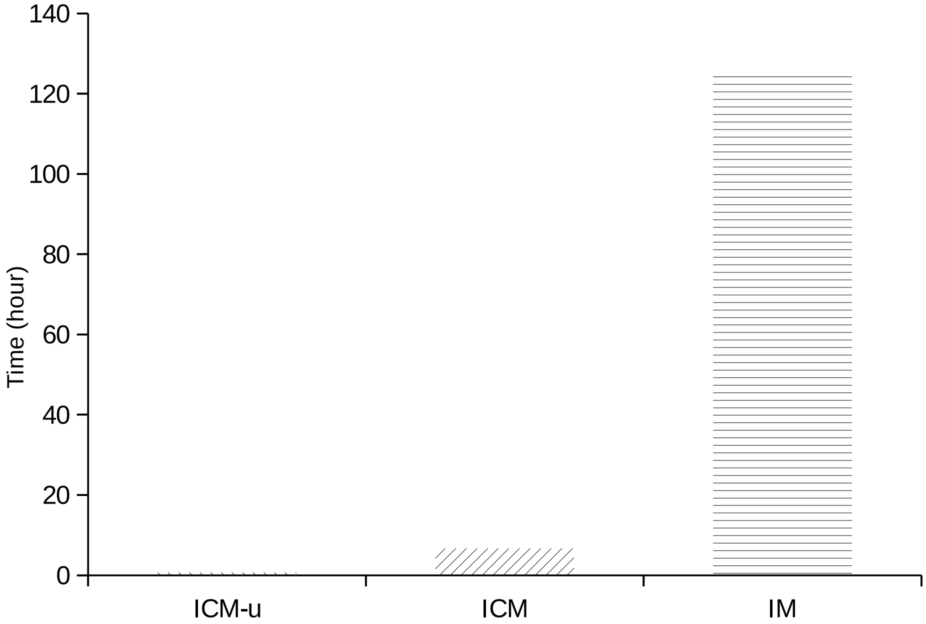

Figure 10: Time results of ICM-u, re-run, and IM for large data set

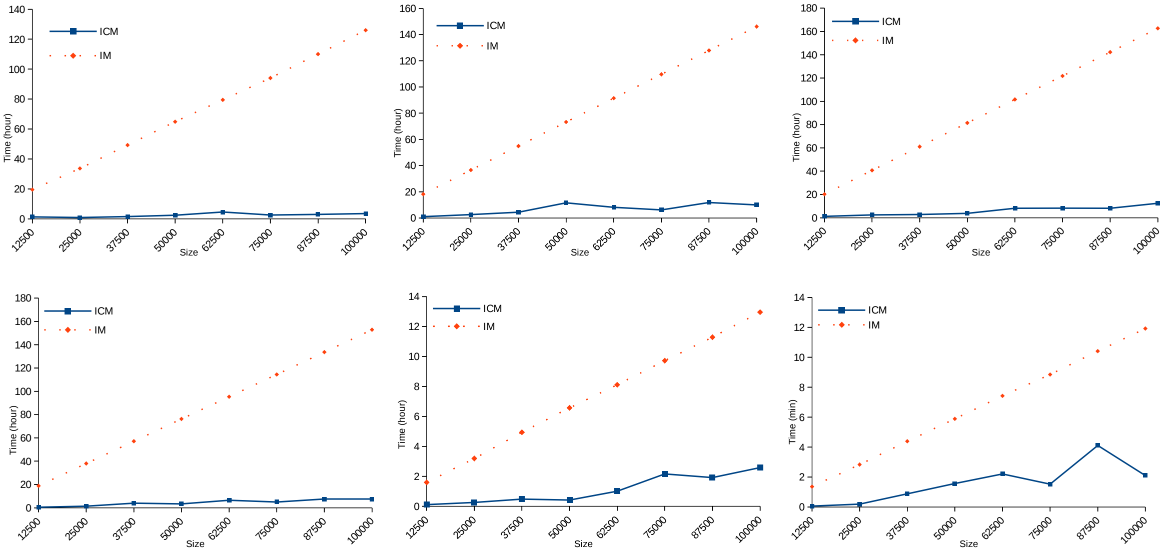

Figure 11: Time of ICM and IM on large data set

the accuracy by 50%. In others, the differences in accuracies are within a 1% margin showing that if the individual models already have a high accuracy, ICM cannot improve the models.

6 Conclusions

We addressed the problem of continuous forecasting of multiple time series for scalable predictive analytics. We proposed two approaches: one with clustering and modeling of data performed simultaneously (ICM), and another where data is first clustered then modeled (SCM).

The ICM approach clusters the time series according to their AIC values. A time series belongs to the cluster which gives the lowest AIC value estimated using the SARIMA model of the cluster and the time series itself. We improve the cluster models using iterative nonlinear optimization algorithms, which enables efficient dynamic model updates as new time points arrive. ICM is not restricted to the SARIMA models and can be applied to any modeling with a given minimization procedure. It is more scalable with comparable accuracies to individual modeling (IM). Each IM with a SARIMA model takes several seconds to minutes, hence, it takes significant time to continuously update even a couple of thousand series. For example, on the usage traffic data of 2497 telecommunication customers, IM takes around 173 minutes on a standard PC. ICM takes around 9 minutes for the same data set.

