SCALING FORECASTING ALGORITHMS USING CLUSTERED MODELING

A THESIS

SUBMITTED TO THE DEPARTMENT OF COMPUTER ENGINEERING

AND THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Mehmet Güvercin

August, 2013

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Hakan Ferhatosmanoğlu (Advisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Assoc. Prof. Dr. Uğur Güdükbay

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Nilgün Ferhatosmanoğlu

Approved for the Graduate School of Engineering and Science:

Prof. Dr. Levent Onural
Director of the Graduate School

ABSTRACT

SCALING FORECASTING ALGORITHMS USING CLUSTERED MODELING

Mehmet Güvercin
M.S. in Computer Engineering
Supervisor: Assoc. Prof. Dr. Hakan Ferhatosmanoğlu
August, 2013

Research on statistical forecasting has traditionally focused on building more accurate models for a given time-series. The models are mostly applied only to limited data due to their limitation on efficiency and scalability. However, many enterprise applications such as Customer Relationship Model (CRM) and Customer Experience Management (CEM) require scalable forecasting on large number of data series. For example, telecommunication companies need to forecast each of their customers' traffic load individually to understand their needs and behavior, and to tailor targeted campaigns. Forecasting models are easily applied on aggregate traffic data to estimate the total traffic volume for revenue estimation and resource planning. However, they cannot be 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. We address the problem of building and updating forecasting models continuously for multiple data series and propose dynamic clustered modeling optimized for forecasting. We introduce representative models as an analogy to cluster centers, and apply the models to each individual series through iterative nonlinear optimization. The approach performs modeling and clustering simultaneously, makes forecasts by applying representative models to each data, and updates the model parameters for a continuous forecasting process. Our findings indicate that understanding an individual's behavior within its segment's model provides more scalability and accuracy than computing the individual model itself. Experimental results from a real telecom CRM application show the method is highly efficient and scalable, and also more accurate than having separate individual models.

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

ÖZET

KÜMELEME MODELLEMESİ TABANLI ÖLÇEKLENEBİLİR TAHMİNLEME ALGORİTMALARI

Mehmet Güvercin
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Yöneticisi: Assoc. Prof. Dr. Hakan Ferhatosmanoğlu
Ağustos, 2013

İstatistiksel tahminleme konusunda yapılan araştırmalar geleneksel olarak daha çok verilen zaman serisi için daha doğru modeller oluşturmaya odaklanmaktadır. Bu modeller verimlilik ve ölçeklenebilirlik kısıtlarından dolayı sadece sınırlı sayıda zaman serisine uygulanabilmektedir. Ancak, Müşteri İlişkileri Yönetimi (MİY) ve Müşteri Deneyimi Yönetimi (MDY) gibi bazı kurumsal uygulamalar büyük veri seti üzerinde çalışabilen ölçeklenebilir tahminleme gerektirmektedir. Örnek olarak, telekomünikasyon firmaları müşterilerin ihtiyaçlarını ve davranışlarını anlamak veya müşteriye özel kampanya üretmek için her bir müşterinin ayrı ayrı trafik yükünün tahminine ihtiyaç duyarlar. Tahminleme modelleri, gelir tahmini veya kaynak planlaması için gerekli olan toplam trafik hacmi tahmininde toplu trafik verisi üzerinde kolayca kullanılabilir. Bununla birlikte, çok sayıda kullanıcı için ayrı ayrı model oluşturmak çok fazla zaman aldığı için bu durumda tahminleme modelleri uygulabilirliğini yitirmektedir. Tahminleme sürecinin sürekli olduğu ve modellerin periyodik olarak güncellenmesi gerektiği durumlarda problem daha da içinden çıkılmaz hale gelmektedir.

Biz bu çalışmada, birden fazla zaman serisi için tahminleme modellerini oluşturma ve sürekli bir şekilde güncelleme problemini ele almaktayız ve tahminleme için optimize edilmiş dinamik kümeleme modellemesini sunmaktayız. Çalışmada küme merkezleri için temsili model oluşturmayı ve bu modelleri tekrarlı doğrusal olmayan optimizasyon kullanarak her bir zaman serisine ayrı ayrı uygulamayı önermekteyiz. Öne sürülen yaklaşım, modelleme ve kümeleme işlemlerini eş zamanlı olarak yerine getirmekte, temsili modelleri her bir zaman serisine uygulayarak tahminleme yapmakta ve sürekli tahminleme süreci için model parametrelerini güncellemektedir. Elde ettiğimiz bulgular bireysel model davranışlarını kendi segment modeli üzerinden değerlendirmenin bireysel modeller hesaplamaya göre daha ölçeklenebilir ve daha doğru olduğunu göstermektedir.

Gerçek bir telekom MİY uygulaması üzerinde yapılan deneyler, önerilen yöntemin her bir kişiyi ayrı ayrı modellemeye göre yüksek verimli, ölçeklenebilir ve daha doğru olduğunu ortaya koymaktadır.

Anahtar sözcükler: Ölçeklenebilir tahminleme, zaman serisi modelleri, dinamik sürdürme, gruplandırma modellemesi, akan veri, performans, doğruluk.

Acknowledgement

I first would like to express my thanksgiving to my supervisor Assoc. Prof. Dr. Hakan Ferhatosmanoğlu from the bottom of my heart for his excellent guidances, enthusiasm and valuable suggestions.

I would like to thank to Assoc. Prof. Dr. Uğur Güdükbay and Asst. Prof. Dr. Nilgün Ferhatosmanoğlu for reading this thesis.

I am grateful to my family members for their continuous support in every period of my life.

I would like to thank İzzeddin Gür for his collaboration during all stages of this research.

I would like to thank to TUBITAK for financial support during my graduate studies.

Contents

List of Figures

List of Tables

Chapter 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 total revenue or network traffic using an aggregate time-series model on that attribute. Such a collective analysis provides insights on common patterns but not on understanding the customers and their needs. CRM and CEM applications are based on a customer centric view that requires scalable and dynamic models on multiple evolving data series. In terms of accuracy, a separate individual model for each series would be expected to perform well as each model is tailored for its corresponding data. However, this approach is not scalable since common forecasting models, such as Seasonal Auto Regressive Integrated Moving Average (SARIMA) [1], take nontrivial time even for a single time-series. A scalable approach is needed to scale forecasting models for 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 CRM example that originally motivated this research, among many Business Intelligence (BI) applications. Telecommunication companies predict their future network traffic load, such as total 3G connections or call volumes generated by their customers, for resource planning and revenue estimation. The forecasts are typically performed using well-formed statistical models such as Holtz Winter [2], exponential smoothing [1, 2], and SARIMA [2–4]. The models are practically applied to an aggregate time-series of total traffic on the company network. While aggregating the data makes the analysis more feasible, CRM requires understanding each customer's usage traffic patterns individually. For example, if the companies can forecast a customer's future traffic, they can design personalized campaigns to improve both the customer's experience and the company revenue.

Given the complexity of statistical forecasting models, individual modeling (IM) and their dynamic maintenance 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 forecasting model. Fortunately, data series in most real applications, such as customers call traffic, are not independent and show correlations through segments and due to regular and irregular events. We revisit statistical forecasting in the context of dynamic large-scale analytics and develop a less costly modeling approach that considers correlations and preserves accuracy. The ideal method would be accurate in forecasting future data for each series, efficient in building models and capturing correlations for a large number of time series, and scalable in accuracy and speed.

Our approach scales the forecasting algorithms by a continuous clustered modeling (CM) optimized for forecasting. We form groups of data simultaneously with appropriate forecasting models based on similarity in the context of forecasting. We use a forecasting model as an analogy to cluster center and apply the representative model to each series separately. A common model in a cluster eliminates the need for modeling every individual and results in efficiency as applying the model is significantly cheaper than building the model. The individual focus is also captured since individual data is used in model application that leads to more accurate fit as shown in the experimental results. Following this methodology, we develop two specific algorithms: the first one integrates clustering and modeling simultaneously, and the second where they are applied sequentially. We focus on improving SARIMA family of models for better integration to current enterprise

CRM and BI applications. In fact, they are also shown to perform better in forecasting aggregate traffic than more complicated and time consuming models such as neural networks [5,6]. We note that the proposed methodology is independent of the underlying linear or nonlinear modeling approach, and can benefit from any model selection method.

Our first method builds SARIMA based clusters on multiple series and continuously updates them through iterative non-linear optimization. For a single time series, the best model is obtained by minimizing the (Akaike Information Criterion) AIC 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 total AIC in groups instead of individually. We seek through the space of SARIMA models to group correlated data series into their segments according to their evolution pattern and find the ones minimizing total AIC starting with initial SARIMA models. The search and update are done through an iterative nonlinear optimization. As we search to minimize AIC, we also adjust clusters to decrease AIC further. Clusters are dynamically adjusted with model parameter updates to continuously decrease AIC.

Our second approach utilizes time-series clustering within the proposed representative forecasting models. We cluster time series using an Linear Prediction Cepstrum (LPC) based representation and build models on each cluster representative. Each series is applied its corresponding model and forecasts are made by applying the representative model. As data evolve, updates are applied only on the parameters of the representative models, not the whole data set.

The proposed grouped models avoid over-fits and capture common motifs even on noisy data. While the current forecasting models need pre-processing to smooth local outliers, the proposed approach is robust to bursty data with outliers and handles them automatically. We use one single model for all time-series in the cluster, however, the produced forecast for each time-series is different and tailored for the corresponding time-series. An appealing outcome of our work is to achieve the two seemingly contradictory goals at the same time: more accurate and more efficient forecasts compared to modeling each time series individually.

We discuss the related work in Chapter 2 and present the background in Chapter 3. In Chapter 4, we explain the proposed methodology including the specific approaches following the two methods. We first model multiple data series using non-linear optimization on groups of data. We then use time series representations within our new cluster definition and assign models to each individual time series by fitting models for each corresponding cluster. We present the experimental study and results in Chapter 5. Finally, we conclude in Chapter 6.

Chapter 2

Related Work

Time series clustering methods can be broadly categorized into three groups in terms of data they use: raw data, features extracted from time series, and models on raw time series [7].

Li and Prakash propose a time series clustering method to identify the category of the motion from given motion sequence [8]. They note that feature based clustering does not give appealing results for motion category identification since they 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 metric as a dissimilarity measure for time series classification and clustering [9]. They define AR distance from ARIMA processes, and derive asymptotic distribution of the squared AR distance to compute time series dissimilarity. They apply AR metric on two real problems and conclude that AR metric is well defined for seasonal and non-seasonal, long and short, stationary and non-stationary time series.

In management science community, Kumar and Patel use clustering for predictive analytics in retail merchandising [10]. The method depends on the trade-off between decreased variance and increased bias instead of just considering only decrease in variance as in Fishers method. It finds the number of clusters using the trade-off between decreased variance and increased bias. To calculate similarity of time series, they use next period forecasts and its variance instead of using historical data.

