Universidade Federal do Ceará
Centro de Ciências
Pró-Reitoria de Pesquisa e Pós-Graduação
Mestrado e Doutorado em Ciência da Computação

S-SWAP: SCALE-SPACE BASED WORKLOAD ANALYSIS AND PREDICTION

Gustavo Adolfo Campos dos Santos

DISSERTAÇÃO DE MESTRADO

Fortaleza
Outubro - 2013

Universidade Federal do Ceará
Centro de Ciências
Pró-Reitoria de Pesquisa e Pós-Graduação

Gustavo Adolfo Campos dos Santos

S-SWAP: SCALE-SPACE BASED WORKLOAD ANALYSIS AND PREDICTION

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial para a obtenção do título de Mestre.

Orientador: Prof. Dr. Javam de Castro Machado
Co-orientador: Prof. Dr. José Gilvan Rodrigues Maia

Fortaleza
Outubro - 2013

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca de Ciências e Tecnologia

S235s Santos, Gustavo Adolfo Campos dos.
S-SWAP: scale-space based workload analysis and prediction. / Gustavo Adolfo Campos dos Santos. – 2013.
82f. : il. color., enc. ; 30 cm.

Dissertação (mestrado) – Universidade Federal do Ceará, Centro de Ciências, Departamento de Computação, Programa de Pós-Graduação em Ciência da Computação, Fortaleza, 2013.
Área de Concentração: Sistemas de informação.
Orientação: Prof. Dr. Javan de Castro Machado.
Coorientação: Prof. Dr. José Gilvan Rodrigues Maia.

1. Computação em nuvem. 2. Análise de séries temporais. 3. Teoria da previsão. I. Título.

CDD 005

S-SWAP: Scale-Space based Workload Analysis and Prediction

Gustavo Adolfo Campos dos Santos

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal do Ceará, como requisito parcial para a obtenção do título de Mestre.

Prof. Dr. Javam de Castro Machado
Universidade Federal do Ceará

Prof. Dr. José Gilvan Rodrigues Maia
Universidade Federal do Ceará

Prof. Dr. José Antônio Fernandes de Macêdo
Universidade Federal do Ceará

Prof. Dr. Edmundo Roberto Mauro Madeira
Universidade Estadual de Campinas

Aos meus pais, Jonas e Iara, e aos meus irmãos, Victor e Felipe.

ACKNOWLEDGEMENTS

Agradeço a Deus, por me dar coragem para enfrentar os inúmeros obstáculos impostos, e por me guiar para mais essa conquista.

Aos meus pais e heróis, Jonas e Iara, por todo amor e esforço incondicional dedicados em todos os momentos.

Aos meus irmãos e melhores amigos, Victor e Felipe, pelo constante apoio e pela convivência feliz que sempre tivemos.

Ao meu orientador, professor Javam Machado, pela oportunidade concedida, e pela confiança depositada para a realização deste trabalho e paciência nos momentos mais delicados.

Ao meu co-orientador, professor Gilvan Maia, pelos valiosos conhecimentos transmitidos, pela paciência nas inúmeras reuniões ao longo deste processo e pelo constante incentivo nos momentos de desânimo.

Ao professor Leonardo Moreira, pela força, pelo didatismo e paciência em todas as discussões e por sempre ter estado disposto a colaborar de alguma forma para o sucesso deste trabalho.

Ao professor Flávio Sousa, pelas discussões enriquecedoras e pelo auxílio com a bibliografia.

A cada um dos professores do Departamento de Computação da Universidade Ferderal do Ceará com os quais tive o prazer de aprender algo que contribuiu para minha formação.

Aos amigos do mestrado e do grupo de pesquisa em bancos de dados, ARiDa, com quem compartilhei valiosos momentos de estudo e intercâmbio de conhecimentos, momentos de angústia e de descontração.

Aos meus colegas de trabalho e amigos da Secretaria de Tecnologia da Informação da Universidade Federal do Ceará, STI-UFC, pelas palavras de incentivo, pela compreensão e pelos bons momentos de descontração.

Ao diretor executivo da STI-UFC, professor José Antônio, e ao diretor adjunto, professor José Ramos, pelo apoio para a realização deste mestrado.

A todos os amigos que direta ou indiretamente colaboraram para o sucesso deste trabalho, e que compreenderam o seu significado, incentivando mesmo nos muitos momentos em que estive ausente do seu convívio.

Aos bons amigos que fiz no laboratório BiMo, na Universidade Federal do Ceará, pelo apoio, pelos momentos de descontração e por me terem permitido usufruir da utilização do seu espaço de forma sempre solícita.

A todos aqueles com quem tive algum contato, e que, por mais breve que tenha sido, algo acrescentaram a minha maneira de enxergar o mundo.

Ao gênero 'chill out' de música, por me ajudar a manter meu quociente de paciência estável nos longos momentos de experimentos, de escrita e de introspecção.

Ph'nglui mglw'nafh Cthulhu R'lyeh wgah'nagl fhtagn.
—THE CALL OF CTHULHU, BY H. P. LOVECRAFT

ABSTRACT

This work presents a scale-space based approach to assist dynamic resource provisioning. The application of this theory makes it possible to eliminate the presence of irrelevant information from a signal that can potentially induce wrong or late decision making.

Dynamic provisioning involves increasing or decreasing the amount of resources allocated to an application in response to workload changes. While monitoring both resource consumption and application-specific metrics is fundamental in this process since the latter is of great importance to infer information about the former, dealing with these pieces of information to provision resources in dynamic environments poses a big challenge. The presence of unwanted characteristics, or noise, in a signal that represents the monitored metrics favors misleading interpretations and is known to affect forecast models.

Even though some forecast models are robust to noise, reducing its influence may decrease training time and increase effciency. Because a dynamic environment demands decision making and predictions on a quickly changing landscape, approximations are necessary. Thus it is important to realize how approximations give rise to limitations in the forecasting process. On the other hand, being aware of when detail is needed, and when it is not, is crucial to perform efficient dynamic forecastings. In a cloud environment, resource provisioning plays a key role for ensuring that providers adequately accomplish their obligation to customers while maximizing the utilization of the underlying infrastructure. Experiments are shown considering simulation of both reactive and proactive strategies scenarios with a real-world trace that corresponds to access rate. Results show that embodying scale-space theory in the decision making stage of dynamic provisioning strategies is very promising. It both improves workload analysis, making it more meaningful to our purposes, and lead to better predictions.

Keywords: workload analysis, forecast, scale-space.

CONTENTS

Chapter 1—Introduction 1

1.1 Motivation and Problem Characterization 1

1.2 Objectives and Contributions 3

1.3 Publications 4

1.4 Structure of the Thesis 5

Chapter 2—Theoretical Background 6

2.1 Scale-Space 6

2.2 Predictive Analytics 9

2.2.1 Time-Series Forecasting 9

2.2.1.1 ARIMA 11

2.2.2 Machine Learning 12

2.2.2.1 Support Vector Machines 13

2.3 Error Metrics 17

2.4 Conclusion 19

Chapter 3—Workload Analysis 20

3.1 Proactive Solutions 20

3.1.1 Prediction of Cloud Data Center Networks Loads Using Stochastic and Neural Models [1] . . . 20
3.1.2 PRESS [2] . . . 21
3.1.3 RPPS [3] . . . 22
3.1.4 Marlowe [4] . . . 23
3.1.5 Predictive Data Grouping and Placement for Cloud-based Elastic Server Infrastructures [5] . . . 24
3.1.6 SmartSLA [6] . . . 25
3.1.7 Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud [7] . . . 26
3.2 Reactive Solutions . . . 28
3.2.1 SLA-Based and Consumer-Centric Dynamic Provisioning for Cloud Databases [8] . . . 28
3.2.2 RepliC [9] . . . 29
3.3 Discussion and Comparison . . . 29
3.4 Conclusion . . . 32

Chapter 4—Scale-Space based Workload Analysis and Prediction 34

4.1 Workload . . . 34
4.2 Objective . . . 36
4.3 Architecture . . . 40
4.4 Implementation . . . 42
4.5 SVM Parametrization . . . 43
4.6 Conclusion . . . 48

Chapter 5—Experimental Evaluation 49

5.1 Experimental Setup 49

5.2 Experiments 51

5.2.1 The Effect of Scale-Space Over a Monitored Signal 51

5.2.2 Forecasting 53

5.2.2.1 Short-Range Forecasting 53

5.2.2.2 Long-Range Forecasting 63

5.3 Conclusion 69

Chapter 6—Conclusion 72

6.1 Results 72

6.2 Future Works 73

LIST OF FIGURES

2.1 A multi-scale representation of a signal. 6

2.2 Interpolating the signal obtained by a high value of *'s'* in the scale-space algorithm with the original one helps bringing the extrema close to their original amplitude. This is very helpful, since the detection of extrema plays a key role in resource provisioning. 8

2.3 A schematic illustration, obtained from [10], of SVR using ϵ-sensitive loss function. 14

4.1 Workload trace during the enrollment period is shown in (a), which happened from 01/22/2012 to 01/31/2012. In (b), a workload trace of 50min, from the pointed region in (a), is presented. 36

4.2 Our approach. Scale-space filtering is optionally applied to a signal $f_i(x)$, which generates another signal $L_i(x, s)$ in a new scale. The obtained signal might be used by a reactive provisioning algorithm. Also it can be used at the prediction module to generate a forecast that may be used by a proactive provisioning algorithm. 38

4.3 Direct Forecasting 39

4.4 Sliding-Window Forecasting 40

4.5 Sampling 41

4.6 S-SWAP architecture. 42

4.7 The influence of the hyper-parameter ε for fixed values of $C = 1$ and γ set to SVM-TOY's default value, $\frac{1}{k}$, where k is the number of attributes in the input. In our case, $\gamma = 0,5$. Note that, in (d), even increasing the constant C to very high values, the region remains unchanged, since loss occurs only if a point lies outside the ε-insensitive region. 44

4.8 The influence of the hyper-parameter C for fixed values of $\varepsilon = 0,03$ and $\gamma = 100$. 45

4.9 Gaussian RBF with centre c = 0 and radius r = 1 46

4.10 The influence of the hyper-parameter γ for fixed values of $C = 10$ and $\varepsilon = 0,03$. 46

5.1 Experimental Setup 49

5.2 Application of scale-space algorithm over a region of the workload (a) and a shorter range of it (b). The shorter timespan in (b) is obtained by removing samples from the right of the interval shown in (a). 52

5.3 Application of moving average algorithm over a region of the workload (a) and a shorter range of it (b). The shorter timespan in (b) is obtained by removing samples from the right of the interval shown in (a). 53

5.4 Application of scale-space algorithm with different values of *"s"* for the same signal. In (a) we have s = 1.5 and in (b) s = 5.0. 54

5.5 ARIMA behavior over a signal with scale-space applied to and considering both no sliding-window and *steps 1 and 2* sliding-windows of size 40 . . . 55

5.6 ARIMA behavior over a signal with scale-space applied to and considering both no sliding-window and *steps 1 and 2* sliding-windows of size 50 . . . 56

5.7 ARIMA behavior over a signal with scale-space applied to and considering both no sliding-window and *steps 1 and 2* sliding-windows of size 60 . . . 57