The SCM approach applies clustering and modeling sequentially. We use the invertibility of a SARMA model and construct LPC coefficients using AR representation of every SARMA model. We use the results of k-Means and PAM clustering structures and corresponding centers, and build a SARMA model for each cluster center. Forecasts for each individual time series are achieved using the model on the center of their corresponding cluster. We showed that LPC based SCM provides more accurate results than IM with some overhead.

We compare ICM and SCM to IM, and to each other. Experimental results show that ICM is up to 20 times faster and 37% more accurate than IM, and SCM has 76% improvement over IM with a speed overhead of 9% on the real telco traffic series. The proposed methodology is independent of the underlying linear or nonlinear modeling approach, and can benefit from any model selection method.

References

[1] http://www.r-project.org/

[2] Charu C. Aggarwal, Jiawei Han, Jianyong Wang, and Philip S. Yu. A framework for clustering evolving data streams. *VLDB proceedings*, 29:81–92, 2003.

[3] A.M. Alonso, J.R. Berrendero, A. Hernandez, and A. Justel. Time series clustering based on forecast densities. *Comput Statist Data Anal*, 51:762–776, 2006.

[4] George E. P. Box, Gwilym M. Jenkins, and Gregory C. Reinsel. *Time Series Analysis: Forecasting & Control.* 3rd edition, 1994.

[5] Richard H. Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16(5):1190–1208, September 1995.

[6] Deepayan Chakrabarti and Christos Faloutsos. Large-scale automated forecasting using fractals, 2002.

[7] Kin-Pong Chan and Ada Wai chee Fu. Efficient time series matching by wavelets. In *In ICDE*, pages 126–133, 1999.

[8] Marcella Corduas and Domenico Piccolo. Time series clustering and classification by the autoregressive metric. *Computational Statistics & Data Analysis*, 52(4):1860–1872, January 2008.

[9] Konstantinos Kalpakis Dhiral, Konstantinos Kalpakis, Dhiral Gada, and Vasundhara Puttagunta. Distance measures for effective clustering of arima time-series. In *In proceedings of the 2001 IEEE International Conference on Data Mining*, pages 273–280, 2001.

[10] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. *SIGMOD Rec.*, 23(2):419–429, May 1994.

[11] S. Furui. *Digital Speech Processing, Synthesis, and Recognition.* New York and Basel: Marcel Dekker, Inc., 1989.

[12] Liangjie Hong, Dawei Yin, Jian Guo, and Brian D. Davison. Tracking trends: incorporating term volume into temporal topic models. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '11, pages 484–492, New York, NY, USA, 2011. ACM.

[13] Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, 27(3):1–22, 7 2008.

[14] Flip Korn, H. V. Jagadish, and Christos Faloutsos. Efficiently supporting ad hoc queries in large datasets of time sequences. *SIGMOD Rec.*, 26(2):289–300, June 1997.

[15] Mahesh Kumar and Nitin R. Patel. Using clustering to improve sales forecasts in retail merchandising. *Annals OR*, 174(1):33–46, 2010.

[16] Irina Kevecka. Forecasting traffic loads: Neural networks vs. linear models. *Computer Modelling and New Technologies*, 14(2):20–28, 2010.

[17] Lei Li and B. Aditya Prakash. Time series clustering: Complex is simpler! In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pages 185–192, New York, NY, USA, June 2011. ACM.

[18] Lei Li, B. Aditya Prakash, and Christos Faloutsos. Parsimonious linear fingerprinting for time series. *Proc. VLDB Endow.*, 3(1-2):385–396, September 2010.

[19] Jessica Lin, Michail Vlachos, Eamonn Keogh, and Dimitrios Gunopulos. Iterative incremental clustering of time series. In Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Bhm, and Elena Ferrari, editors, *Advances in Database Technology - EDBT 2004*, volume 2992 of *Lecture Notes in Computer Science*, pages 106–122. Springer Berlin Heidelberg, 2004.

[20] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, April 1975.

[21] Yasuko Matsubara, Yasushi Sakurai, Christos Faloutsos, Tomoharu Iwata, and Masatoshi Yoshikawa. Fast mining and forecasting of complex time-stamped events. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '12, pages 271–279, New York, NY, USA, 2012. ACM.

[22] Pedro Pereira Rodrigues, João Gama, and Joao Pedroso. Hierarchical clustering of time-series data streams. *IEEE Trans. on Knowl. and Data Eng.*, 20(5):615–627, May 2008.

[23] Makridakis S., Wheelwright S. C., and R. J. Hyndman. *Forecasting: methods and applications*. John Wiley and Sons, 3rd edition, 1998.

[24] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples (Springer Texts in Statistics)*. Springer, 2nd edition, May 2006.

[25] Maciej Szmit and Anna Szmit. Usage of pseudo-estimator lad and sarima models for network traffic prediction: Case studies. In Andrzej Kwiecie, Piotr Gaj, and Piotr Stera, editors, *Computer Networks*, volume 291 of *Communications in Computer and Information Science*, pages 229–236. Springer Berlin Heidelberg, 2012.

[26] T. Warren Liao. Clustering of time series data-a survey. *Pattern Recogn.*, 38(11):1857–1874, November 2005.