Kalpakis et al. study clustering of time series modelled with ARIMA models [11]. They use LPC coefficients as 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 demonstrate results better than existing time series representations (DFT, DWT, etc.) in terms of similarity metric and Silhouette coefficient.

Alonso et al. propose a clustering approach that considers evolution time series [12]. For dissimilarity calculation they use the full forecast densities instead of point forecasts and used squared Euclidean distance between full forecast densities. Authors also derived an approximation for the L_2 distance between forecast densities. Rodrigues proposes Online Divisive-Agglomerative Clustering (ODAC) for whole time series clustering [13]. Clusters are on the leaves of a binary tree and updated incrementally. Each leaf can be split or aggregated after testing confidence level which is given by the Hoeffding bound. The computation of 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 data stream clustering problem is presented in [14]. The stream clustering is divided into two sub-processes. In the first sub-process, which is called online process, summary statistics of data streams are stored periodically. In the second one, 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 online process.

An anytime iterative incremental clustering version of partitional clustering algorithms is introduced in [15]. They 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.

Recently Matsubara et al. [16] 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. Li et al. [17] propose to 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. [18] 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 [19]. 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. Also they divide the data into training and holdout set 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.

Kim et al. introduce a forecasting method that is able to capture special characteristics of Epilepsy EEG data in [20]. Epilepsy EEG data has such nonlinearity, nonnormality and nonperiodicity special characteristics that deteriorates performance of traditional forecasting methods. They propose to use coercively adjusted auto regression(CA-AR) to model this type data. They forcefully adjusted AR coefficients to deal with special characteristics of Epilepsy EEG data and they used a random coefficient between -1 and 1. Experimental results of authors show that CA-AR performs well for nonperiodic data. It is also faster and more accurate compared to the other existing forecasting methods.

The current approaches have mostly 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 dynamic clustered modeling 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 individual modeling approaches, such as the recent ones proposed by databases and data mining community [16–18].

Chapter 3

Background

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_x dimension

$$x = (x_1, x_1, ..., x_{n_x}) \tag{3.1}$$

where $x_i \in \mathbf{R^k}$, and n_x is called the length of the time series x. If $k > 1$ then it is called multivariate time series, in the other case it is called 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 to 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; transformation based models: PCA [21], SVD [22], spectral domain models: DFT [22], DWT [4], time domain models: ARMA [1, 3], ARIMA [1, 3], SARIMA [1, 3], GARCH [3], and state-space models: ARMAX [1,3]. The use of these models may vary from problem to problem, but we focus on 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 in the following parts.

SARIMA is one of the most widely used time domain model that have desirable theoretical and asymptotic behaviors. SARIMA family of models exploit 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 past time points and white noise is formed by both periodic and non-periodic components. Following the notation in [1], we give the definitions of operators and obtain a more compact representation for SARIMA.

Definition ***(Operators):*** The operators

$$\phi(B) \quad = 1 - \phi_1 B - ... - \phi_p B^p, \tag{3.2}$$

$$\theta(B) \quad = 1 + \theta_1 B + ... + \theta_q B^q, \tag{3.3}$$

$$\Phi_P(B^s) = 1 - \Phi_1 B^s - ... - \Phi_P B^{Ps}, and \tag{3.4}$$

$$\Theta_Q(B^s) = 1 + \Theta_1 B^s + ... + \Theta_Q B^{Qs} \tag{3.5}$$

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

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 ***(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,

$$cov(x_s, x_t) = \gamma(s,t) = \gamma(|(s-t)|). \tag{3.6}$$

In case of 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,

$$y_t = \begin{Bmatrix} (x_t^\alpha - 1)/\alpha \\ log x_t \end{Bmatrix}, \tag{3.7}$$

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 non-stationarity

$$y_t = \nabla^d x_t = (1 - B)^d x_t, \tag{3.8}$$

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 generic SARMA model using operators:

Definition *(SARMA):* A SARMA model is defined as

$$\Phi_P(B^s)\phi(B)x_t = \Theta(B^s)\theta(B)w_t, \tag{3.9}$$

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 calculating 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 [1]. We start with 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 ***(Log-likelihood):*** The log-likelihood of a $SARMA(p,q)x(P,Q)_s$ model built on x is defined as

$$\ell(\beta;x) = -\frac{n}{2}ln(2\pi\sigma_w^2) - \frac{1}{2\sigma_w^2}\sum_{t=1}^{n} w_t^2, \tag{3.10}$$

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 ***(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

$$SS(\beta;x) = \sum_{t=-\infty}^{n} w_t^2(\beta), \tag{3.11}$$

where β is the parameter vector of the model, $w_t = E(w_t|x_1, x_2, ..., x_n)$.
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 ***(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

$$CSS(\beta;x) = \sum_{t=p+1}^{n} \hat{w}_t^2(\beta), \tag{3.12}$$

where β is the parameter vector of the model, $\hat{w}_t = E(w_t|x_1, x_2, ..., x_p)$.
Model selection. Although parameter estimation methods seem to be enough for modeling, it is known that increasing the number of parameters give 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 [1]. Our algorithms are not specific to any model selection, but we will focus on AIC.

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

$$AIC(\beta;x) = -2\ell(\beta;x) + 2k \tag{3.13}$$

or

$$AIC(\beta;x) = n(1 + log(2\pi)) + nlog(CSS(\beta;x)) + 2k, \tag{3.14}$$

where k 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 parameter β. Given a time series x, the best model parameters are found by

$$\beta^{'} = \arg\min_{\beta} AIC(\beta;x). \tag{3.15}$$

List of symbols and list of abbreviations used throughout the this thesis is presented in Table 3.1 and Table 3.2 respectively.

Table 3.1: Table of symbols

SYMBOL	DESCRIPTION
x	A time series
n_x	Length of time series
ϕ	Auto regressive operator
θ	Moving average operator
Φ	Seasonal auto regressive operator
Θ	Seasonal moving average operator
s	Seasonal period
α	Power of the transformation
d	Order of differencing
w_t	Gaussian white noises
p	Auto regressive order
q	Moving average order
P	Seasonal auto regressive order
Q	Seasonal moving average order
β	Parameter vector of the model
σ_w^2	Variance of the w_t
$\hat{w}$	Expected value of w_t
k	Total number of parameters of a SARIMA model
β'	Best model parameters
n	Number of time series
k	Number of clusters
P	A subset of time series
τ	Parameter vector of a subset of time series
$C(F, X)$	A model cluster
F	Common forecasting model
$tseries$	s multiple time series
M	A minimization algorithm
o	Orders vector
$\{C_i^{(0)}\}_{i=1}^{k}$	The set of initial clusters
ϵ	Stopping margin for average AIC
h	Forecasting period
c	LPC coefficient

Table 3.2: Table of abbreviations

SYMBOL	DESCRIPTION
CRM	Customer Relationship Model
CEM	Customer Experience Management
MY	Müşteri İlişkileri Yönetimi
MDY	Müşteri Deneyimi Yönetimi
SARIMA	Seasonal Auto Regressive Integrated Moving Average
BI	Business Intelligence
IM	Individual Modeling
CM	Clustered Modeling
AIC	Akaike Information Criterion
LPC	Linear Prediction Cepstrum
AR	Auto Regression
ARIMA	Auto Regressive Integrated Moving Average
DFT	Discrete Fourier Transform
DWT	Discrete Wavelet Transform
ODAC	Online Divisive-Agglomerative Clustering
LDS	Linear Dynamical System
SVD	Singular Value Decomposition
EEG	Electroencephalography
CA-AR	Coercively Adjusted Auto Regression
PCA	Principal Component Analysis
ARMA	Auto Regressive Moving Average
ARMAX	Auto Regressive Moving Average with Exogenous Inputs
SARMA	Seasonal Auto Regressive Moving Average
SMA	Seasonal Moving Average
SAR	Seasonal Auto Regression
MA	Moving Average
MLE	Maximum Likelihood Estimation
SSE	Sum of Squares Estimation
CSSE	Conditional Sum of Squares Estimation
CSS	Conditional Sum of Squares
BIC	Bayesian Information Criteron
BFGS	Broyden-FletcherGoldfarbShanno
MAPE	Mean Absolute Percentage Error
PAM	Partitioning Around Medoids
CM-u	Clustered Modeling Update
REG-ARIMA	Regression with Arima Errors

Chapter 4

Proposed Methodology

We now present our forecasting methodology for multiple co-evolving and correlated data series that is scalable, faster and more accurate compared to individual forecasting. The initial step is to fit a common model on data series to minimize their AICs, complement each other where possible, and remove redundancy in forecasting. Real time series are noisy and react to events resulting in local outliers. If the events are apriori known, they can be modeled easily. For example, during holidays there are more personal phone calls, and less calls made by commercial customers. If an event is ad hoc and lacks a certain pattern, it would introduce noise to an individual model. By identifying data with similar models and fitting a common model on them, we avoid over-fits and remove the tendency to capture the local outliers. An individual SARIMA model cannot exploit these events to increase forecast accuracy.

We develop the notion of similarity for an effective clustering in the context of forecasting. Then for each cluster, we devise a representative model that minimizes the forecasting errors. These representatives are updated continuously as the data keeps streaming. We then enhance the representative models and re-cluster the data until AIC no further decreases. Representative models are iteratively updated and the data is clustered to minimize errors. Forecasts are performed by applying the representative model to each series independently, and parameters are updated incrementally as data evolve.

Following the above approach, we first present our clustered modeling that simultaneously builds and enhances the models while clustering. We then develop representative models utilizing time series clustering. Our first approach includes k model building and n application of a model to a series, whereas the second approach and individual modeling require building n models. 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 the formula (3.9) and estimating the residuals. Next period forecasts for each series are derived by applying the corresponding model and using this model. We note that this approach of applying the model to a data series is negligible in time compared to building the model from scratch. However, as the application process involves the data series itself, it also produces accurate results; in fact more accurate on average than building the model from scratch.

We formalize 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 $X = (X_1, X_2, ..., X_N)$, let β_i be the model parameters of the time series X_i minimizing its $AIC(\beta_i; X_i)$, we seek whether the proposition below is true or not;

$$\forall\tau, \sum_{X_i \in P} Error(\tau; X_i, P) > \sum_{X_i \in P} Error(\beta_i; X_i), \tag{4.1}$$

where $Error(\beta_i; X_i)$ is the forecast error of time series X_i using model parameters β_i and $Error(\tau; X_i, P)$ is the forecast error of time series X_i using a common model estimated on a subset P of X. The left hand side of the inequality represents a group of time-series modeled collectively where all the time series $X_j \in P$ share the same model parameters τ. The right hand side represents the errors of individual models. As also evidenced in 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 clearly decreases the time required to give each time series a successful model.