5.8 In (a), we show SVR behavior compared to ARIMA over a signal with scale-space applied to. No sliding-window is applied, and we consider an observation window of size 40 and prediction window of size 24. In (b), we include *steps 1* sliding-window. In (c), no scale-space was applied to the signal and also no sliding-window. 59

5.9 SVR and ARIMA forecasting for an observation window of size 50 and prediction window of size 24. In (a) we have scale-space smoothed signal, while in (b) we have the original one. 60

5.10 Example of a situation in which none of the forecasting methods perform accurate time-series forecasting. 61

5.11 Example of a case in which both methods show good accuracy for a diminished prediction window of 12min. In (a), forecasting is performed over the scale-space smoothed signal and based on an abservation window of 40min. In (b), observation window is increased to 50min and the result obtained is, in (c), compared to forecasting over the original signal. . . . 62

5.12 Sequence of short-range forecasting over a period of one day. In (a) and (b), ARIMA and SVR applied over the scale-space smoothed signal. In (c) and (d), both methods are applied over the same region in the original signal. 63

5.13 Interpolating the signal obtained with the scale-space *"s"* parameter set to s = 120 helps bringing the extrema close to their original amplitude. This is very helpful, since the detection of extrema plays a key role in resource provisioning. 64

5.14 SVR and ARIMA forecasting for an observation window of size 5829 and prediction window of size 700. In (a) and (b) we have scale-space smoothed signal (SS) with the *"s"* parameter set to s = 120, while in (c) and (d) we have the original one (OS). 65

5.15 *RMSE* values for long-range predictions with ARIMA and SVR considering min, max, med and avg sampling methods for 1/3rd, 1/5th, 1/7th and 1/10th of the total points. Results for both the scale-space smoothed signal and the original one are presented. 67

5.16 *MAPE* values for long-range predictions with ARIMA and SVR considering min, max, med and avg sampling methods for 1/3rd, 1/5th, 1/7th and 1/10th of the total points. Results for both the scale-space smoothed signal and the original one are presented. 68

5.17 *PRED(25)* values for long-range predictions with ARIMA and SVR considering min, max, med and avg sampling methods for 1/3rd, 1/5th, 1/7th and 1/10th of the total points. Results for both the scale-space smoothed signal and the original one are presented. 69

5.18 Training times obtained with ARIMA and SVR considering min, max, med and avg sampling methods for 1/3rd, 1/5th, 1/7th and 1/10th of the total points. Results for both the scale-space smoothed signal and the original one are presented. 70

5.19 SVR and ARIMA forecasting for an observation window of size 5829 and prediction window of size 700. In (a) and (b) we have scale-space smoothed signal (SS) with the *"s"* parameter set to s = 120, while in (c) and (d) we have the original one (OS). In all the cases, we have 1/5th of the points obtained through average 71

LIST OF TABLES

3.1 33

5.1 ARIMA's total training time in milliseconds for observation windows of size 40, 50, 60 with no sliding-window (NSW), *step 1* sliding-window (SW1) and *step 2* sliding-window (SW2). 55

5.2 Errors obtained for ARIMA prediction without sliding-window, (NSW), and with *step 1* sliding-window, (SW1), and *step 2* sliding-window, (SW2) for observation windows of sizes 40min, 50min and 60min. 58

5.3 Errors obtained for both ARIMA and SVR prediction with an observation window of size 40min and prediction window size of 24. Values are presented for three different cases: with scale-space applied and no sliding-window (SS, NSW); with scale-space and *step 1* sliding-window (SS, SW1) and finally with no scale-space and no sliding-window (OS, NSW) 58

5.4 Errors obtained for both ARIMA and SVR prediction with an observation window of 50min and prediction window of 24min. Values are presented for predictions over scale-space smoothed signal (SS), and the original one (OS). 60

5.5 Errors obtained for both ARIMA and SVR prediction with observation windows of 40min and 50min and prediction window of 12min. Values are presented for predictions over scale-space smoothed signal (SS), and the original one (OS). 61

5.6 ARIMA and SVR training times in milliseconds for observation windows of size 40 and 50. In the first case results are show over the scale-space smoothed signal (SS), in the latter both scale-space smoothed (SS) and original signal (OS) values are shown for comparison. 61

5.7 Comparison SVR x ARIMA in both the Original Signal (OS) and Scale-Space smoothed signal (SS) for an interval of one day of short-range predictions. 62

5.8 Comparison SVR x ARIMA in both the Original Signal (OS) and Scale-Space smoothed signal (SS) for an observation window of 5829min and a prediction window of 700min. 65

5.9 Comparison SVR x ARIMA in both the Original Signal (OS) and Scale-Space smoothed signal (SS) for an observation window of 5829min and a prediction window of 700min. 65

5.10 Comparison SVR x ARIMA in both the original ignal (OS) and scale-space smoothed signal (SS) for 1/5th of the all the points, which are obtained through average. 68

5.11 Comparison SVR x ARIMA in both the Original Signal (OS) and Scale-Space smoothed signal (SS) for 1/5th of the all the points, which are obtained through average. 69

CHAPTER 1

INTRODUCTION

This master's thesis presents *Scale-Space based Workload Analysis and Prediction (S-SWAP)*, an approach based on the scale-space theory to assist dynamic resource provisioning strategies. It intends to provide means to extract relevant information out of the analysed data and to obtain good performance from the adopted forecasting techniques. In this chapter we present the motivation to the development of this work and list the objectives we intend to achieve.

1.1 MOTIVATION AND PROBLEM CHARACTERIZATION

Capacity planning has been around for quite some time, with roots in everything from economics to engineering. In a basic sense, capacity planning is resource management. It determines what your system needs and when it needs it [11]. In order for service providers, which may include from distributed systems, to web applications and cloud environments, to assure high performance and availability to their customers, effective and careful capacity planning must be performed. It aims utilizing resources efficiently while simultaneously supporting a large customer base and meeting Quality of Service (QoS) requirements. In that sense, application-specific metrics, like response time, throughput and access rate, to name a few, are also important to be considered in dynamic systems, since they usually present strong correlation with the demand for resources in such environments.

Since services in a cloud environment are offered on a pay-per-usage basis, customers are more stringent to have their demands met. Cloud providers are thus supposed to meet service level objectives (SLO), which may be composed of one or more QoS measurements, agreed with the customer through a Service Level Agreement (SLA), being subject to penalties for service unavailability. As such, a platform that provides cloud computing services must be able to dynamically provision its various resources (e.g. computing, storage and networking) to its various users according to their instantaneous

needs and in compliance with negotiated SLAs. It is their goal to maximize revenues by minimizing both the number of resources available and the incurred penalties. The necessity of finding the proper tradeoff between having an ammount of resource available to meet SLA requirements and maximizing revenues makes efficient resource provisioning a special need to the cloud computing paradigm. It plays a key role ensuring that the service providers adequately accomplish their obligation to costumers while maximizing the utilization of underlying infrastructure [12]. In this scenario, it is clear that one needs to avoid both under-provisioning, which leads to application slowdown, and over-provisioning, which leads to unnecessary resource costs [13]. Service providers can reduce the inefficiency caused by these situations by optimizing the number of active servers to support a given user base [14].

Dynamic provisioning techniques are designed to handle workload fluctuations [8] so that SLA violations and their associated contractual penalties can be avoided or limited, and reduce cost. These techniques usually take actions based on workload observation and can be classified as either reactive or proactive. Proactive solutions apply sophisticated system models for prediction [15] and use resulting forecasts for triggering allocations in advance of expected need. In contrast, reactive approaches do not use prediction, but rather detect and react to existing resource bottlenecks by means of predefined thresholds for application-level or system metrics. In the last years, researchers have been employing both these techniques in order to deal with the resource provisioning issue [9][8][16][13] [1][3][2][4][6][5].

Even though releasing resources is usually not such a complex task, acquiring resources incurs performance overheads [17], specifically their setup time, but also, in case of the database tier, the time needed to handle replication and synchronization of the associated disk state of the database in order to preserve data integrity [18]. Once these actions take certain time to be effectively performed, it is suitable to tackle the resource demand problem from the time series analysis perspective, in which data are represented in a line graph that records the values of a given set of variables (datapoints) that describes the system's state within a period of time. This representation plays a central role in the development of resource provisioning approaches.

Thus, independently on which approach is used, reactive or proactive, being able to determine the degree of relevance of an observed datapoint and, consequently, also being able to eliminate the influence of irrelevant ones are desirable features. In our context, being a relevant datapoint means whether it is in a peak or in a depression

in the observed signal that lasts a period of time long enough to justify the addition or removal of resources. If it is not the case, then it is irrelevant and might misguide system's decisions if kept for consideration. This may lead to the addition (removal) of resources that will most likely be removed (added) in a short period of time, thus causing unecessary overheads. An important observation is that irrelevant datapoints also include noisy data, which are often indicative either of measurement error or some system instability that may have occurred during monitoring. Instabilities in the studied phenomena may also happen due to a monitoring strategy that interferes with the system's normal operation stressing or altering it.

1.2 OBJECTIVES AND CONTRIBUTIONS

In this research work, we apply scale-space theory [19] to assist dynamic resource provisioning strategies. This theory is a qualitative signal description in multi-scale measurement [20], thus enabling multiple interpretations of the data. Also, it allows us to derive such multi-scale representations in a mathematically sound way [21]. By means of choosing the right scale, it is a powerful technique for eliminating, or considerably diminishing, the presence of irrelevant information from a signal that can potentially induce wrong or late decision making by both proactive and reactive methods. More importantly, it does so with mathematical guarantees that no additional structures, i.e. new peaks and depressions, are introduced in the process as the original sampling is kept unchanged over time. This is a significant difference between this approach and similar representations usually applied elsewhere. We also apply this technique to two widely used forecasting methods, Support Vector Machines for Regression (SVM) and Autoregressive Integrated Moving Average (ARIMA), which are representative, respectively, from machine learning and from statistical time-series analysis. We compare both theses methods reactions to the appliaction of scale-space smoothing. As such, we make an effort in trying to get the most out of them in order to maximize the benefits that may be extracted from our approach.

Thus, our main contributions are:

- we specifically carry out time-series analysis in order to eliminate irrelevant information and thus obtain a signal that better explains the underlying behavior of interest;

- we apply scale-space theory in the context of dynamic provisioning of resources, which, to the best of our knowledge, has not yet been experimented. Thus, information provided might be used in decision-making for provisioning algorithms;
- we show that the Support Vector Machine for Regression (SVR) ϵ-SVR [22] and the Autoregressive Integrated Moving Average (ARIMA) [23] forecasting models benefit from our approach and yield reasonable results;
- we implement an extensible workload analysis and prediction framework that can be used in dynamic environments.

The appoach is robust to noise and may be used together with both reactive and proactive solutions. This time-series analysis constitutes the core of our work and we can easily extend it to other time-series of interest even in the multivariate context.

1.3 PUBLICATIONS

The efforts during the research process for this thesis made it possible the following publication:

- SANTOS, G. A. C. ; MAIA, J. G. R. ; MOREIRA, L. O. ; SOUSA, F. R. C. ; MACHADO, J. C. "Scale-Space Filtering for Workload Analysis and Forecast". In: *IEEE 6th International Conference on Cloud Computing (IEEE CLOUD)*, 2013.