4.1 Clustered Modeling and Forecasting

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

Definition ***(Model Cluster):*** A model cluster $C(F, X)$ is a set of time series $X = (X_1, X_2, ..., X_m)$, and a common forecasting model F with parameters τ where $AIC(\tau; X_i)$ is minimum over all clusters.

Based on this definition, model clustering of a set of time series $X = (X_1, X_2, ..., X_m)$ is a partitioning $P = (P_1, P_2, ..., P_l)$ of these time series and a vector of forecasting models $F = (F_1, F_2, ..., F_l)$ where the model F_i is a common model for all the time series in the set P_i. The common model F_i includes model parameters and the variance of the white noise is specific to each time series in a cluster.

Analogous to a cluster center, a forecasting model F_i of a cluster is the best (closest) in minimizing the AIC. More formally let τ_i be the parameters of the model F_i then

$$\tau_i = \arg\min_{\tau} \sum_{x \in X} AIC(\tau; x). \tag{4.2}$$

Assuming that we are given a cluster $C(F, X)$, we need to find a way to adjust the parameter vector β of F so that it is optimal for each time series belonging to the corresponding cluster. 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 total AIC will be equal to minimizing AICs individually which would give us the best model for each series. Similar to this, our approach is to minimize total AIC but using a set of grouped models instead of modeling these series individually. For this purpose, we define the following optimization function for each cluster i, which is the basis of our approaches;

$$minimize \sum_{x \in P_i} AIC(\beta^i; x), \tag{4.3}$$

where β^i is the parameter vector of the SARIMA model minimizing above on the set of P_i. The optimization we introduced in (4.3) is generic to any time-series modeling approach, not specific to SARIMA models. Thus our approach can be applied to any time-series modeling. To avoid the computational burden of searching through the order space we assume that the number of parameters does not change in each cluster but the values of the parameter vectors change. So minimizing total AIC with respect to model parameter vector will be equal to minimizing the negative of the sum of the log-likelihoods, or the (un)conditional sum-of-squares and eventually the followings:

$$minimize\ \psi(P_i) = -\sum_{X_j \in P_i} \ell(\beta_i; X_{ij}) \tag{4.4}$$

or

$$minimize\ \psi(P_i) = \sum_{X_j \in P_i} CSS(\beta_i; X_{ij}). \tag{4.5}$$

For minimizations in (4.4) and (4.5), we utilize iterative nonlinear optimization algorithms. We use quasi-Newton method (BFGS) [23] as it is parameter free and relatively fast. 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. In Appendix I, we give the gradient of our optimization function which will be required for BFGS algorithm.

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 easily update common models for each cluster and preserve accuracy. Algorithm 1 shows the general outline of clustered modeling. The problems in how to construct and maintain proper clusters are to i) find an initial representative model F for each cluster, ii) appropriately assign a time series to one of the clusters given, iii) enhance representative models for every cluster, iv) update representative models as new data arrive, and v) forecast future data points.

Algorithm 1 Clustered Modeling

```
clustered-modeling(tseries,k, M)
 tseries: s multiple time series
 k: number of clusters
 M: a minimization algorithm
  1. $\{C_i^{(0)}\} \leftarrow initialize(tseries, k, M)$
  2. $\{C_i^{(1)}\} \leftarrow form-clusters(tseries, \{C_i^{(0)}\}_{i=1}^{k})$
  3. $t \leftarrow 1$
  4. $tseries^{(0)} \leftarrow tseries$
while new data points arrive do
    Update $tseries^{(t-1)}$ with new data, $tseries^{(t)}$
    $\beta^{(t)} \leftarrow update(tseries^{(t)}, M, \beta^{(t-1)})$
    Forecast future data points
end while
```

4.1.1 Finding Initial Representatives

We first handle the problem of finding the initial clusters of series, and initial model selection for each cluster. To search for an appropriate initial model of a cluster, we need the number of parameters and a modeling schema. We first group time-series into k clusters with equal number of time-series. We estimate the center of each cluster. We use the median of each time point and construct the "median time-series" as cluster centers to handle outlier time points. The number of parameters is estimated by fitting an optimal SARIMA model on median time-series minimizing its AIC. Given the estimated orders of the common model (p, q, P, Q, d), we can estimate the parameter values by minimizing (4.4) and (4.5). Algorithm 2 gives the details of the initial model selection.

$model(P_i^0, M, o_i)$ estimates the best model parameters minimizing (4.4) and (4.5) given a minimization schema M, time-series of the corresponding cluster P_i^0, and the orders o_i.

Algorithm 2 Finding initial representative models

```
initialize(tseries,k, M)
 tseries: s multiple time series
 k: number of clusters
 M: a minimization algorithm
  1. Group time-series into k clusters $C_i^{(0)}(F_i^{(0)}, P_i^{(0)})$ for $i = 1, 2, ..., k$
  2. Estimate median time-series of each cluster, $T_i$, for $i = 1, 2, ..., k$
  3. $v_i = \arg\min_\beta AIC(\beta; T_i)$
  4. Extract orders $o_i = (p, q, P, Q, d)_i$ from $v_i$
  5. $\beta_i^0 \leftarrow model(P_i^0, M, o_i)$ for $i = 1, 2, ..., k$
  6. return $\{C_i^{(0)}(F_i^{(0)}, P_i^{(0)})\}$
```

4.1.2 Forming the Model Clusters

There are two problems that need to be considered when forming the clusters. The first one is how to assign a time series to a cluster, and the second is how to enhance the representative models. We first deal with assignment of a time series. Given a set of clusters $C_1, C_2, ..., C_n$, the best approach to select the appropriate cluster for a time series X_i would be to assign a time series $X_i \in P_k$ from C_k to C_j and then update the model parameters of C_k and C_j after the assignment. The cluster C_j that causes most total AIC reduction is the new cluster of time series X_i. Although this approach seems to work well, it needs to update model parameters at every consideration of every series, which becomes infeasible as data size grows. Assigning a new object to a cluster will change its model but this may cause an increase in the AIC of some objects while reducing the AIC of the others. This may increase total AIC of time series. We need a fast update scheme that guarantees the decrease in AIC.

Theorem 1. *Given a minimization algorithm M and two clusters $C_p(F_p, P_p)$ and*

$C_q(F_q, P_q)$ having model parameters β_p and β_q, respectively, with $X_t \in C_p$, if

$$AIC(\beta_p; X_t) > AIC(\beta_q; X_t), \tag{4.6}$$

assigning X_t from C_p to C_q always decreases total AIC.

Proof. Let $X = \{X_1, X_2, , X_N\}$ be the set of time series available, $P_p^{'} = P_p \backslash \{X_t\}$, $P_q^{'} = P_q \cup \{X_t\}$ and $\beta_p^{'}$ and $\beta_q^{'}$ be the parameter vectors of clusters $C_p^{'}$ and $C_q^{\prime}$, respectively, adjusted by M after the assignment of X_t from C_p to C_q. Assigning X_t from C_p to C_q is the best approach if

$$\begin{aligned}\sum_{X_i \in P_p^{\prime} \cup P_q^{\prime}} AIC(\beta_p^{'}; X_i) &< \sum_{X_i \in P_p \cup P_q} AIC(\beta_p^{'}; X_i) \\ &= \sum_{X_i \in P_p^{\prime} \cup P_p^{\prime}} AIC(\beta_p; X_i) + AIC(\beta_p; X_t) - AIC(\beta_q; X_t).\end{aligned}$$

As M will update the model parameter vectors as long as the total AIC decreases, it will never increase total AIC of $C_p^{'}$ and $C_q^{'}$ after assignment of X_t from C_p to C_q, and following statements will be satisfied:

$$\sum_{X_i \in P_p^{\prime}} AIC(\beta_p^{'}; X_i) < \sum_{X_i \in P_p^{\prime}} AIC(\beta_p; X_i)$$

and

$$\sum_{X_i \in P_q^{\prime}} AIC(\beta_q^{'}; X_i) < \sum_{X_i \in P_q^{\prime}} AIC(\beta_q; X_i)$$

At the simplest level, it will keep parameter vectors the same and the relations above will strictly be equalities. Based on the relations we obtained, if (4.2) is satisfied then

$$\begin{aligned}AIC(\beta_p; X_t - AIC(\beta_q; X_t) + \sum_{X_i \in P_p^{\prime} \cup P_q^{\prime}} AIC(\beta_p; X_i) &> \sum_{X_i \in P_p^{\prime} \cup P_q^{\prime}} AIC(\beta_p; X_i) \\ \sum_{X_i \in P_p \cup P_q} AIC(\beta_p; X_i) &> \sum_{X_i \in P_p^{\prime} \cup P_q^{\prime}} AIC(\beta_p^{'}; X_i).\end{aligned}$$

Thus the inequality in (4.6) will be satisfied. □

Based on the theorem above, for a time series X_t we choose the cluster having the smallest AIC to assign it into. To avoid empty clusters, we skip the reconsideration of the series in singleton clusters.

After we find the best cluster for each time series, we update the models for the altered clusters using (4.4) and (4.5). We continue the reassignment and update steps successively until the average AIC can no further be improved. Forming the model clusters is given in Algorithm 3.

Algorithm 3 Forming the Model Clusters

```
*form-clusters(tseries,* ${\{C_i^{(0)}\}}_{i=1}^{k}$*, $\epsilon$)*

 tseries : s multiple time series

 ${\{C_i^{(0)}\}}_{i=1}^{k}$ : the set of initial clusters

 $\epsilon$ : Stopping margin for average AIC

  1. $\triangle AIC^{(0)} \leftarrow \infty$

  2. $t \leftarrow 0$

  3. Estimate $AIC(\beta_j; X_i)$ by (3.13) and (3.14) for $i = 1, 2, ..., s, j = 1, 2, ..., k$

**while** $\triangle AIC^{(t)} > \epsilon$ **do**
    Assign $X_i$ to the cluster $C_l^t$ where
        $l = \arg\min_j AIC(\beta_j; X_i)$ for $i = 1, 2, ..., s$

    $|\beta_i^{(t)}| \leftarrow update(P_i^{(t)}, M, |\beta_i^{(t-1)}|)$

    Estimate $AIC(\beta_j; X_i)$ by (3.13, 3.14) for $i = 1, 2, ..., s, j = 1, 2, ..., k$

    $t \leftarrow t + 1$

    $AIC^{(t)} \leftarrow 1/s \sum_{j=1}^{k} \sum_{X \in P_j} AIC(\beta_j; X)$

    $\triangle AIC^{(t)} \leftarrow (AIC^{(t-1)} - AIC^{(t)})/AIC^{(t-1)}$
**end while**
return ${\{C_i^{(t)}\}}_{i=1}^{k}$
```

4.1.3 Updating Model Parameters

As new data points arrive, we update models in each cluster instead of modeling from scratch. Given the time series updated by new data points *tseries*, model updates are accomplished using the function

$$\beta_i = update(tseries, M, \beta^{(t-1)}),$$

which uses a minimization algorithm M initialized with $\beta^{(t-1)}$, and estimates next parameters $\beta^{(t)}$ minimizing the total AIC of *tseries*. We initialize the minimization algorithm M with previous parameters $\beta^{(t-1)}$ because this hastens the convergence as clusters also stabilize in time.

4.1.4 Forecasting Future Data Points

Let's assume that given a time series X, h-step ahead forecasts is 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 τ then we give the h-step ahead forecasts of X using the following equation.

$$\tilde{X}_{t+h} = f(X; \tau, h). \tag{4.7}$$

4.2 Modeling on LPC-based Clustering

Although most time-series clustering and representation approaches are not specifically designed for forecasting, we also investigate ways to utilize them in forecast modeling. The intuition of our approach here is similar to the proposed method in previous section, where identifying and applying a suitable common model is more accurate than building a separate model for each data series. Using a time series representation, we cluster data and assign models build on each cluster center as the corresponding representative model for the cluster. 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 $C_i(M_i, P_i)$, $i = 1, 2, , k$ using a representation and a distance measure where M_i is the cluster center, and P_i is the time series in cluster C_i. Next we fit a SARMA model F_i to each of the cluster centers M_i. We construct our model clusters by transforming previous clusters using the following equation:

$$C_i^{'}(F_i, P_i) = C_i(s(M_i), P_i), \tag{4.8}$$

where s gives the SARMA model for M_i by minimizing the corresponding AIC.

Any representation and clustering approach can be utilized within this methodology. We specifically adapt LPC (Linear Prediction Cepstrum) coefficients which is the cepstral representation of the Linear Prediction Coefficients, used in speech and image processing [24]. It was also shown to be effective for time series clustering in terms of silhouette coefficient [11]. We 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 [25]. 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 [1]. Thus based on our SARMA definition with the integration, AR representation of the corresponding SARIMA model can be found by solving

$$\phi^{'}(B) = \sum_{i=0}^{\infty} \phi_i^{'} x_{t-i} = \frac{\Phi_P B^S \phi(B)(1-B)^d}{\Theta_P B^S \theta(B)} x_t = w_t. \tag{4.9}$$

Given an AR representation $\phi^{'}(B)$, LPC coefficients can be defined as follows [24]:

$$c_i = \begin{cases} -\phi_1' & \text{if } i = 1 \\ -\phi_i' - \sum_{j=1}^{i-1} (1 - \frac{j}{i})\phi_j' c_{i-j} & \text{if } 1 < i \le p \\ -\sum_{j=1}^{i-1} (1 - \frac{j}{i})\phi_j' c_{i-j} & \text{if } p < i \end{cases}. \tag{4.10}$$

Our experimental results confirm that while traditional representations of PCA, DWT and DFT do not provide meaningful forecasts, LPC provides highly accurate results. However, its execution time is comparable to individual modeling and significantly slower than clustered modeling presented in Section 4.1. The LPC based approach requires models of each time series in advance, which is not that case in the clustered modeling approach.

4.3 REG-ARIMA Modeling and Forecasting

We also propose to use Regression with Arima Errors (REG-ARIMA) model to capture information such as holidays, special days, etc. in time series. We try to utilize regression model to handle information of unexpected events in time series. A Regression with Arima Errors model is a combination of a regression model and an ARIMA model. For a REG-ARIMA model, firstly a regression model is applied to time series, then an ARIMA model is applied to residuals of regression model. One can build a SARIMA model on residuals of regression model instead of ARIMA model. A REG-ARIMA model is in the form of equation (4.11)

$$X_t = b_0 + b_1 X_{1,t} + b_2 X_{2,t} + ... + b_k X_{k,t} + N_t, \tag{4.11}$$

where $X_{1,t}, X_{2,t}, ..., X_{k,t}$ are the k explanatory variables and N_t is the error term contains auto correlations modeled as in formula (4.12)

$$\Phi_P(B^s)\phi(B)x_t = \Theta(B^s)\theta(B)w_t, \tag{4.12}$$

where w_t is white noises series.

4.3.1 Individual REG-ARIMA Modeling

We propose to use individual REG-ARIMA modeling and compare its performance to clustered modeling, individual modeling and clustered REG-ARIMA modeling. Individual REG-ARIMA modeling is the methodology described in Section 4.3. Summation of regression predicts and SARIMA forecasts give the total forecast. Experimental results show that it is more accurate than individual modeling.

4.3.2 Clustered REG-ARIMA Modeling

Clustered REG-ARIMA modeling uses REG-ARIMA modeling in the context of clustered modeling. Clustered modeling given in Section 4.1 is applied on residuals of regression models of time series. For the forecasting, predicted values from regression models and forecast valued from clustered modeling are summed up.

Chapter 5

Performance Evaluation

We demonstrate efficiency, accuracy, and scalability of the proposed approaches compared to the alternatives, in particular to the individual forecasting approaches. We observed that each individual forecast with a SARIMA model takes several seconds to minutes, hence even a couple of thousand series cannot be updated continuously with individual modeling. For example, on a traffic load of 2497 VIP commercial customers of a major telecom company, individual modeling takes 173 minutes and clustered modeling takes around nine minutes, 20 times faster. This gives 4.16 seconds on average for each individual modeling for traffic. The results also show that our method is scalable, and increasing the number of series does not degrade the accuracy and time performance.

5.1 Experimental Setup

We used one real data set and four synthetic data sets in our experiments. The first data set is a telco traffic time-series consisting 2497 series each having a length of 867 time points. These are traffic load generated by major commercial customers of a telecom company. The time series shows highly seasonal behavior and is sensitive to special and unexpected events, e.g. weekends, holidays, sudden events, campaigns. We also generated several synthetic data sets based on this real data through aggregations, introducing random local outliers, and enhancing its size following the time series generation methodology presented in [11]. 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 $\sigma = 0.05$ which preserves the invertibility and causality of the generated SARMA model. We generated four datasets each having 100,000 time series. Our clustered modeling approach intrinsicly handles stationary issues by differencing and Box-Cox transformations. The modeling part of our system utilizes the packages in R Project that involves several statistical packages useful in our analyses [26]. To fit the best model minimizing (3.13) or (3.14) of a single time series, we use the R Package [27].

To show performance of our proposed method, we took the first 839 time points for building model and we made forecasts for later 28 points. To evaluate the dynamic update of clustered modeling, 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),

$$MAPE = \frac{1}{h}\left(\sum_{i=1}^{h}\left|\frac{x_{n+i} - f_i}{x_{n+i}}\right|\right) \tag{5.1}$$

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 ones coming from individual forecasts?
- How does the speed of our clustered modeling change compared to the average speed of the individual fits?
- How does the accuracy and speed of clustered modeling change compared to other clustering algorithms we adjusted for forecasting purposes?

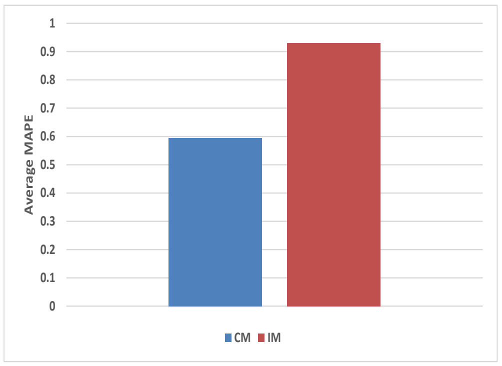

Figure 5.1: MAPE results of CM and IM

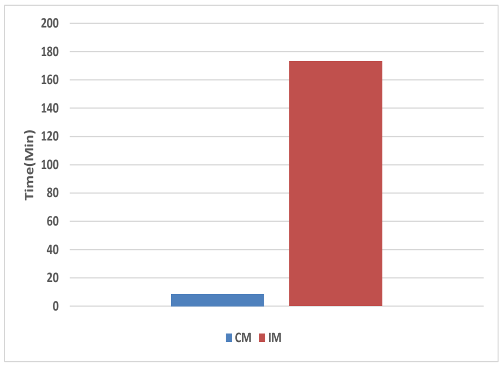

Figure 5.2: CM and IM time

- How does the number of clusters affect the speed and accuracy results?
- How does the accuracy and speed of our dynamic update change?
- Is the proposed method scalable?

Our results first focus on telecom VIP customer traffic which originally motivated this work. Our software is in active use by the company. We also present results on synthetic data sets. We give results to answer the questions above, then we compare clustered modeling to representations and clustering algorithms we adjusted for forecasting. We finally experiment for scalability of the algorithms.

5.2 Efficiency and Accuracy of Initialization

Considering our first question, Fig. 5.1 shows the average MAPE results over all time series of individual modeling and clustered modeling. It is seen that, clustered modeling gives better accuracy results compared to individual modeling.

We take the weekly average forecast error over all time series. The error of clustered modeling is 0.59, and for individual modeling it is 0.93. On average clustered modeling provides 37% improvement on weekly MAPE over individual modeling.

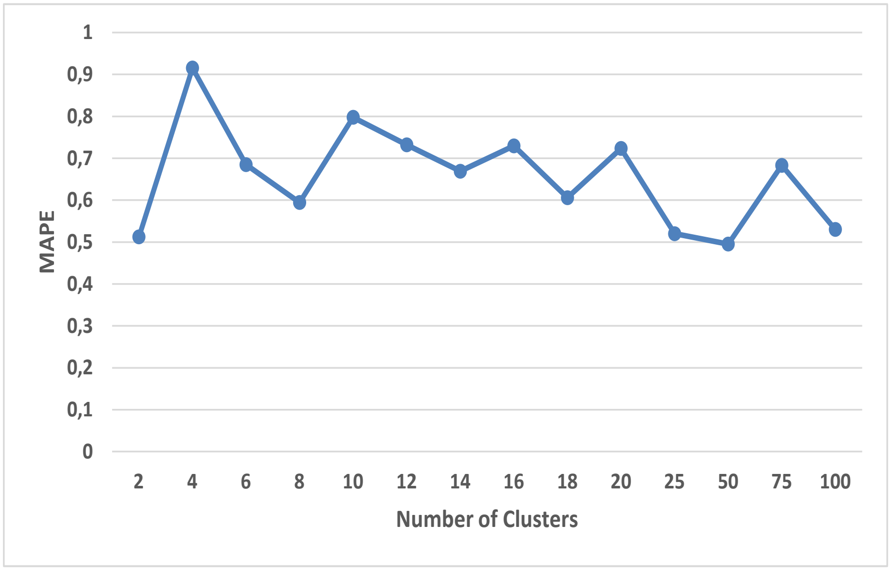

Figure 5.3: Average MAPE vs number of clusters using CM

Fig. 5.2 shows the time requirement of clustered modeling and individual modeling. While individual modeling takes hours to fit models, clustered modeling is significantly faster by providing this in minutes. On average clustered modeling takes only nine minutes, while it takes 173 minutes to model each time series individually.