Although the following publications are not direct related to this research work, many concepts and ideas have been developed together with this master thesis:

- MOREIRA, L. O. ; SOUSA, F. R. C. ; MAIA, J. G. R. ; FARIAS, V. A. E. ; SANTOS, G. A. C. ; MACHADO, J. C. "A Live Migration Approach for Multi-Tenant RDBMS in the Cloud". In: *28th Brazilian Symposium on Databases (SBBD)*, 2013.
- FARIAS, V. A. E. ; MAIA, J. G. R. ; SOUSA, F. R. C. ; MOREIRA, L. O. ; SANTOS, G. A. C. ; MACHADO, J. C. "A Machine Learning Approach for SQL Queries Response Time Estimation in the Cloud". In: *28th Brazilian Symposium on Databases (SBBD)*, 2013.

- SOUSA, F. R. C. ; MOREIRA, L. O. ; SANTOS, G. A C. ; MACHADO, J. C. “Quality of Service for Database in the Cloud". In: *2st International Conference on Cloud Computing and Services Science (CLOSER)*, 2012.

1.4 STRUCTURE OF THE THESIS

The next chapters of this thesis are structured as follows:

- Chapter 2 - presents the key concepts involved in this work. It addresses the subjects of Scale-Space Theory, Predictive Analytics, Time-Series Forecasting, ARIMA, Machine Learning and SVM.
- Chapter 3 - analyzes related works.
- Chapter 4 - brings the characteristcs of our solution. The workload, objectives, architecture, implementation details and a study concerning the obtention of good SVM parametrization are presented.
- Chapter 5 - consists in the experiments.
- Chapter 6 - makes the final considerations of this research and outlines the possibilities that may be followed by future works.

CHAPTER 2

THEORETICAL BACKGROUND

This chapter brings the theoretical backgroung related to our work. Here, in the following sections, we present Scale-Space Theory, Predictive Analytics, Time-Series Forecasting, ARIMA, Machine Learning and SVM.

2.1 SCALE-SPACE

As stated in [24], the main idea of creating a multi-scale representation of a signal, as shown in *Figure 2.1*, is by generating a family of one-parameter (scale) derived signals, each of them based on the original one and presenting a decreasing level of detail as the scale increases. As a result, unnecessary features and noise are removed or strongly attenuated at a wider scale so the signal processing may be concentrated over features shown at that scale.

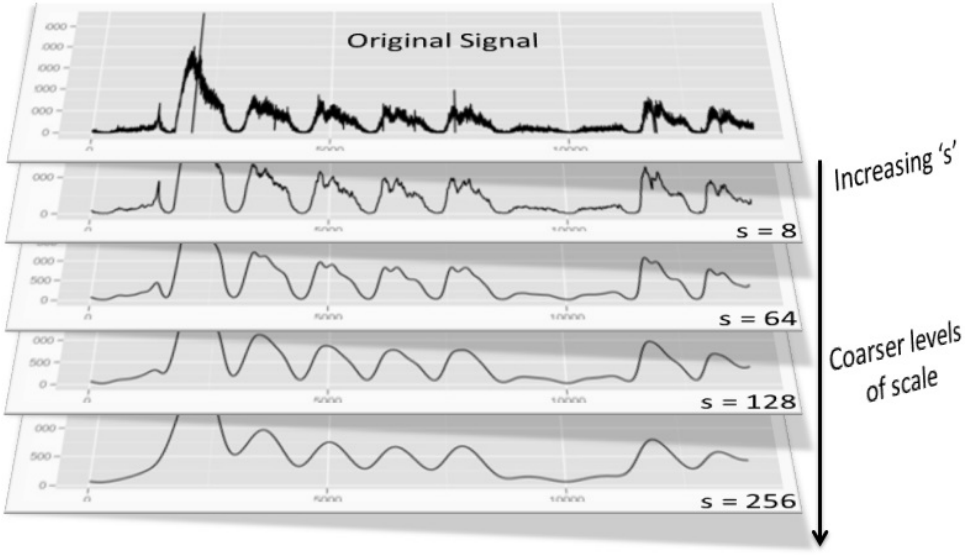


Figure 2.1 A multi-scale representation of a signal.

The procedure, as it is formulated for one-dimensional continuous signals is as follows:

Given a signal $f : \mathbb{R} \to \mathbb{R}$, the scale-space representation $L : \mathbb{R} \times \mathbb{R}_+ \to \mathbb{R}$ is defined such that the representation at "zero scale" is equal to the origial signal

$$L(\cdot; 0) = f(\cdot), \tag{2.1}$$

and the representations at coarser scales are given by convolution of the given signal with Gaussian kernels of successively increasing width

$$L(\cdot; s) = g(\cdot; s) * f. \tag{2.2}$$

In terms of explicit integrals, the result of the convolution operation '$*$' is written

$$L(x, s) = f(x) * g(x, s) = \int_{-\infty}^{+\infty} f(u) \frac{1}{s(2\pi)^{\frac{1}{2}}} e^{\frac{-(x-u)^2}{2s^2}} du \tag{2.3}$$

The Gaussian function is used because it is the unique kernel satisfying the *Scaling Theorem* [25]. It states that Gaussian is the only kernel for which local maxima either remains the same or decrease and local minima either remains the same or increase as the bandwidth of the filter *'s'* is increased, i.e., with increasing scale. The reverse is also verified: if a convolution kernel never introduces additional structure, then it must be gaussian. Therefore, no additional structures are introduced by this process [26]. Moreover, most of the structures in the signal with a characteristic length less than *'s'* are removed after convolving a signal by $g(\cdot; s)$. Hence, the bandwidth of the filter controls the type of high frequency information we want to eliminate from the signal.

Consequently, we can also minimize the effect of data points that, although are not noisy objects or outliers, are still irrelevant or only weakly-relevant [27] to the underlying data analysis. In our context, such datapoints are extrema whose duration might be considered irrelevant to be taken into account by provisioning strategies.

However, as we apply higher values of *'s'*, the interval between the original and the obtained extrema increases. Although the resulting signal does capture the system's behavior, it becomes less useful in expressing the system's real demands, since the observed values will be somewhere below the expected ones, which should be as close as possible to the original value. Such behavior is expected, given that for small values of *'s'* the Gaussian convolution approaches the un-smoothed signal and for large *'s'* the signal's mean [19].

We overcome this problem by using linear interpolation [28], which allows us to restore the amplitudes of the smoothed signal to their original values. This is depicted in *Figure 2.2*.

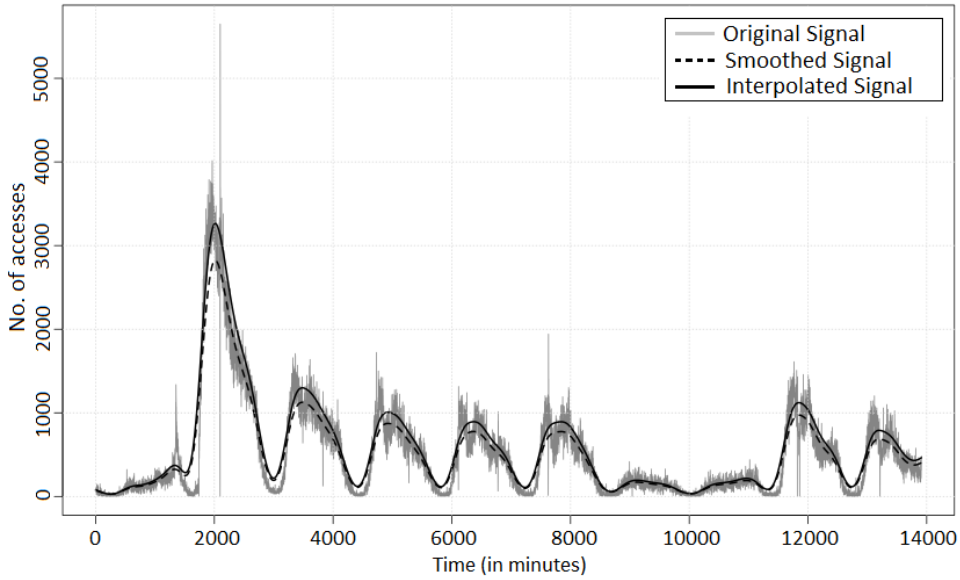


Figure 2.2 Interpolating the signal obtained by a high value of *'s'* in the scale-space algorithm with the original one helps bringing the extrema close to their original amplitude. This is very helpful, since the detection of extrema plays a key role in resource provisioning.

The first step is the generation of an interpolation weight that describes how points in the source signal, the smoothed one, are related to points in the destination signal (the original one). Instead of a constant interpolation weight, in order not to alter the signal structure, interpolation weights are calculated from *equation 2.4*. The second stage is the application of the interpolation formula, *equation 2.5*, to produce the values on the destination signal.

$$w(x_i) = \frac{g(x_i) - min(g_x)}{max(g_x) - min(g_x)} \tag{2.4}$$

$$f(x_i) = w(i) * \frac{1}{2}[max(f_x) + max(g_x)] + [1 - w(i)] * min(g_x) \tag{2.5}$$

where *'f'* is the function that represents the original signal and *'g'*, the smoothed one.

With the application of both scale-space smoothing on the signal that represents

the monitored metrics and interpolation on the obtained smoothed signal we end up with a dataset that is either free of noise or contains an easily treatable amount of it and in which the relevant extrema correspond to their counterparts in the original signal.

2.2 PREDICTIVE ANALYTICS

Predictive analytics involves the use of data mining, mathematical modeling, and statistical analysis to provide actionable predictions based on trends, patterns, relations and correlations in data to help driving the decision-making process. It considerably enriches decision-making by providing intelligent predictions as the basis for the process. Predictive analytics is becoming more mainstream as a result of advanced machine learning capabilities, technology advancements and availability of large volumes of data [29].

The approaches and techniques used to conduct predictive analytics include decision trees, time-series forecasting, neural networks, genetic algorithms, support vector machines, and other mathematical algorithms, to name a few [30] [31].

2.2.1 Time-Series Forecasting

Whatever we observe or measure is bound to be different at differrent points in time [32]. When we collect a set of such measurements with the objective of drawing conclusions from their observation considering the time they were measured, we're dealing with a problem with a temporal aspect. This means that in the input data there is a time-dependency. In other words, an input at one timestep does not contain enough information to get the correct output. The output is thus dependent on a number of input patterns [33]. Thus, time series data has a natural temporal ordering. It differs from typical data mining/machine learning applications where each data point is an independent example of the concept to be learned, and the ordering of data points within a data set does not matter. Examples of time series applications include: capacity planning, inventory replenishment, sales forecasting and future staffing levels [34].

A time series is a set of observations x_t, each one being recorded at a specified time t. A discrete-time series is one in which the set T_0 of times at which observations are made is a discrete set, as is the case for example when observations are made at fixed time intervals. Continuous-time series are obtained when observations are recorded

continuously over some time interval, e.g. when $T_0 = [0, 1]$ [35]. It is a record of a phenomenon irregularly varying over time [36] and, as such, the excitations that are imposed on the system under observation are not under the observer's control, and are, in many cases, unknown to him or her [37].

Time series analysis is the process of using statistical techniques to model and explain a time-dependent series of data points [34]. It accounts for the fact that data points taken over time may have an internal structure, such as autocorrelation, trend or seasonal variation, that should be accounted for [38], and it also provides tools for selecting a model that can be used to forecast future events.

The general problem of time series forecasting can be rephrased as the problem of finding a model able to forecast the future evolution of a time series given its past evolution. Most of the time, the forecasting problem is limited to a short-term time series prediction. In other words, as one tries to model the future evolution of a time series, the usual goal is to be able to perform a onestep ahead prediction. The main reason motivating such approach is reliability of the predicted values. One-step ahead predictions can be reasonably reliable, while the uncertainty on future values increases with the time horizon [39], i.e., during the first step of a multi-step prediction, the predicted value depends entirely on measurement data, and is therefore more likely to be accurate than in subsequent steps, when the predictions depend on previously predicted data points that are by themselves associated with a degree of uncertainty already [37].

Plummer [40] lists some difficulties that may arise when performing time series forecasting:

1. limited quantity of data, which may be the foremost difficulty;
2. noise, i.e., any unwanted anomaly in the data, which can be caused by:
   - errouneous data points;
   - components that obscure the underlying form of the data series.
3. nonstationarity, i.e., data that do not have the same statistical properties, like mean and variance, at each point in time;
4. forecasting technique selection from a multitude of options from statistics to artificial intelligence. One of the simplest techniques is to search data series for similar

past events and use the matches to make a forecast. One of the most complex ones is to train a model on the series and use the model to make a forescast.

2.2.1.1 ARIMA Auto Regressive Integrated Moving-Average (ARIMA) models are a form of regression analysis that use lagged values of the dependent variable and/or random disturbance term as explanatory variables. The approach was first popularized by Box and Jenkins [41], and ARIMA models are often referred to as Box-Jenkins models.

It is generally referred to as an $ARIMA(p, d, q)$ model where:

- p is the number of autoregressive terms,
- d is the number of non-seasonal differences, and
- q is the number of lagged forecast errors in the prediction equation.

To identify the appropriate ARIMA model for a time series, one begins by identifying the order(s) of differencing needed to stationarize the series and remove the gross features of seasonality. Thus, one has to take as many differences of the original series as are needed to reduce it to stationarity. A strong reason for using a stationary data sequence instead of a non-stationary sequence is that non-stationary sequences, usually, are more complex and take more calculations when forecasting is applied to a data series.

When the differencing of the series is done successfully it is proposed that the differenced series can be treated in the same way as a stationary series, which has had no need of differencing.

This leads to a wider family of models which are ARMA models after differencing. As such, ARIMA models are a generalization of ARMA models obtained by introducing the differencing into the model, with I indicating "integrated" and referencing the differencing procedure.

With a stationary series in place ($d = 0$), a basic model can now be identified. After obtaining both the *AR order* p and *MA order* q, we are led to one of three basic models: AR (autoregressive), MA (moving average) and a combined ARMA.

- When both $p \neq 0$ and $q \neq 0$ we have the usual ARMA model, that is $ARIMA(p, 0, q) = ARMA(p, q)$.

- When only $p \neq 0$, we have $ARIMA(p, 0, 0) = AR(p)$.
- Finally, when only $q \neq 0$, $ARIMA(0, 0, q) = MA(q)$.

Next, one has to estimate the coefficients of the model, which consists on finding both φ and θ in the first case, or only φ in the second case and θ in the third one. These models are given by:

$$AR(p) : X_t = C + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t \tag{2.6}$$

$$MA(q) : X_t = \mu + \varepsilon_t + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{2.7}$$

$$ARMA(p, q) : X_t = C + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i} \tag{2.8}$$

where φ_i and θ_i are parameters, the random variable ε is white noise error, C is a constant and μ is the expectation of X_t (often assumed to be 0). In practice, estimation is fairly transparent to the user.

Fianally, the model has to be checked. This step is also called diagnostic checking, or verification [42]. Two important elements of checking are: (1) to ensure that the residuals of the model are random; and (2) to ensure that the estimated parameters are statistically significant. Usually the fitting process is guided by the principal of parsimony, by which the best model is the simplest possible one, i.e., that with the fewest parameters, which adequately describes the data.

2.2.2 Machine Learning

Machine learning is programming computers to optimize a performance criterion using example data or past experience [43]. To be intelligent a system that is in a changing environment should have the ability to learn. If the system can learn and adapt to such changes, the system designer does not need to foresee and provide solutions for all possible situations. Instead, the system should be capable to infer hypotheses from given data or experience, which results in some model that generalizes the data and hopefully draws correct conclusions.

The aim of machine learning is rarely to replicate the training data but the

prediction for new cases, which means to generate the right output for an input instance outside the training set, one for which the correct output is not given in the training set. How well a model trained on the training set predicts the right output for new instances is called generalization [43].

This modeling approach with the ability to learn from experience is very useful for many practical problems since it is often easier to have data than to have good theoretical guesses about the underlying laws governing the systems from which data are generated [44].

2.2.2.1 Support Vector Machines Support Vector Machines (SVM) comprise a new class of supervised learning algorithms, motivated by results of statistical learning theory. Originally developed for pattern recognition, they represent the decision boundary in terms of a typically small subset of all training examples, called the support vectors. In order for this sparseness property to carry over to the case of Support Vectors for Regression (SVR), Vapnik devised the so-called ϵ-insensitive loss function,

$$L_{\varepsilon}\left(f\left(x_i\right), y_i\right) = \begin{cases} 0 & \text{if } \left|y_i - f\left(x_i\right)\right| \leq \varepsilon \\ \left|y_i - f\left(x_i\right)\right| - \varepsilon & \text{otherwise} \end{cases} \tag{2.9}$$

which does not penalize errors below some $\epsilon > 0$, chosen *a priori* [45]. Thus, SVR learns samples based on the ϵ-loss function foundation. It makes an ϵ-tube around the training samples so that the samples within the ϵ-tube are not counted as errors, while samples outside of the ϵ-tube become support vectors (SVs) that will be used for test. Hence, in some sense, support vectors are an ideal subset to be selected [46].

The basic idea of SVR is to map the input data x into a higher dimensional feature space F via a nonlinear mapping ϕ and then a linear regression problem is obtained and solved in this feature space. Considering a data set $D = \{(x_i, y_i)\}_{i=1}^{l}$, with $x \in \mathbb{R}^d$ (d-dimensional input space) and $y \in \mathbb{R}$, the goal is to find a function

$$f(x) = \sum_{i=1}^{l} w_i \phi_i(x) + b \tag{2.10}$$

where $\{\phi_i(x)\}_{i=1}^{l}$ are the features of the inputs, $\{w_i\}_{i=1}^{l}$ and b are coefficients, that has at most ϵ deviation from the actually obtained targets y_i for all the training data, and

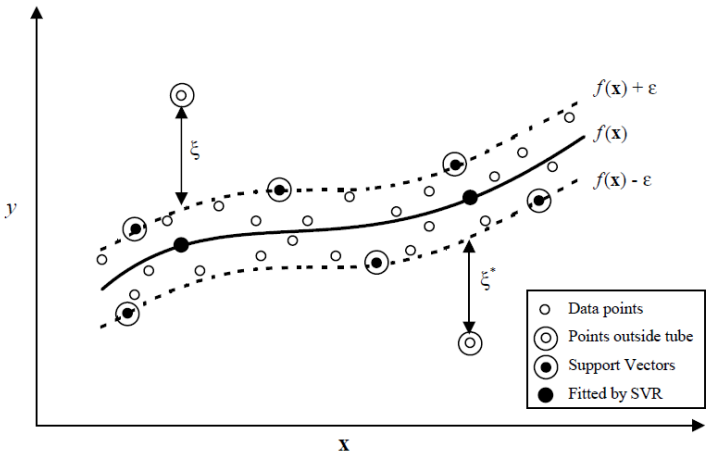


Figure 2.3 A schematic illustration, obtained from [10], of SVR using ϵ-sensitive loss function.

at the same time is as flat as possible, to prevent overfitting. Flatness, om the case of (2.10) means that one seeks a small w. One way to ensure this is to minimize the norm, i.e., $\|w\|^2 = w^T w$. We can write this problem as a convex optimization problem:

$$\min_{w} \frac{1}{2} w^T \cdot w$$

$$s.t. \begin{cases} y_i - \left(w^T \cdot \phi\left(x_i\right) + b\right) & \leq & \varepsilon \\ \left(w^T \cdot \phi\left(x_i\right) + b\right) - y_i & \leq & \varepsilon \end{cases} \tag{2.11}$$

The tacit assumption in (2.11) was that such a function f actually exists that approximates all pairs (x_i, y_i) with ϵ precision, or in other words, that the convex optimization problem is *feasible* [47]. If we want to allow some errors we should introduce

some slack-variables that enlarge the tolerance of the machine:

$$\min_{w} \frac{1}{2} w^T \cdot w + C \sum_{i=1}^{l} \left(\xi_i + \xi_i^*\right)$$

$$s.t. \begin{cases} y_i - \left(w^T \cdot \phi\left(x_i\right) + b\right) & \leq & \varepsilon + \xi_i \\ \left(w^T \cdot \phi\left(x_i\right) + b\right) - y_i & \leq & \varepsilon + \xi_i^* \\ \xi_i^{(*)} & \geq & 0 \qquad i = 1..l \end{cases} \tag{2.12}$$

Everything above ϵ is captured in slack variables $\xi_i^{(*)}$, which are penalized in the objective function via a regularization constant C, chosen *a priori*.

The constrained optimization problem given by (2.12) can be reformulated into dual problem formalism by introducing Lagrange multipliers. This allows us to rewrite the Support Vector algorithm as follows:

$$\max_{\alpha,\alpha^*} \begin{cases} -\frac{1}{2} \sum_{i,j=1}^{l} \left(\alpha_i - \alpha_i^*\right) \left(\alpha_j - \alpha_j^*\right) \phi\left(x_i\right)^T \phi\left(x_j\right) \\ -\varepsilon \sum_{i=1}^{l} \left(\alpha_i + \alpha_i^*\right) + \sum_{i=1}^{l} y_i \left(\alpha_i - \alpha_i^*\right) \end{cases}$$

$$s.t \begin{cases} \sum_{i=1}^{l} \left(\alpha_i - \alpha_i^*\right) & = & 0 \\ \alpha_i^{(*)} & \in & [0, C] \end{cases} \tag{2.13}$$

Given the solution of equation (2.13), the regression function (2.10) can be written as:

$$f\left(x\right) = \sum_{i=1}^{l} \left(\alpha_i - \alpha_i^*\right) \phi\left(x_i\right)^T \phi\left(x_j\right) + b \tag{2.14}$$

Although simply preprocessing the training patterns x_i by a map $\phi : \mathbb{R}^d \to F$ into some feature space F, and then applying the standard SVR algorithm could be achieved,

this approach is not feasible at all and, thus, a computationally cheaper way has to be used, the so-called kernel trick:

$$\max_{\alpha,\alpha^*} \begin{cases} -\frac{1}{2}\sum_{i,j=1}^{l}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)k\left(x_i,x_j\right) \\ \\ -\varepsilon\sum_{i=1}^{l}\left(\alpha_i+\alpha_i^*\right)+\sum_{i=1}^{l}y_i\left(\alpha_i-\alpha_i^*\right) \end{cases} \tag{2.15}$$

$$s.t \begin{cases} \sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right) & = & 0 \\ \\ \alpha_i^{(*)} & \in & [0,C] \end{cases}$$

and equation (2.14), the regression function, becomes equal to:

$$f\left(x\right)=\sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right)k\left(x_i,x\right)+b \tag{2.16}$$

where $k\left(x_i,x_j\right)=\phi\left(x_i\right)^T\phi\left(x_j\right)$ is called the kernel function. An interesting fact is that for many choices of the set $\left\{\phi_i\left(x\right)\right\}_{i=1}^{l}$, including infinite dimensional sets, the form of k is is analytically known and very simple, and the features ϕ_i never need to be computed in practice because the algorithm relies only on computation of scalar products in the feature space [48]. Any function that satisfies Mercer's theorem can be used as a kernel function [49]. Though new kernels are being proposed by researchers, the typical examples of the kernel function are as follows:

$$\begin{array}{lll} \text{Linear:} & k\left(x_i,x_j\right)= & x_i^Tx_j \\ \\ \text{Polynomial:} & k\left(x_i,x_j\right)= & \left(\gamma x_i^Tx_j+r\right)^d,\gamma>0 \\ \\ \text{Radial Basis Function (RBF):} & k\left(x_i,x_j\right)= & \exp\left(-\gamma\left\|x_i-x_j\right\|^2\right),\gamma>0 \\ \\ \text{Sigmoid:} & k\left(x_i,x_j\right)= & \tanh\left(\gamma x_i^Tx_j+r\right) \end{array} \tag{2.17}$$

Here, γ, r and d are kernel parameters. They should be carefully chosen as it implicitly defines the structure of the high dimensional feature space $\phi(x)$ and thus controls the complexity of the final solution [50].