In Fig. 5.3, we observe the relationship between the number of clusters and the accuracy of clustered modeling solution. Although there is a slight decrease in MAPE, MAPE and the number of clusters does not have regular pattern. We find the best number of cluster having a local minimum. We increase the clusters one at a time, until we observe a second increase in MAPE. So we use eight clusters for VIP time series.

We investigate the relationship between the number of clusters and the time that clustered modeling requires. Fig. 5.4 shows that increasing the number of clusters slightly increase the running time. The reason is that, optimization algorithm is almost linear in time series size. The default number of clusters is set to 8, which has a considerably low running time.

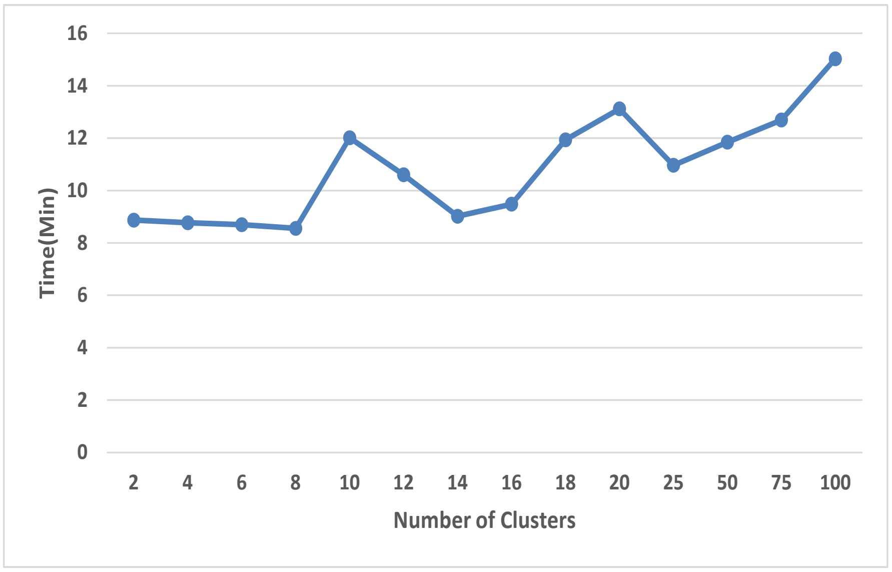

Figure 5.4: Average time for each number of clusters using CM

5.3 Results on Time Series Clustering Approaches

We performed experiments adapting time-series clustering using 5 different representations, LPC, DFT, DWT, PCA, and raw data. We use the resulting clusters and raw time series to forecast future points. With median and mean of the time series belonging to each cluster, we have 10 different approaches. We use first 10 features extracted using one 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 is shown to be descriptive enough for these approaches [11]. To the best of our knowledge, our work presents the first results on clustering for forecasting purposes.

Fig. 5.5 shows the comparison of modeling using representations with Euclidean distance. For DFT, DWT, PCA and raw data, clustered modeling provides considerably better results in comparison to both median and mean approaches. The only exception is LPC which works better than clustered modeling which has 0.22 and 0.22 MAPE results for median and mean respectively while clustered modeling has 0.59. While LPC is more accurate than clustered modeling, it requires SARMA models to be available to construct cepstral coefficients. Thus it is computationally much more expensive than the clustered modeling approach. The execution time of each algorithm is presented in Fig.

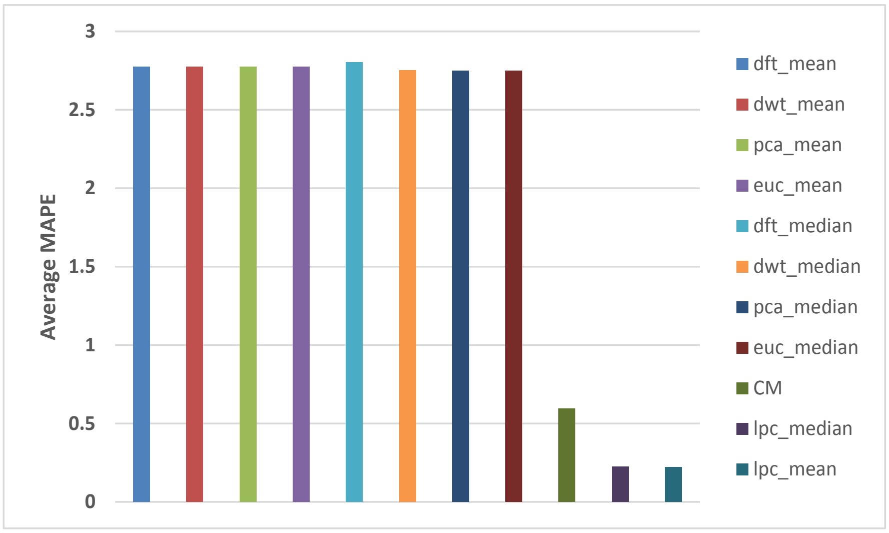

Figure 5.5: MAPE results of CM vs time series clustering approaches

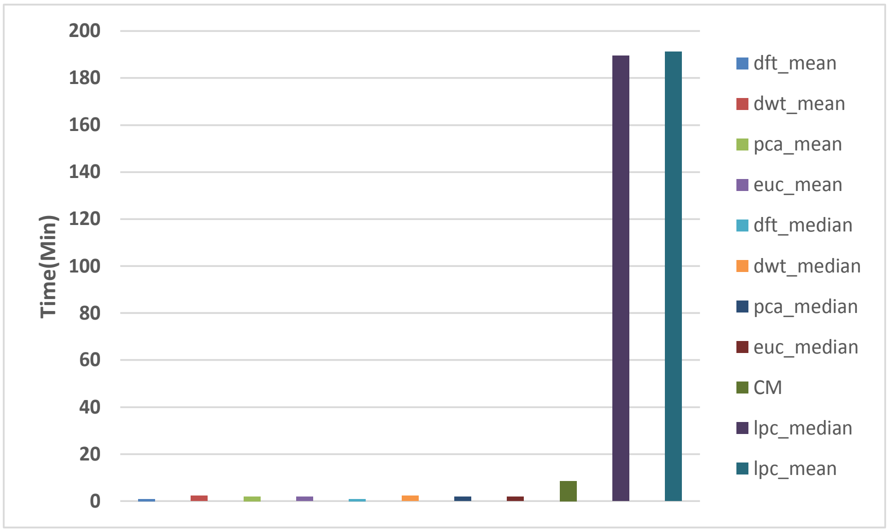

Figure 5.6: Comparison of the running time of CM with existing time series representations

5.6. While DFT, DWT, PCA and raw data can manage faster results, LPC is doing as worse as individual modeling in terms of time. On average LPC takes 189 and 191 minutes for median and mean respectively.

Overall, LPC provides the most accurate results, but has the same efficiency problem with individual modeling. LPC is very suitable in small-scale applications where the models can easily be obtained. For small-scale data, LPC based approach is preferable over individual modeling as it significantly improves accuracy. For large-scale data, clustered modeling is preferable over both as individual modeling and LPC based approaches are infeasible due to efficiency and scalability problems. As new time points come, cepstral coefficients have to be updated as well as 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, clustered modeling is both fast and accurate and hence more practical.

5.4 Dynamic Update of Model Parameters

As our optimization algorithms are iterative in nature, we can easily 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 new data. To compare, we re-run our clustered modeling algorithm from the scratch and show that our Clustered Modeling Update (CM-u) takes less time than re-run thus transitively it takes considerably less time than individually fitting data as new points come.

Fig. 5.7 shows the comparison of MAPE results of clustered modeling update as new points come with with re-run of clustered modeling and individual modeling. Clustered modeling update and re-run are comparable with each other, and both are more accurate than individual modeling. Clustered modeling update takes 1.32 minutes which is faster than both re-run and individual modeling.

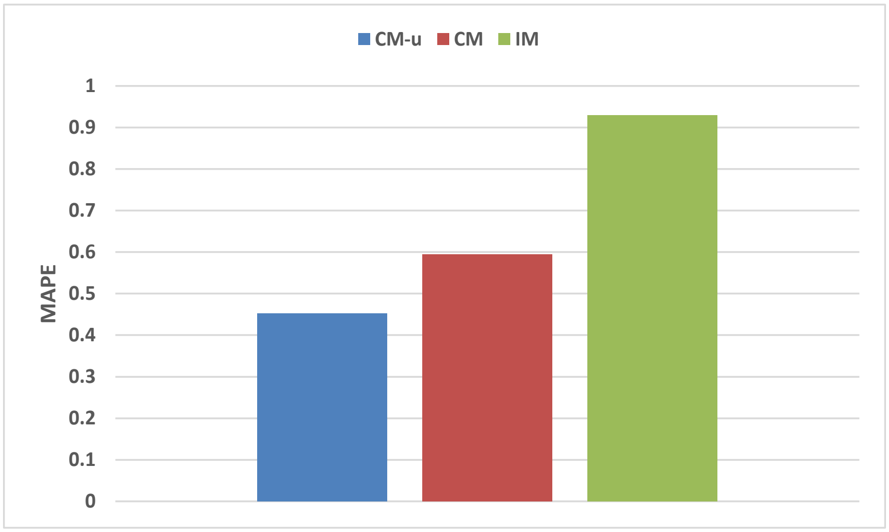

Figure 5.7: MAPE comparison of our CM-u algorithm with re-run and IM

The reason is that clusters stabilize at the end of clustered modeling, and clustered modeling update converges very fast as we initialize it with the resulting clusters of clustered modeling. This is summarized in Fig. 5.8.

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. 5.9 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. 5.10 exhibits a linear relationship between the size of the data and the time it takes. As the data size doubles, clustered modeling takes approximately double time but with a very small slope compared to individual modeling.

5.5 Experiments for Large Dataset

We first compared clustered modeling update modeling with clustered modeling and individual modeling. We then compared accuracy and running time performance of clustered modeling and individual modeling as the data size increases on synthetic data set.