Thus, the kernel function transforms the nonlinear input space into a high dimensional feature space in which the solution of the problem can be represented as being a straight linear problem [51]. Note that, in the nonlinear setting, the optimization problem corresponds to finding the flattest function in feature space, not in input space.

Note that the parameter ϵ can be useful if the desired accuracy of the approximation can be specified beforehand. In some cases, however, we want the estimate to be as accurate as possible without having to commit ourselves to a specifc level of accuracy *a priori* [45]. In their work, they describe a modification of the ϵ-SVR algorithm, called ν-SVR, which automatically minimizes ϵ.

The ν-SVR optimization problem is given by:

$$\max_{\alpha,\alpha^*} -\frac{1}{2}\sum_{i,j=1}^{l}\left(\alpha_i-\alpha_i^*\right)\left(\alpha_j-\alpha_j^*\right)k\left(x_i,x_j\right)+\sum_{i=1}^{l}y_i\left(\alpha_i-\alpha_i^*\right)$$

$$s.t\begin{cases}\sum_{i=1}^{l}\left(\alpha_i-\alpha_i^*\right) & = & 0\\ \alpha_i^{(*)} & \in & [0,C]\\ \sum_{i,j=1}^{l}\left(\alpha_i-\alpha_i^*\right) & \leq & C\cdot\nu\end{cases} \tag{2.18}$$

2.3 ERROR METRICS

In order to evaluate the accuracy of our forecastings, we adopt three metrics, where y_i is the actual output, $\hat{y}_i$ is the predicted output and n is the number of observations in the dataset for which the prediction is made:

1. Root Mean Squared Error (RMSE), which is defined as follows:

$$RMSE=\sqrt{\frac{\sum_{i=1}^{n}\left(\hat{y}_i-y_i\right)^2}{n}} \tag{2.19}$$

2. Mean Absolute Percentage Error (MAPE), defined by the following formula:

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|\hat{y}_i - y_i|}{y_i} \tag{2.20}$$

3. PRED(25), given by:

$$PRED(25) = \frac{\text{Number of observations with relative error} \leq 25\%}{\text{Number of observations}} \tag{2.21}$$

Where the relative error is given by:

$$RE = \frac{|y - \hat{y}|}{y} \tag{2.22}$$

The following observations are relevant, as pointed out by Islam et al. [7] and Nau [52]:

- a lower value of $RMSE$ indicates a more effective prediction scheme;
- a small $MAPE$ value implies a better fit of the prediction model, thus indicating superior prediction accuracy;
- $PRED(25)$ indicates the percentage of observations whose prediction accuracy falls within 25% of the actual value. A value closer to 1 indicates a better fit of the prediction model;
- $RMSE$ is more sensitive than other measures to the occasional large error: the squaring process gives disproportionate weight to very large errors. If an occasional large error is not a problem in decision situation, then MAPE may be a more relevant criterion. In many cases these statistics will vary in unison, i.e., the model that is best on one of them will also be better on the others. Actually, if one model is best on one measure and another is best on another measure, they are probably pretty similar in terms of their average errors. In such cases one probably should give more weight to some of the other criteria for comparing models;
- it makes no sense to say 'The model is good (bad) because the RMSE error is less (greater) than x', unless you are referring to a specific degree of accuracy that is relevant to one's forecasting application;

- if one model's RMSE is 30% lower than another's, that is probably very significant. If it is 10% lower, that is probably somewhat significant. If it is only 2% better, that is probably not significant. These distinctions are especially important when we are trading off model complexity against the error measures.

These points are the reason why we chose three different metrics. That will give us more solid grounds to compare our results.

2.4 CONCLUSION

In this chapter we presented the theories behind our research work. Scale-space theory constitutes the core of it. We also covered time-series forecasting and the theory behind ARIMA, one of the prediction approaches we adopt. Then we briefly mentioned machine learning and described SVM theory, the other forecasting method we use in our work. Besides the good results obtained in the literature for different scenarios, these forecasting methods are representatives from two different paradigms: statistical time-series analysis and machine learning. To finish, we remark on the metrics we employ to measure the accuracy of our predictions.

CHAPTER 3

WORKLOAD ANALYSIS

Among the works that propose solutions to meet QoS requirements based on workload analysis, we may classify them in the two main procedures to cope with this task, the reactive and proactive:

- in the former, they monitor a signal to detect unwanted situations and reactively act in order to minimize as much as possible the effect of such events;
- in the latter they perform history-based load predictions, which aim to foresee the workload behavior based on its observation and anticipate actions in an attempt to avoid unwanted scenarios. Some, in an attempt to minimize even more the negative effects of unexpected surges in workloads, propose solutions that make use of both procedures.

Next we present works that make use of these techniques and are directly related to our work. Thus, although we mention all frameworks' capabilities of related works, we give special attention to the modules that direclty deal with workload preprocessing and prediction.

3.1 PROACTIVE SOLUTIONS

3.1.1 Prediction of Cloud Data Center Networks Loads Using Stochastic and Neural Models [1]

In [1] it is presented a prediction model framework that uses auto-regressive linear prediction model with dithering of the observation sequence to smooth out numerical anomalies and neural network prediction methods to forecast the future load demand profiles. Performance of the methods against two sets of data at multiple look-ahead times is also

presented. The results of the simulation showed that the Linear Predictor modeled the reference data more closely than did the Neural Network, at all prediction intervals.

Sample network traffic data was used to train then simulate the forecasting of network loads for each of the two models. The data used represents URL resource requests of a WWW server at NASA and WWW server at EPA, and the incoming requests were time stamped to the second. The values for the look-ahead intervals used in the simulations were 1s, 5s, 10s, 15s, 20s, 30s, 60s and 90s. For the neural network, the first one-thousand (1000) samples were separated from the data and used to train the network. The Linear Predictor uses a data window of 65 samples, which allowed training to occur with a very small number of the available ones. Both the Mean Squared Error (MSE) and the Root Mean Squared Error (RMSE) were calculated for the predicted results.

According to the authors, both models can effectively predict future network loads, with the Linear Predictor providing the most accurate results. Also they observe an inverse linear relationship between the ability to forecast network loads and the distance into the future the load is being forecasted.

3.1.2 PRESS [2]

The goal in [2] is to develop a light-weight elastic resource allocation scheme while not requiring advance application profiling, model calibration, or deep understanding of the application. It continuously tracks the dynamic resource requirements of applications in an unobtrusive way and predicts resource demands in the near future using two complementary techniques to do so.

For workloads with repeating patterns, PRESS derives a signature for the pattern of historic resource usage, and uses that signature in its prediction. To avoid making assumptions about the length of the repeating pattern, PRESS uses signal processing techniques, namely Fast Fourier Transform (FFT), to discover the signature, or to decide that one does not exist. Since workload signatures may vary over time, this calculation is performed anew each time a prediction is made. For applications without repeating patterns, PRESS uses a discrete-time Markov chain with a finite number of states to build a short-term prediction of future metric values. The resource prediction models are repeatedly updated when resource consumption patterns change.

Evaluation in the PRESS system was conducted using the RUBiS online auction

benchmark (PHP version). Two real-world web traces, per-minute average rates observed in the ClarkNet web server trace and per-minute average rates observed in the web access logs from the World Cup 98 official web site, were used to modulate the request rate of our synthetic RUBiS benchmark. The prediction algorithms were also evaluated using a small sample of real application workload trace data of about 6 hours from a Google cluster, which led them to use shorter training windows. For simplicity of exposition, authors focus on CPU usage, although it is claimed that PRESS is capable of managing memory, I/O, and network usage, too. By default, PRESS makes a prediction every minute, and uses the prediction to scale the resources allocated to the application for the next minute.

Authors state that the prediction algorithms used by PRESS are indifferent to the sampling rate, as long as the input window size is large enough to encompass the most important patterns to track, and there aren't too many samples to analyze. It has been noticed that the further into the future, the weaker the correlation between the model and the actual demand. But, since PRESS only needs to make predictions for the near future, this is not an issue in practice.

3.1.3 RPPS [3]

This paper presents the design, implementation and evaluation of RPPS (Cloud Resource Prediction and Provisioning Scheme), a framework that automatically predict future demand and perform proactive resource provisioning for cloud applications. RPPS employs a load prediction algorithm based on ARIMA model to gradually adapt resource according to the future demand. The resource usage time series are fed into the load predictor model to predict a short-term resource demands. According to authors, since for resource provisioning focus should be on short-term load prediction, they conduct their studies over a period from half an hour to several hours. It also combines both coarse-grained and fine-grained resource scaling under different situations, and adopts a VM complementary migration strategy.

Thus, RPPS architecture is composed of three main parts: The predictor is responsible for forecasting incoming load. History workload is obtained from the monitor. After running the predictor, the controller has got the future demand which can help resource provisioning ahead by managing the state of the cloud nodes. It reacts to new requests or changes in workload by provisioning fine-grained CPU, memory resources or

coarse-grained new VMs and allocating physical resources. The allocator is responsible for resource allocation based on prediction results and real history data.

In the experiments, they use two types of workloads: own-collected workload by typical applications and request rate from a real data center. In the first case, the trace used is part of the several months of data collection. It shows a consistent pattern of load shifts by day. According to authors, a qualitatively similar pattern appears in other internet service traces, with daily peak-to-trough and significant seasonal variations, such as Messenger. In the second, to illustrate a more real load fluctuation, accesses to the Beijing Unicom site in a typical week is analyzed. Experiments are conducted using the CPU usage.

During the experiments, authors point out that it is hard to predict the peak points. They found that RPPS is weak in sharp transient spikes. According to them, if they solve this problem with the pre-reserved resource, they may cause another problem: they should take back these resources in a short time because the spikes not last long. The average error is used in the whole time to evaluate the performance. The experiment results show RPPS works well in the real data center trace.

3.1.4 Marlowe [4]

Marlowe is an automated server provisioning system that aims to meet workload demand while minimizing energy consumption in data centers. It integrates load prediction, system and load management functions, and cost-benefit analysis. Servers are automatically provisioned to meet workload demand while minimizing the energy cost and the reliability cost due to duty cycling.

The framework design has two key components: a load predictor for proactive provisioning and an optimizer framework that uses workload prediction to manage transitions into and out of power states to meet demand while saving energy and reducing the reliability cost.

First, the input workload is characterized and then they employ forecast-based technique based on regression analysis to predict load in the near future. They quantize time into discrete time steps, which may be a few seconds to several minutes, depending on the load variation and the transition delays associated with changing server power states. Authors derive a simple and intuitive linear prediction scheme because they claim

it is easy to implement, has a sound theoretical basis, and works extremely well on their workload traces. Nonetheless, they emphasize the discussion applies no matter which prediction scheme is used.

Second, to optimize state transitions to meet demand while saving energy, they build a Markov state diagram of power state transition for each server by measuring the power draw in each power state and the transition latencies in and out of these states.

Marlowe was evaluated on three workloads: Windows Azure, Live Messenger, and a shared computing cluster (SCC) comprising about 1000 servers. The Live Messenger trace contains login rate and number of connections per server which authors scaled to 25M connections across 300 servers over two weeks; a prior study presented many aspects of load and user behavior for this system. The Azure and SCC traces contain cluster utilization data collected from about 500 servers over three months and from 1008 servers over a week, respectively. They analyze load patterns, auto-correlations, and cluster utilization for these traces.

Results show that Marlowe adapts to offered load and can quickly identify the optimal assignment of servers to power schemes thereby significantly saving energy while meeting workload demand. They compute the standard deviation of the relative error ($|PredictedActual|/Actual$) as goodness of fit metric.

3.1.5 Predictive Data Grouping and Placement for Cloud-based Elastic Server Infrastructures [5]

This paper aims to explore the benefits of leveraging access patterns and workload history in predicting the load and organizing content in an Internet service infrastructure in order to achieve high resource utilization, locality and elasticity. Authors apply autoregressive models for history-based load prediction. This model, according to them, is able to predict growth trends and seasonal patterns and allows for proactive resource provisioning.

First, they propose a forecasting model based on time series, which is used to predict workload based on access history. In order to predict the load in the system at time t, a multiplicative seasonal autoregressive moving average (ARMA) model is adopted. Second, two complementary data grouping and data placement algorithms targeting to increase the locality, while preserving a high server utilization are presented. Third,

they propose algorithms for scaling the system up and down, depending on expected workloads.

They analyze a dynamic workload of the on-line music portal LastFM. Data was crawled through LastFM API by using a distributed crawler deployed in cluster over 20 machines. They traversed the friendship graph in a breadth first search manner and extracted the profiles of a set of 250,000 users including the listened artists, daily, for the period between January 1st to May 22nd, 2009 (142 days). According to them, a limitation of their trace is the fact that they can reconstruct only the aggregate daily load of the system, given that LastFM does not make available the access history at a granularity lower than a day. Therefore, the trace does not contain load variation at minute and hour granularities. The first 14 days are used for training, then no forecasting is done until day 15.

Experiments demonstrate the suitability of applying autoregressive models to global load forecasting for a 142 day trace from LastFM, a music on-line service. Errors metrics used were relative error and normalized root mean square deviation. In addition, they conclude that the utilization of a prediction algorithm allows to scale up and down the number of required servers.

3.1.6 SmartSLA [6]

SmartSLA is a cost aware resource management system that consists of two main components: the system modeling module and the resource allocation decision module. The system modeling module uses machine learning techniques to learn a model that describes the potential profit margins for each client under different resource allocations. Based on the learned model, the resource allocation decision module dynamically adjusts the resource allocations in order to optimize expected profits.

A series of mature machine learning techniques are investigated. They start with a standard simple linear regression model. However, the accuracy turns out to be unsatisfactory, which they show to be due to the nonlinear relationship between the cost and some of the resources. Then, they show that the prediction accuracy is improved by the regression tree model, which takes the nonlinearity into account. To further improve the prediction accuracy, they use a boosting approach, called additive regression method, using regression tree as the underlining weak learner. They use those mature regression

methods in WEKA package, because as they state, the focus of the work is how to apply machine learning techniques in virtualized resource management, not to invent new models.

They develop workload generators to emulate clients where new queries are generated independently to the completion of previous queries. The arrival rate of queries follows a Poisson distribution with the rate set in each test. They choose Poisson distribution because it is widely used to model independent arrival requests to a website. They assume that the system is monitored continuously and the resource allocations can be changed periodically in each time interval. In all experiments, they set the time interval to be 3 minutes. They state this value was chosen because too short intervals cannot capture the randomness of query types while too long intervals make SmartSLA less responsive.

Authors focus mainly on CPU-bound queries and CPU and memory resources. Queries are CPU-bound when the database can reside in the memory. Therefore the whole database size is set to about 280MB to make sure it can fit in the memory. Also, in order to obtain a high-level overview of the relationship between system resources and database system performance, they conduct several experiments and statistical analysis on how the parameters, such as CPU share, memory size, client workload, and replica number, affect the system performance.

SmartSLA is evaluated by using the TPC-W benchmark with workload characteristics derived from real-life systems. The performance results indicate that SmartSLA can successfully compute predictive models under different hardware resource allocations, such as CPU and memory, as well as database specific resources, such as the number of replicas in the database systems.

3.1.7 Empirical Prediction Models for Adaptive Resource Provisioning in the Cloud [7]

In this paper, authors build up prediction models which are able to forecast sudden increase in resource requirement in advance in order to facilitate dynamic and proactive resource management, scheduling and capacity planning. The framework adopts statistical learning algorithms and sliding-window mechanism which according to them are simple yet effective. The proposed algorithms are Error Correction Neural Network (ECNN)

and Linear Regression. It is mentioned however that it is also possible to accommodate other learning methods.

Authors state that the current scope of their work is bound to the development of performance prediction model and its statistical validation only. Thus, they strive to focus on the research problem of developing resource prediction models for facilitating proactive scaling in the cloud so that hosted applications are able to withstand the variation in workload with least drop in performance and availability. As they state, their research problem actually relates to time-series analysis.

To obtain the historical data, they customized the TPC-W client implementation to generate web requests in a linear fashion as it closely resembles the ramp-up phase of flash crowd web traffic. That data was then used for training and testing of the forecasting models. The efectiveness of the prediction framework was validated based on a number of metrics: Mean Absolute Percentage Error (MAPE), PRED(25), Root Mean Squared Error (RMSE) and R^2 Prediction Accuracy.

Thus, experimental setup was divided into two steps: historical data collection in the cloud and training of the prediction system with the historical data. They selected the prediction interval as 12 minutes because, as they state, the setup time of virtual machine instances in the cloud is typically around 5-15 minutes; so the scaling system needs to request for new virtual instances 12-15 minutes ahead of time in order to meet up the surge in resource requirement in the cloud.

According to their tests, the framework is not only able to make accurate projections but also skilled enough to forecast resource demand ahead of the VM instance setup time. The use of an input window has a positive effect on the accuracy of the prediction models and Neural Network based models with an optimal window size yield superior prediction accuracy than Linear Regression ones. However, since the training of Neural Network models take signicant time, the frequency of the training should be determined based on the underlying resource usage behavior of the application in the cloud.

3.2 REACTIVE SOLUTIONS

3.2.1 SLA-Based and Consumer-Centric Dynamic Provisioning for Cloud Databases [8]

In [8], a framework is designed in order to facilitate adaptive and dynamic provisioning of the database tier for cloud-hosted software applications. Authors present an end-to-end framework that enables the software applications to declaratively define the SLA of the application in terms of goals which are subjected to a number of constraints that are specific on its database transactions. It also enables the software provider to declaratively define a set of application-specific rules (action rules) where the admission control of the database tier needs to take corresponding actions in order to meet the expected system performance or to reduce the cost of the allocated cloud resources when they are not efficiently utilized.

The framework continuously monitors the database workload, tracks the satisfaction of the application-defined SLA, evaluates the condition of the action rules and takes the necessary actions when required. It is database platform-agnostic and relies on virtualization-based database replication mechanism. While this paper focuses on the SLA metric of transaction execution time, authors claim that other consumer metrics can be implemented and integrated in the same manner such as freshness of replicated data.

An open source synthetic workload generator which is used to compose their application workload. Experiments were conducted with 4 different rules for achieving elasticity and dynamic provisioning for the database tier in the cloud. Two rules are defined based on the average CPU utilization of allocated virtual machines for the database server as follows: Scale out the database tier (add one more replica) when the average CPU utilization of the virtual machines exceeds of 75% for (R1) and 85% (R2) for a continuous period of 5 minutes. Two other rules are defined based on the percentage of the SLA satisfaction of the workload transactions (the SLA values of the different transactions are defined as specified in the Cloudstone benchmark) as follows: Scale out the database tier when the percentage of SLA satisfaction is less than 97% for (R3) and 90% (R4) for a continuous period of 5 minutes.

Their evaluation metrics are the overall percentage of SLA satisfaction and the number of provisioned database replicas during the experimental time. Results show that the approach provides the cloud consumers with adequate declarative and more flexible

mechanism for controlling the SLA of their applications than relying on monitoring the utilization of the allocated cloud computing resources.

3.2.2 RepliC [9]

This paper presents RepliC, an approach to database replication in the cloud with quality of service, elasticity, and support to multi-tenancy. In RepliC, elasticity adjusts the system's capacity at runtime by adding and removing replicas without service interruption in order to handle the workload variation.

The target workload is given by the rate of transactions. The SLA (i.e. response time) gives the objectives of each database's services. A penalty is applied to the transactions that exceed the SLA value. The main problem is how to ensure the SLA for a set of database services according to the current workload while using resources efficiently with small SLA violations. For their monitoring strategy, the collect process is performed six times with an interval of 10 seconds. For each collect process, RepliC calculates the median and standard deviation. Two medians with lower deviation are selected as the final values to be stored. To these values, it is applied an exponentially weighted moving average. If the monitored value is not in accordance with the defined SLA, more resources are added. On the other hand, if the SLA is satisfied over time, resources are removed.

The addition and removal of replicas is quickly performed without service interruption in order to handle the workload variation. Experiments demonstrate that their approach guarantees quality of service with small SLA violations.

3.3 DISCUSSION AND COMPARISON

The approach presented in this thesis, as in [7], has its scope bounded to time series analysis and to the use of prediction models and their statistical validation to speculate the future surge in workload. Contrary to [6][2][4][3][5][8][9], which perform some kind of resource allocation, we do not focus on the development of a resource allocation module. Integrating our approach to framewoks that perform this task remains a future work.

At a preprocessing step, the application of scale-space theory to our signal allows us to obtain more accuracy in our predictions at different time scales. Thus, contrary to

the presented works on proactive strategy, we are not limited to predictions at a fixed time stamp. Also, we analyse the use of this mehod so that we can guarantee it will not distort the original signal by removing important elements to our goals.