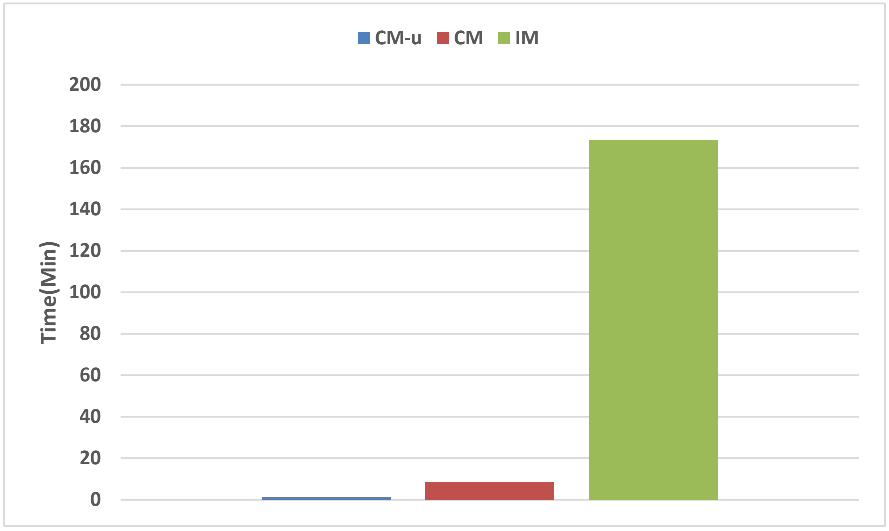

Figure 5.8: The time of CM-u algorithm and re-run vs. IM

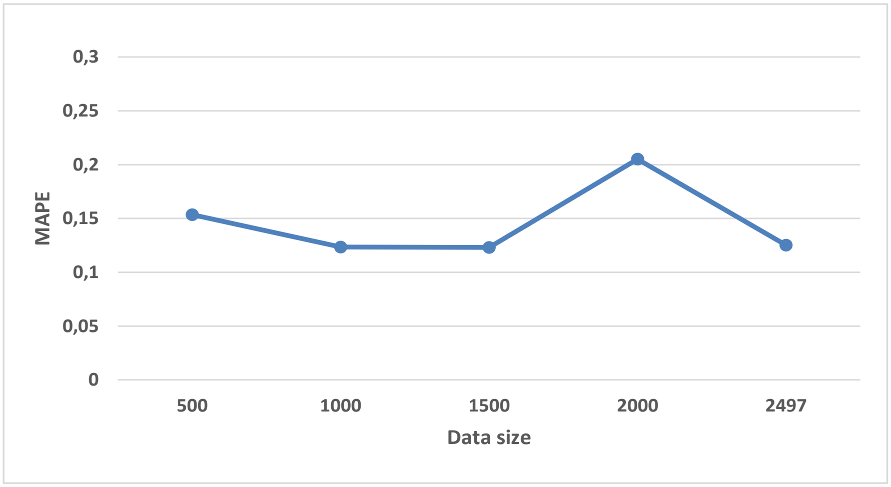

Figure 5.9: Accuracy of 500 time series as the size of the data increases using CM

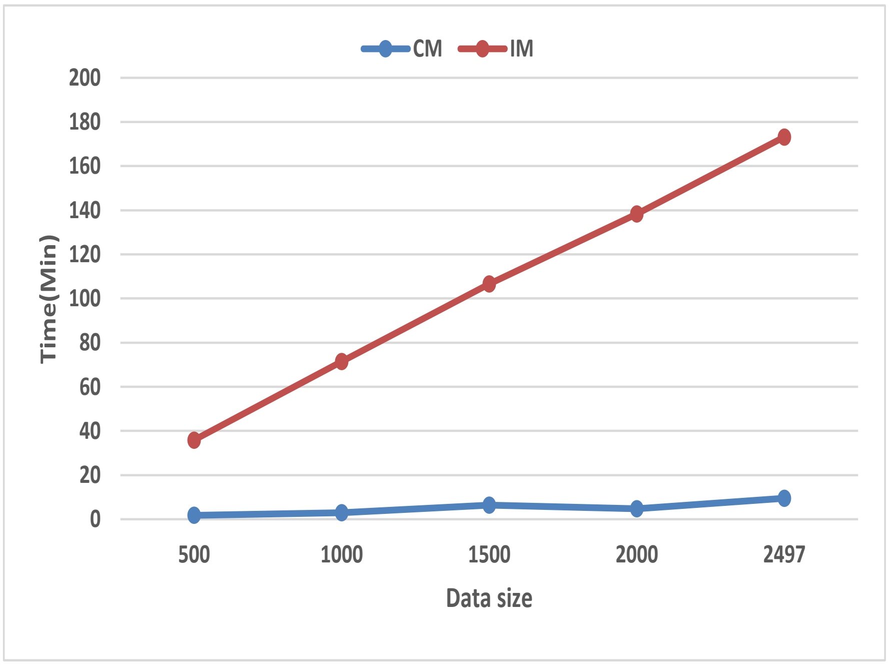

Figure 5.10: The processing time of CM vs. IM as the size of the data increases

Fig. 5.11 illustrates the accuracy comparison of the proposed approach with individual modeling. We vary the dataset size from 2500 to 100,000 and give average results. It shows that clustered modeling and clustered modeling update has lower MAPE than individual modeling, and clustered modeling update competes with clustered modeling. This shows that, instead of building clustered modeling from start, we can use clustered modeling update to obtain the same accuracy.

Fig. 5.12 shows the time that clustered modeling, clustered modeling update, and individual modeling takes. It takes 43 minutes and 6.5 hours for clustered modeling update and clustered modeling respectively, while it takes 126 hours for individual modeling. On average our clustered modeling solution is 19 times faster than individual modeling. If we have available clusters, we can further improve results using clustered modeling update which is 173 times faster than individual modeling.

5.6 Evaluations for Scalability

We run clustered modeling over 4 synthetic datasets to evaluate the scalability of clustered modeling. We divide time series into 10 parts to build individual models to fasten the process. We then sum up the times. Fig. 5.13 shows the

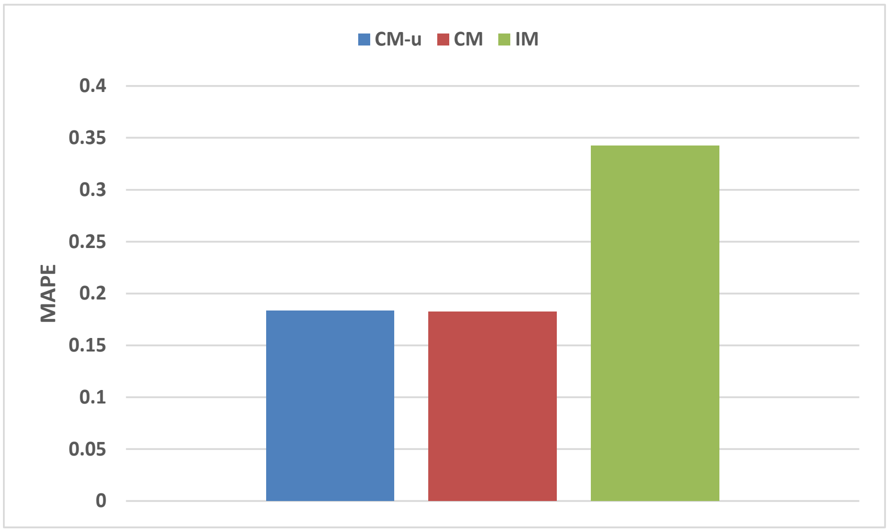

Figure 5.11: MAPE results of CM-u, re-run and IM for large data set

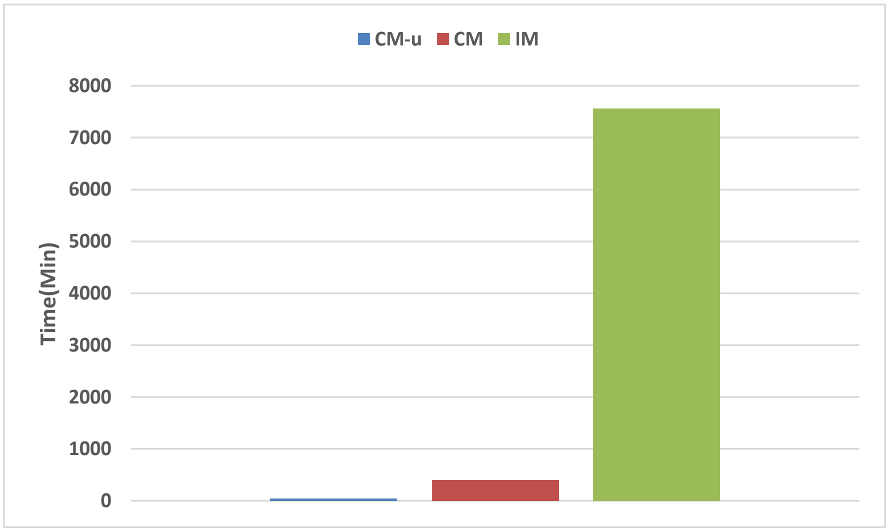

Figure 5.12: Time results of CM-u, re-run, and IM for large data set

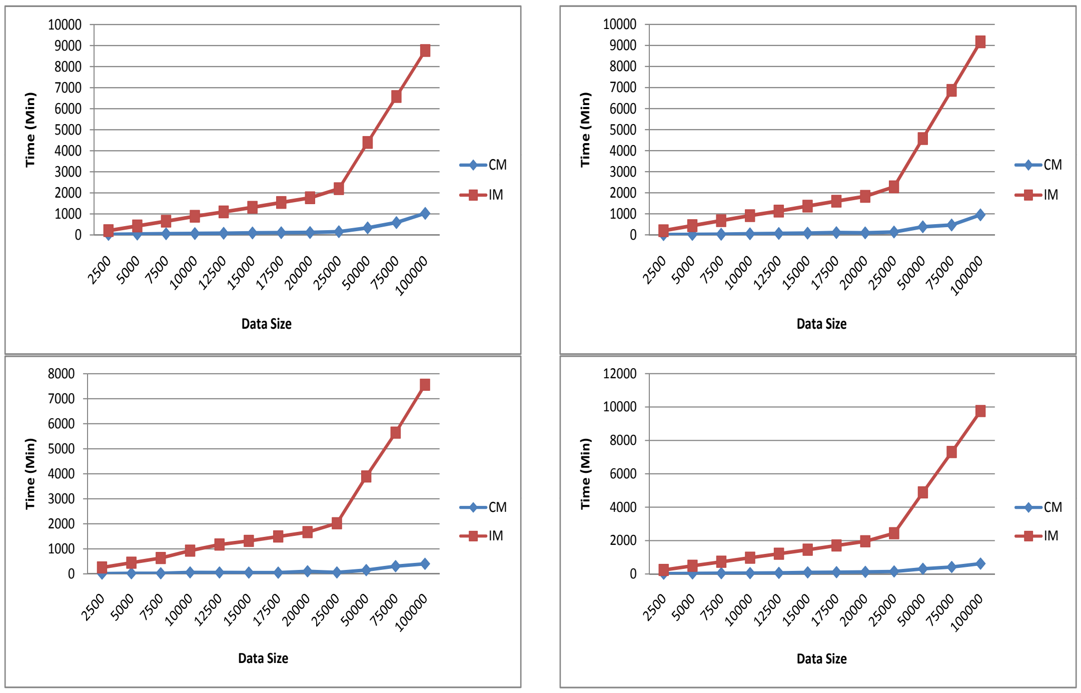

Figure 5.13: Time of CM and IM on large data set

time requirements for clustered modeling and individual modeling over 4 synthetic datasets. On all 4 dataset, clustered modeling is more scalable than individual modeling. On average, clustered modeling is 17.5 times faster than individual modeling. It takes 12 hours for clustered modeling to build models while it takes 147 hours for individual modeling over 100,000 time series. These results show that clustered modeling is more scalable than individual modeling. We should note that putting all the time series in memory can slow down processes. Because we double the size of the data after 25,000, the slope of the time for individual modeling changes. This is the result of in-memory calculations. But we always put time series to memory for clustered modeling. Thus the speed up improvements are lower bounds.