Prevost et al. [1] claims to dither the observation sequence to smooth out numerical anomalies in order to apply the linear model. No analysis, however, is performed with means to evaluate whether this smoothing process eliminates important characteristics from the original signal. Since, as stated in this work, the linear predictor proves more accurate results than neural networks, it remains unclear though if this result is due to dithering or if it might be because of a linear input-output relationship. It is known that deliberate smoothing might create artificially high correlations between any two smoothed series, and, also, an important consideration in developing linear regression models is that the relationship one is trying to model might not in fact be linear. Even if the relationship is not linear, one can still mechanically apply the formulas to calculate values for the regression-model parameters that will minimize the sum of the square of the residuals. However, the resulting model will be wrong in the sense that it will give poor predictions for output values. Furthermore, applying a linear model to a nonlinear system will give a misleading impression about the system's overall behavior. Consequently, it is very important to verify that the inputs and outputs appear to be linearly related [53].

Authors in [4] point out that in order for their prediction approach to be robust to noise, thay discard small singular values obtained by the singular value decomposition method to solve the least-squares equations from their linear prediction mehod. However, no discussion is done as to the type of information being eliminated.

Gong et al. [2] first employs Fast Fourier Transform (FFT) to identify repeating patterns called signatures that are used for its predictions. Guenter et al. [4] also analyze load patterns, auto-correlations, and cluster utilization for their traces. Although we do not analyze load patterns, we now understand its importance and load characterization is part of future work.

Xiong et al. [6] conducts several experiments and statistical analysis on how the parameters affect the system performance. They conclude that some features such as memory size affects the system performance in a nonlinear manner. In their case, however, they observe that when the arrival rate (number of queries per second) of the analysed workload becomes higher, the system is under more stress and there is a higher chance

for a query to have queuing delay and so its average cost grows. Therefore, in order to avoid frequent saturation, they only vary the rate from 1 to 12 queries per second in the experiments. Besides, the whole database size is set to about 280MB to make sure it can fit in the memory. However, systems with highly dynamic workloads may easily overcome these restrictions, and any method applied to deal with such workloads must be ready to cope with this task. In our approach, we use the visit rate observed in a real-world trace to modulate the request rate of the TPC-C benchmark. We make no restrictions about the rates and the database used in our experiments easily reach the order of gigabytes, thus approaching our strategy to real scenarios of systems with dynamic behavior.

Prevost et al. [1] considers look-ahead intervals that vary from 1s to 90s, and Gong et al. [2] performs predictions for the next minute. Considering that for a fixed observation window size the hability to forecast tends to diminish as the ammount of data one is trying to forecast is increased, this might lead to behaved situations that might not reflect a method's limitations under stressed scenarios. While the interval used by Islam et al. [7], 12 minutes, is acceptable to the provisioning of stateless systems, stateful systems such as DBMSs, on the other hand, require higher look-aheads intervals to deal with the requirement of maintaining consistency of the database that they manage. Guenter et al. [4] expects load prediction to work reasonably well for intervals of tens of seconds to a few minutes. Authors in [3] conduct their studies over a period from half an hour to several hours, but no details as to the size of the prediction window is given. In our work, we test the methods' accuracy varying the prediction window from 10 to 24 minutes in the case of short-range predictions and 700 minutes when considering long-range forecastings. As the history of observed data increases, we then become able to increase the prediction window to obtain a higher prediction margin in different scales.

As for the size of the observation window, in [2] it is claimed that the input window size must be large enough to encompass the most important patterns to track. For the RUBiS trace they use 4320 samples, while for the trace from a Google cluster, which is shorter, they use training windows of 512 samples. That might bring the risk to over-speacialization. Fang et al. [3] and Islam et al. [7], the latter in its solution without sliding-windows, do not explicitly state what the training window size was. It is only mentioned that observations from a specified trace have been used to train the model. Authors, in [1], use training windows of 1000 samples to train the network and that the Linear Predictor uses a data window of 65 samples. Xiong at al. [6] only states that each experiment runs over 9000 seconds and that it takes about 76 seconds for the

learner to build the boosting model which can be done offline. Although Guenter et al. [4] quantizes time into discrete time steps, which may be a few seconds to several minutes, the relation between observation window and prediction window is also not clear. In our work, we empirically tested the methods habillity to forecast for a given ammount of data observed. By analysing this relationship, we are able to determine how far we can take the methods we are working with to avoid inefficiency by considering too small windows or, given its application context, to avoid detaining ourselves only to take into account higher windows sizes, which, to a certain point, tend to provide better results.

Fang et al. [3] also points out that in the trace they consider, there are many load spikes or flash crowds and that their solution is weak in sharp transient spikes. As authors state, solving this problem with the pre-reserved resource, they may cause another problem: the necessity of taking back these resources in a short time because the spikes do not last long. We believe our solution is able to easily handle such scenarios, with the use of the scale-space technique, by removing irrelevant information from the signal we are analysing in a controlled way.

In *Table 3.1*, we evidence important aspects related to each of the related works.

3.4 CONCLUSION

In this chapter, we presented some approaches found in the literature which relate to proactive and reactive techniques. We described their characteristics and provided a discussion on how certain aspects are taken into account in these works in comparison to our research. Although many of the listed works already include a provisioning module, we gave special attention, in these cases, to the ones related to workload analysis and forecasting.

Work \ Feature	Forecasting	Efficacy Metric	Forecasting Granularity	Observed Metric	Env.
[1]	Neural Network; Autoregressive Linear Model	MSE; RMSE	Seconds	URL Resource Requests	Cloud
[2]	Signature-Based Approach; Discrete-Time Markov Chain	Over- and Underestimation Errors	Minutes	CPU Usage	Cloud
[3]	ARIMA	Average Error	Minutes	CPU Usage	Cloud
[4]	Linear Prediction Scheme	Standard Deviation of the Relative Error	Minutes	Login Rate and Number of Connections; Cluster Utilization	Web; Cloud; Parallel
[5]	ARMA	Relative Error; Normalized Root Mean Square Deviation	Days	Daily Access	Web
[6]	Linear Regression Model; Regression Tree Model; Additive Regression Method	RMSE; Relative Absolute Error	Minutes	CPU and Memory	Cloud (DB focused)
[7]	Linear Regression; Neural Network	MAPE; PRED(25); RMSE; R2	Minutes	CPU Usage	Cloud
[8]	Does Not Apply	SLA Satisfaction; Number of Provisioned Database Replicas	Does Not Apply	CPU Usage; Percentage of SLA Satisfaction	Cloud
[9]	Does Not Apply	SLA Violations	Does Not Apply	Response Time	Cloud

Table 3.1

CHAPTER 4

SCALE-SPACE BASED WORKLOAD ANALYSIS AND PREDICTION

This chapter describes the objectives we intend to achieve with *S-SWAP*, our *Scale-Space based Workload Analysis and Prediction* framework. Before doing that, we describe the characteristics of the workload used in our analysis. Then we explain the core of *S-SWAP* and the techniques we adopt. We also present the architecture we developed and implementation details. Given that we are not able to determine SVM parameters automatically, a study on their influence in determining SVR generalization capabilities is also conducted in the last section.

4.1 WORKLOAD

When one thinks about provisioning, what is most important are the metrics which are chosen to be measured, and the metrics to which particular attention is given, since unnecessary detail means wasted time and lacking the proper detail can be fatal.

To illustrate the latter situation, consider the scenario described by [54], where, during an experiment, the CPU is only loaded till around 30%. After this point, as load increases, the response time shoots up even though the CPU is underloaded. The memory and bandwidth also remain much below its capacity. Although, at first, one might think strange behavior is seen in this experiment, it is evident that there is some other bottleneck in the system which causes the response time to shoot up. The model used understandably cannot predict this behavior since the “invisible bottleneck" is not modeled. Thus it estimates the CPU utilization to increase with load while, in reality, it saturates around 30%. Finding the root cause of this bottleneck is hard since it might require investigating immense amount of code and analyzing various scenarios. In a real-world scenario it could also be caused by third-party libraries, the code for which may not be available. So, it is important to have in mind that the term saturation should not be inferred to indicate that a device (e.g., CPU, disk) reached 100 percent utilization,

because service level agreements can be violated well before the 100 percent utilization mark.

So, computer systems often have bottlenecks. The key to effective capacity planning is identifying them, and making decisions based on that information. It bears mentioning that picking the right metric to follow can be difficult, since not all bottlenecks are obvious, and the chosen metric can change as the architecture and hardware limitations vary.

For this reason, to measure higher-level metrics specific to the application should also be considered. As such, to our purposes we will study a common application-specific workload. Since our intention is to build a module that receives metrics to apply our techniques with means to turn them more effective to provisioning strategies, we will not focus on these monitoring challenges. So, although this type of workload presents strong correlation with the demand for resources in such environments, by analysing this type of metric we are not able to detect what resources are bottlenecks, although they are useful to infer QoS. In future work, we intend to analyse the impact of system resource metrics, like CPU, memory and virtual machine state, and also other application specific ones.

We gathered a workload trace from our institution's academic management system during the enrollment period of the undergraduate students for the first semester of 2012. It happened from 01/22/2012 to 01/31/2012. The number of accesses during this period is presented in intervals of one minute, totalizing ten days of collected data. This can be seen in *Figure 4.1*

The reason we chose this workload to our tests is that it resembles the type of workload seen in typical dynamic applications [18][55]. They are usually characterized by dynamically varying workloads that contain both long-term and short-term fluctuations. The first type, from which one can normally extract a pattern or pinpoint factors that influence it, includes variations that can be affected by the season, time of the day, or day of the week, to name a few. These are usually predictable, but may also trick a forecast method due to external influences like last hour schedule modifications, delays or changes to certain functionalities, which happen in our system from time to time. The latter type is normally due to flash crowds and can be very unpredictable. In *Figure 4.1* we can see both these types of fluctuations in (a) and (b) respectively.

Our workload allows us to perform analysis in different time scales (hours, days or weeks for example). Besides, although not linearly, other metrics' behaviors are related

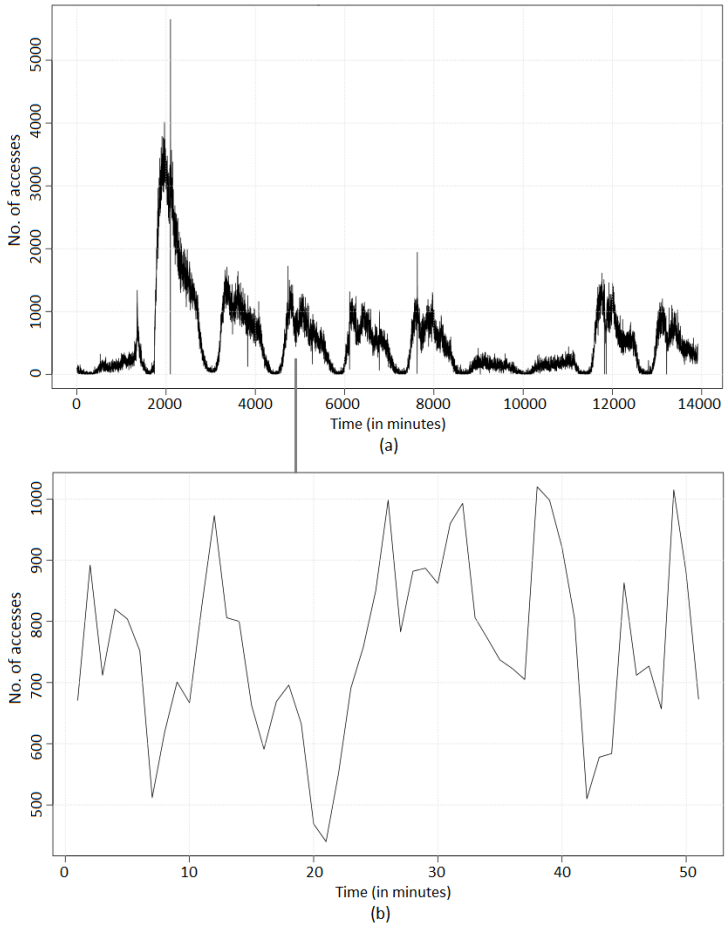


Figure 4.1 Workload trace during the enrollment period is shown in (a), which happened from 01/22/2012 to 01/31/2012. In (b), a workload trace of 50min, from the pointed region in (a), is presented.

to it, and we believe that if we are able to improve our analysis in this scenario, we might also take more informed decisions in future multivariate studies. Still, we highlight that any metric or set of metrics may be used, just like any reactive or proactive approaches.