When the data has local outliers, clustered modeling results in significantly better accuracies than individual modeling, as it has an aggregate effect to remove outliers. If the data does not have any outliers, than clustered modeling is comparable with individual modeling. In two of the four datasets, the error of clustered modeling is around half of individual modeling. In the other two, clustered modeling and individual modeling are comparable within a 0.01 margin.

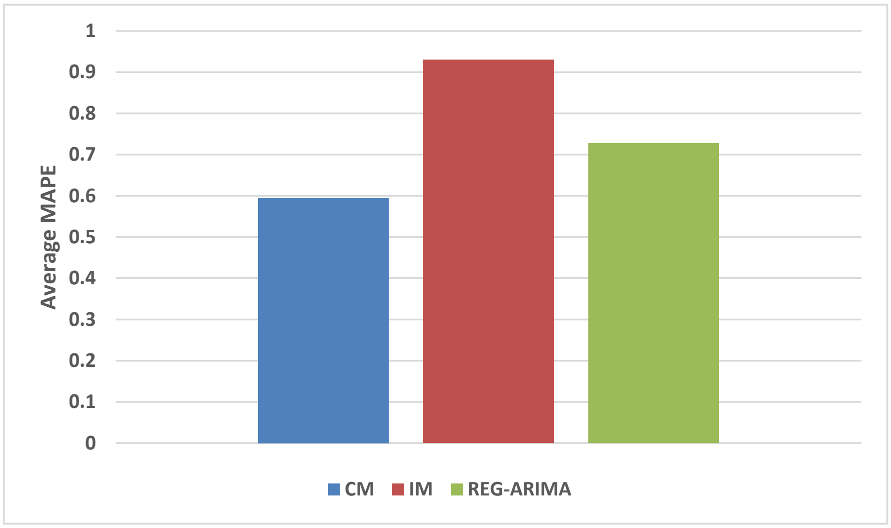

Figure 5.14: MAPE results of CM, IM and REG-ARIMA

5.7 Experiments for REG-ARIMA Modeling

We run individual REG-ARIMA modeling and clustered REG-ARIMA modeling on real telco traffic time series. Fig. 5.14 shows the accuracy results of clustered modeling, individual modeling and individual REG-ARIMA modeling. Individual REG-ARIMA modeling has lower MAPE than individual modeling, and clustered modeling has lower MAPE than both individual modeling and individual REG-ARIMA modeling. Clustered REG-ARIMA modeling is the worst when accuracy results compared. This is expected, as REG-ARIMA modeling has ability to capture special events it results better accuracies than individual modeling. Also since our clustered modeling performs better on time series that contain some special events or outliers, applying this model on residuals of regression models does not give good results as expected.

The processing time of clustered modeling, individual modeling and individual REG-ARIMA modeling is displayed on Fig. 5.15. Processing time for the individual REG-ARIMA modeling is slightly more than processing time for the individual modeling since individual REG-ARIMA modeling builds a regression model before SARIMA model for each time series.

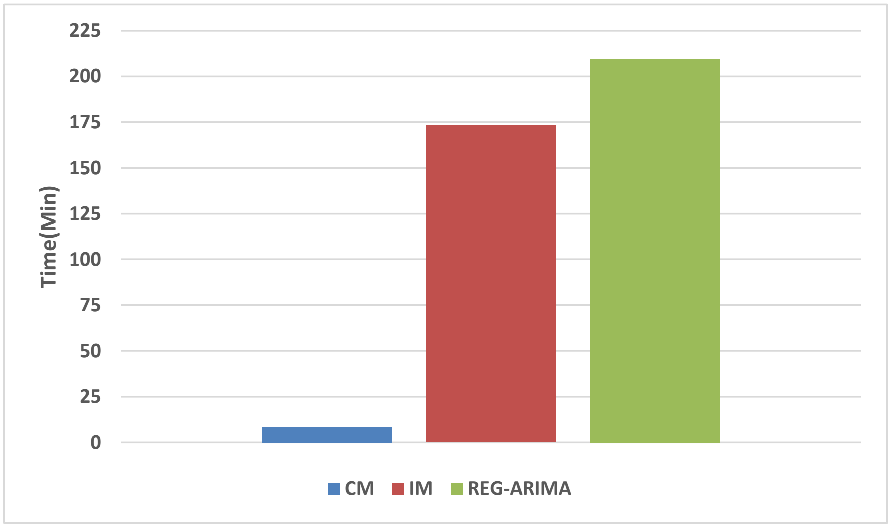

Figure 5.15: Processing time of CM, IM and REG-ARIMA

Chapter 6

Conclusions

We addressed the problem of continuous forecasting of multiple time series in the context of scalable data analytics, and proposed two approaches: one with clustering and modeling of data performed simultaneously, and another where data is first clustered then modeled. The proposed methodology is independent of the underlying linear or nonlinear modeling approach, and can benefit from any model selection method.

The first method is significantly faster and more scalable with comparable accuracies to individual modeling. We cluster time series according to their AIC values. A time series belongs to the cluster having the lowest AIC value for the corresponding time series. Thus two time series is considered similar if they have the lowest AIC values in the same cluster center. We improve each cluster model using iterative nonlinear optimization algorithms. This makes the approach also suitable for dynamic model updates as new time points arrive. Our clustered modeling paradigm is not restricted to SARIMA models and is applicable to any individual modeling approach with a given minimization procedure.

The second approach is significantly more accurate than individual modeling, with comparable run times to the individual approach. We adapt time-series feature extraction and clustering to forecasting. We used the invertibility of a SARMA model and construct LPC coefficients using AR representation of every

SARMA model. We used results of k-Means and PAM clustering structures and corresponding centers, and build SARMA models on each of the cluster centers. Forecasts for each individual time series are achieved using the model built on the center of their corresponding cluster. We showed that LPC based approach provides more accurate results than individual modeling with a negligible computational overhead.

We compare clustered modeling and modeling using representations to individual modeling and each other. Experimental results show that both of the proposed approaches perform significantly better than individual modeling. The clustered modeling is up to 20 times faster and 37% more accurate than individual modeling and modeling with LPC has 76% improvement over individual modeling with a speed overhead of 9% on real telco traffic series.

Bibliography

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

[2] S. G. Makridakis, S. C. Wheelwright, and R. J. Hyndman, *Forecasting: Methods and Applications.* John Wiley and Sons, 3rd ed., 1998.

[3] G. E. P. Box, G. M. Jenkins, and G. C. Reinsel, *Time Series Analysis: Forecasting & Control.* 3rd ed., 1994.

[4] K.-P. Chan and A. W. chee Fu, "Efficient time series matching by wavelets," in *In ICDE*, pp. 126–133, 1999.

[5] I. Kevecka, "Forecasting traffic loads: Neural networks vs. linear models," *Computer Modelling and New Technologies*, vol. 14, no. 2, pp. 20–28, 2010.

[6] M. Szmit and A. Szmit, "Usage of pseudo-estimator lad and sarima models for network traffic prediction: Case studies," in *Computer Networks* (A. Kwiecie, P. Gaj, and P. Stera, eds.), vol. 291 of *Communications in Computer and Information Science*, pp. 229–236, Springer Berlin Heidelberg, 2012.

[7] T. Warren Liao, "Clustering of time series data-a survey," *Pattern Recogn.*, vol. 38, pp. 1857–1874, Nov. 2005.

[8] L. Li and B. A. Prakash, "Time series clustering: Complex is simpler!," in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* (L. Getoor and T. Scheffer, eds.), ICML '11, (New York, NY, USA), pp. 185–192, ACM, June 2011.

[9] M. Corduas and D. Piccolo, “Time series clustering and classification by the autoregressive metric,” *Computational Statistics & Data Analysis*, vol. 52, pp. 1860–1872, January 2008.

[10] M. Kumar and N. R. Patel, “Using clustering to improve sales forecasts in retail merchandising,” *Annals OR*, vol. 174, no. 1, pp. 33–46, 2010.

[11] K. K. Dhiral, K. Kalpakis, D. Gada, and V. Puttagunta, “Distance measures for effective clustering of arima time-series,” in *In proceedings of the IEEE International Conference on Data Mining*, pp. 273–280, 2001.

[12] A. Alonso, J. Berrendero, A. Hernandez, and A. Justel, “Time series clustering based on forecast densities,” *Comput. Statist. Data Anal.*, vol. 51, pp. 762–776, 2006.

[13] P. P. Rodrigues, J. a. Gama, and J. Pedroso, “Hierarchical clustering of time-series data streams,” *IEEE Trans. on Knowl. and Data Eng.*, vol. 20, pp. 615–627, May 2008.

[14] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, “A framework for clustering evolving data streams,” *VLDB proceedings*, vol. 29, pp. 81–92, 2003.

[15] J. Lin, M. Vlachos, E. Keogh, and D. Gunopulos, “Iterative incremental clustering of time series,” in *Advances in Database Technology - EDBT 2004* (E. Bertino, S. Christodoulakis, D. Plexousakis, V. Christophides, M. Koubarakis, K. Bhm, and E. Ferrari, eds.), vol. 2992 of *Lecture Notes in Computer Science*, pp. 106–122, Springer Berlin Heidelberg, 2004.

[16] Y. Matsubara, Y. Sakurai, C. Faloutsos, T. Iwata, and M. 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, (New York, NY, USA), pp. 271–279, ACM, 2012.

[17] L. Li, B. A. Prakash, and C. Faloutsos, “Parsimonious linear fingerprinting for time series,” *Proc. VLDB Endow.*, vol. 3, pp. 385–396, Sept. 2010.

[18] L. Hong, D. Yin, J. Guo, and B. 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, (New York, NY, USA), pp. 484–492, ACM, 2011.

[19] D. Chakrabarti and C. Faloutsos, “Large-scale automated forecasting using fractals,” 2002.

[20] S.-H. Kim, C. Faloutsos, and H.-J. Yang, “Coercively adjusted auto regression model for forecasting in epilepsy EEG,” *Comp. Math. Methods in Medicine*, vol. 2013, 2013.

[21] F. Korn, H. V. Jagadish, and C. Faloutsos, “Efficiently supporting ad hoc queries in large datasets of time sequences,” *SIGMOD Rec.*, vol. 26, pp. 289–300, June 1997.

[22] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos, “Fast subsequence matching in time-series databases,” *SIGMOD Rec.*, vol. 23, pp. 419–429, May 1994.

[23] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, “A limited memory algorithm for bound constrained optimization,” *SIAM J. Sci. Comput.*, vol. 16, pp. 1190–1208, Sept. 1995.

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

[25] J. Makhoul, “Linear prediction: A tutorial review,” *Proceedings of the IEEE*, vol. 63, pp. 561–580, Apr. 1975.

[26] D. Bates, J. Chambers, P. Dalgaard, S. Falcon, R. Gentleman, K. Hornik, S. Iacus, R. Ihaka, F. Leisch, U. Ligges, T. Lumley, M. Maechler, D. Murdoch, P. Murrell, M. Plummer, B. Ripley, D. Sarkar, D. T. Lang, L. Tierney, and S. Urbanek, “R project.” `http://www.r-project.org/`.

[27] R. J. Hyndman and Y. Khandakar, “Automatic time series forecasting: The forecast package for R,” *Journal of Statistical Software*, vol. 27, pp. 1–22, 7 2008.

Appendix A

Gradient of Log-likelihoods

To construct an algorithm to update the model parameters for BFGS, we need to calculate the gradient of the equation in (4.4). Given that

$$\gamma = (\phi_{i1}, ..., \phi_{ip_i}, \Phi_{i1}, ..., \Phi_{iP_i}, \theta_{i1}, ..., \theta_{iq_i}, \Theta_{i1}, ..., \Theta_{iQ_i})$$

is the parameter set of the model belonging to cluster $C_i(F_i, P_i)$ where β_i is the parameter vector of the model F_i,

$$\begin{aligned}
\frac{\partial \Psi}{\partial \gamma_m} &= \sum_{X^{(j)} \in P} \frac{\partial \ell(\beta_i; X^{(j)})}{\partial \gamma_m} = \sum_{X_j \in P} \frac{\partial}{\partial \gamma_m} \left(-\frac{1}{2\sigma^2_{w^{(j)}}} \sum_{t=1}^{n} (w_t^{(j)})^2 \right) \\
&= \sum_{X_j \in P} \left(-\frac{1}{2\sigma^2_{w^{(j)}}} \sum_{t=1}^{n} 2 w_t^{(j)} \frac{\partial w_t^{(j)}}{\partial \gamma_m} \right), \\
&= -\sum_{X_j \in P} \frac{1}{\sigma^2_{w^{(j)}}} \sum_{t=1}^{n} w_t^{(j)} \frac{\partial w_t^{(j)}}{\partial \gamma_m},
\end{aligned}$$

where

$$\begin{aligned}
\frac{\partial w_t}{\partial \phi_{ik}} &= -(1 - \Phi_1 B^s - ... - \Phi_P B^{Ps}) x_{t-i}, \\
\frac{\partial w_t}{\partial \Phi_{ik}} &= -(1 - \phi_1 B - ... - \phi_p B^p) x_{t-si}, \\
\frac{\partial w_t}{\partial \theta_{ik}} &= -(1 + \Theta_1 B^s + ... + \Theta_Q B^{Qs}) w_{t-i}, \\
\frac{\partial w_t}{\partial \Theta_{ik}} &= -(1 + \theta_1 B + ... + \theta_q B^q) w_{t-si},
\end{aligned}$$

and

$$X^{(j)} = (x_1^{(j)}, x_2^{(j)}, ..., x_n^{(j)}),$$

Thus

$$\Delta \varPsi = \left(\frac{\partial \varPsi}{\partial \gamma_1}, \frac{\partial \varPsi}{\partial \gamma_2}, ..., \frac{\partial \varPsi}{\partial \gamma_{(p_i + P_i + q_i + Q_i)}} \right).$$

We provide matrix forms of these equations next. Thanks to this form, one can also utilize the large body of software packages that are optimized for matrix operations.

$$\begin{aligned}
\frac{\partial \Psi}{\partial \phi_m} &= \sum_{X_j \in P} \frac{\partial \ell(\beta_i; X^{(j)})}{\partial \phi_m} \\
&= \left([1, -\Phi_1, ..., -\Phi_P] \sum_{j=1}^{n} \frac{1}{\sigma_j^2} \begin{bmatrix} w_t^{(j)} x_{t-m}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-m}^{(j)} \\ \vdots & \ddots & \vdots \\ w_t^{(j)} x_{t-m-Ps}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-m-Ps}^{(j)} \end{bmatrix} \right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \\
\frac{\partial \Psi}{\partial \Phi_m} &= \sum_{X_j \in P} \frac{\partial \ell(\beta_i; X^{(j)})}{\partial \Phi_m} \\
&= \left([1, -\phi_1, ..., -\phi_P] \sum_{j=1}^{n} \frac{1}{\sigma_j^2} \begin{bmatrix} w_t^{(j)} x_{t-sm}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-sm}^{(j)} \\ \vdots & \ddots & \vdots \\ w_t^{(j)} x_{t-sm-p}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-sm-p}^{(j)} \end{bmatrix} \right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \\
\frac{\partial \Psi}{\partial \theta_m} &= \sum_{X_j \in P} \frac{\partial \ell(\beta_i; X^{(j)})}{\partial \theta_m} \\
&= \left([1, -\Theta_1, ..., -\Theta_Q] \sum_{j=1}^{n} \frac{1}{\sigma_j^2} \begin{bmatrix} w_t^{(j)} x_{t-m}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-m}^{(j)} \\ \vdots & \ddots & \vdots \\ w_t^{(j)} x_{t-m-Qs}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-m-Qs}^{(j)} \end{bmatrix} \right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \\
\frac{\partial \Psi}{\partial \Theta_m} &= \sum_{X_j \in P} \frac{\partial \ell(\beta_i; X^{(j)})}{\partial \Theta_m} \\
&= \left([1, -\theta_1, ..., -\theta_q] \sum_{j=1}^{n} \frac{1}{\sigma_j^2} \begin{bmatrix} w_t^{(j)} x_{t-sm}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-sm}^{(j)} \\ \vdots & \ddots & \vdots \\ w_t^{(j)} x_{t-sm-q}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-sm-q}^{(j)} \end{bmatrix} \right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}
\end{aligned}$$

Appendix B

Gradient of Conditional Sum of Squares

To construct an algorithm to update the model parameters for BFGS, we need to calculate the gradient of the equation in (4.5). Given that

$$\gamma = (\phi_{i1}, ..., \phi_{ip_i}, \Phi_{i1}, ..., \Phi_{iP_i}, \theta_{i1}, ..., \theta_{iq_i}, \Theta_{i1}, ..., \Theta_{iQ_i})$$

is the parameter set of the model belonging to cluster $C_i(F_i, P_i)$ where β_i is the parameter vector of the model F_i,

$$\frac{\partial \Psi}{\partial \gamma_m} = \sum_{X^{(j)} \in P} \frac{\partial CSS(\beta_i; X^{(j)})}{\partial \gamma_m} = \sum_{X_j \in P} \frac{\partial}{\partial \gamma_m} \left(\sum_{t=p+1}^{n} \hat{w}_t^2(\beta) \right)$$

where $\hat{w}_t = E(w_t | x_1, x_2, ..., x_p)$ and if $t > p$

$$\hat{w}_t = w_t$$

$$\sum_{X_j \in P} \frac{\partial}{\partial \gamma_m} \left(\sum_{t=p+1}^{n} \hat{w}_t^2(\beta) \right) = 2 \sum_{X_j \in P} \sum_{t=p+1}^{n} w_t^{(j)} \frac{\partial w_t^{(j)}}{\partial \gamma_m},$$

where

$$\frac{\partial w_t}{\partial \phi_{ik}} = -(1 - \Phi_1 B^s - ... - \Phi_P B^{Ps}) x_{t-i},$$
$$\frac{\partial w_t}{\partial \Phi_{ik}} = -(1 - \phi_1 B - ... - \phi_p B^p) x_{t-si},$$
$$\frac{\partial w_t}{\partial \theta_{ik}} = -(1 + \Theta_1 B^s + ... + \Theta_Q B^{Qs}) w_{t-i},$$
$$\frac{\partial w_t}{\partial \Theta_{ik}} = -(1 + \theta_1 B + ... + \theta_q B^q) w_{t-si},$$

and

$$X^{(j)} = (x_1^{(j)}, x_2^{(j)}, ..., x_n^{(j)}),$$

Thus

$$\Delta\Psi = \left(\frac{\partial \Psi}{\partial \gamma_1}, \frac{\partial \Psi}{\partial \gamma_2}, ..., \frac{\partial \Psi}{\partial \gamma_{(p_i + P_i + q_i + Q_i)}} \right).$$

Here are the matrix forms of these equations.

$$\begin{aligned}
\frac{\partial \Psi}{\partial \phi_m} &= \sum_{X_j \in P} \frac{\partial CSS(\beta_i; X^{(j)})}{\partial \phi_m} \\
&= \left([1, -\Phi_1, ..., -\Phi_P] \sum_{j=p+1}^{n} 2 \begin{bmatrix} w_t^{(j)} x_{t-m}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-m}^{(j)} \\ \vdots & \ddots & \vdots \\ w_t^{(j)} x_{t-m-Ps}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-m-Ps}^{(j)} \end{bmatrix} \right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \\
\frac{\partial \Psi}{\partial \Phi_m} &= \sum_{X_j \in P} \frac{\partial CSS(\beta_i; X^{(j)})}{\partial \Phi_m} \\
&= \left([1, -\phi_1, ..., -\phi_P] \sum_{j=p+1}^{n} 2 \begin{bmatrix} w_t^{(j)} x_{t-sm}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-sm}^{(j)} \\ \vdots & \ddots & \vdots \\ w_t^{(j)} x_{t-sm-p}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-sm-p}^{(j)} \end{bmatrix} \right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \\
\frac{\partial \Psi}{\partial \theta_m} &= \sum_{X_j \in P} \frac{\partial CSS(\beta_i; X^{(j)})}{\partial \theta_m} \\
&= \left([1, -\Theta_1, ..., -\Theta_Q] \sum_{j=p+1}^{n} 2 \begin{bmatrix} w_t^{(j)} x_{t-m}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-m}^{(j)} \\ \vdots & \ddots & \vdots \\ w_t^{(j)} x_{t-m-Qs}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-m-Qs}^{(j)} \end{bmatrix} \right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \\
\frac{\partial \Psi}{\partial \Theta_m} &= \sum_{X_j \in P} \frac{\partial CSS(\beta_i; X^{(j)})}{\partial \Theta_m} \\
&= \left([1, -\theta_1, ..., -\theta_q] \sum_{j=p+1}^{n} 2 \begin{bmatrix} w_t^{(j)} x_{t-sm}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-sm}^{(j)} \\ \vdots & \ddots & \vdots \\ w_t^{(j)} x_{t-sm-q}^{(j)} & \cdots & w_{n-t+1}^{(j)} x_{n-sm-q}^{(j)} \end{bmatrix} \right) \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}
\end{aligned}$$