4.2 OBJECTIVE

Both reactive and proactive techniques might be sensitive to the highly pushful nature observed in dynamic applications' workloads.

Dynamic provisioning involves increasing or decreasing the ammount of resources allocated to an application in response to workload changes. Thus, it typically involves two problems: when to trigger a capacity increase (or decrease), and how to achieve

the desired capacity addition or reduction [18]. Our focus in this thesis aims to help answering the 'when' problem.

As pointed out in [56], if not addressed carefully, dynamic resource provisioning may induce multiple concurrent provisioning operations that are actually not necessary, but only triggered because the system is in a temporarily instable state. This would, for instance, result in first adding more resources than strictly necessary, and then later on removing the unnecessary resources. As a result, these oscillations would hurt the overall service performance. This problem is made harder by the fact that, in multi-tier applications, two concurring provisioning operations do not always act on the same parts of it, but to different ones. For example, in a three-tier Internet service consisting of a front-end tier of replicated web servers, a middle tier of application servers, and a back-end tier of replicated database servers; the back-end tier may become the bottleneck, which results in an under-load of the front-end tier (which simply waits for responses from the back-end tier). In such a situation, a provisioning operation may be triggered on the set of replicas of the back-end tier (because of its over-load) while an unprovisioning operation may be triggered on the set of replicas of the front-end tier. Obviously, the latter unprovisioning operation on the front-end tier is not necessary, and is only triggered because the back-end tier of the application is in an instable state. Careful capacity planning, in this context, turns out to be of great importance in order to prevent these unnecessary (un)provisioning operations, which cause system oscillation.

It is known that a challenging issue in resource provisioning is to efficiently tackle both situations of gradual load variations and load peaks [56]. This workload variation reflects different client usages at different times.

In a cloud environment, for example, resource provisioning plays a key role in ensuring that the cloud providers adequately accomplish their obligation to costumers while maximizing the utilization of underlying infrastructure. An efficient resource management scheme would require to automatically allocate to each service request, the minimal resources needed for acceptable fulfillment of SLAs, leaving the surplus resources free to deploy more virtual machines. The provisioning choices must adapt to changes in load as they occur, and respond gracefully to unanticipated demand surges.

As we mentioned, the focus of this work is not on the resource provisioning itself. In future work we will define strategies to cope with this task. The main question we intend to aswer is "How can we accurately predict a given behavior, or react to workload

surges efficiently, in order to increase the gap between the moment unwanted situations happen and the moment information is provided with means to avoid it?".

Because we're looking to make judgments and predictions on a quickly changing landscape, approximations will be necessary, and it's important to realize what that means in terms of limitations in the process. Being aware of when detail is needed, and when it's not, is crucial to forecasting budgets and cost models.

As such, we hope to come up with a helpful tool, intended to be attached to provisioning frameworks, which will be able to provide means to avoid the necessity to monitor changes that are not significant to an application's context but which can equally be able to trigger decisions that will most likely need to be readily undone, and to perform meaningful forecastings to efficiently detect peaks.

An overview of the core of our approach is depicted in *Figure 4.2*. Its first component consists in obtaining a higher level description of a set of signals $f_i(x)$, which respresent monitored workload traces, by applying the scale-space filter. This provides us the ability to select a scale or a set of scales by looking at how the interpretation of the data changes as the scale is varied. It allows us to pick that interpretation which is more suitable to the granularity level demanded by the case we will be dealing with. The obtained signal can be stored to be used by reactive provisioning algorithms.

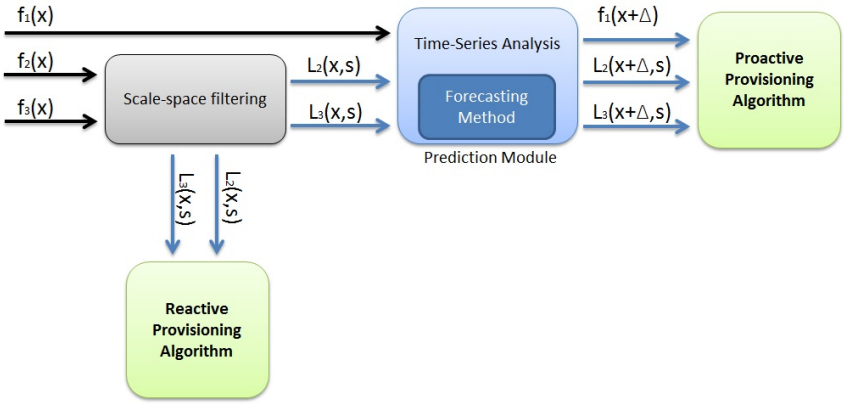


Figure 4.2 Our approach. Scale-space filtering is optionally applied to a signal $f_i(x)$, which generates another signal $L_i(x, s)$ in a new scale. The obtained signal might be used by a reactive provisioning algorithm. Also it can be used at the prediction module to generate a forecast that may be used by a proactive provisioning algorithm.

If it is desired to adopt a proactive solution, the prediction module, which wraps the training and prediction steps using specific forecasting models, might receive signals

which are not preprocessed or the signals in their new scale after application of scale-space. Forecasts are then performed in order to be used by a proactive provisioning algorithm.

With means to improve the models accuracy, forecasting analysis are obtained according to two different approaches: the first obtains all the values for the prediction window at once, based on the observation window, as shown in *Figure 4.3*; the second, known as *sliding-window*, performs several one-step ahead predictions in order to obtain all the values for the prediction window for a fixed size sliding-window. At each iteration, the window is moved one step forward to forecast the next point, as depicted in *Figure 4.4*.

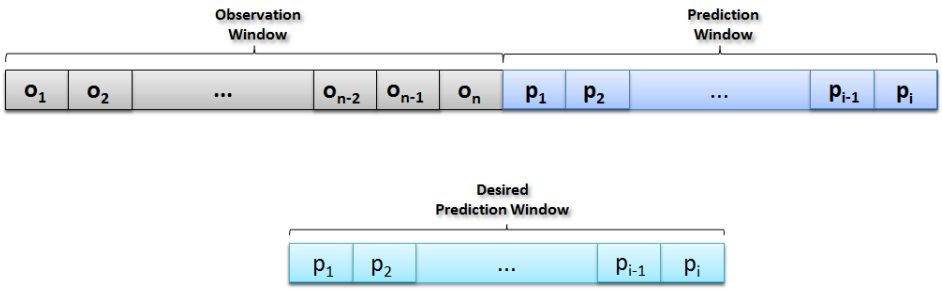


Figure 4.3 Direct forecasting without sliding-window. The desired prediction window is obtained in a sole iteration, in which all its values are obtained

Optionally, at each iteration, we may also move the window $x \geq 1$ steps forward to forecast the next x points.

We also add the possibility of performing sampling. It involves selecting a subset of all the analysed points, being each point from the sample obtained from intervals of a fixed size, as exemplified in *Figure 4.5*, for an interval of size four. The choice of which point to pick in each interval may be based on the minimum or the maximum values, the median of all the points or the average, to name a few.

We made use of a synthetic workload as $f(x)$, as described in the previous section. It was obtained by modulating the request rate of the TPC-C benchmark with a trace that corresponds to the access rate observed in a real-world scenario.

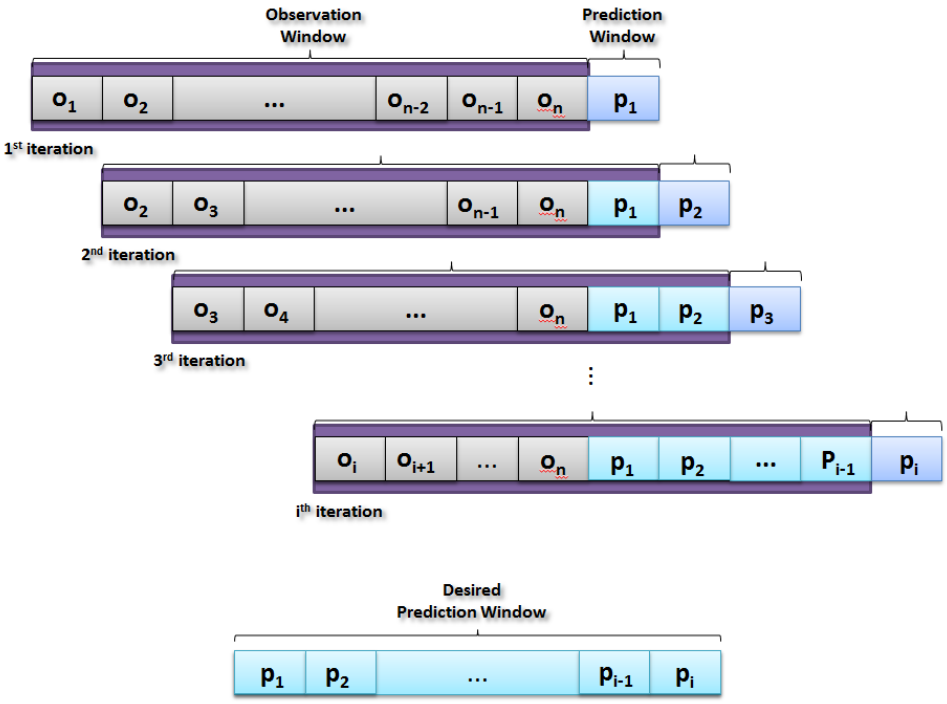


Figure 4.4 Sliding-window forecasting. The desired prediction window is obtained in multiple iterations. One new element from it is obtained in each iteration, which then becomes part of the training set. Also, one element is dropped from the begining as the window slides.

4.3 ARCHITECTURE

Based on the ideas presented in the previous section, we developed a framework to encompass the techniques described as a module that may be used by provisioning strategies. This module's architecture is presented in *Figure 4.6*.

In (A), we receive a signal that may come from a database table or from a file. The latter option gives us more flexibility and might be usefull in case we want to study a signal offline and wish to apply changes to data more easily. The former, is mainly intended to the online application of our method. It allows us to collect a signal from a table in intervals we can easily control through a query. Also, we are able to extend its capabilities by implementing queries that obtain variations of the collected data to feed our module with more specific metrics. We believe this might be useful when we attach our module to provisioning strategies in a cloud environment since we are able to include rules in these queries which may reflect SLA requirements. To the received signal, we may apply *scale-space* smoothing (B). The resulting signal might then be recorded to a file or a database table (C), with means to be used by reactive provisioning algorithms.

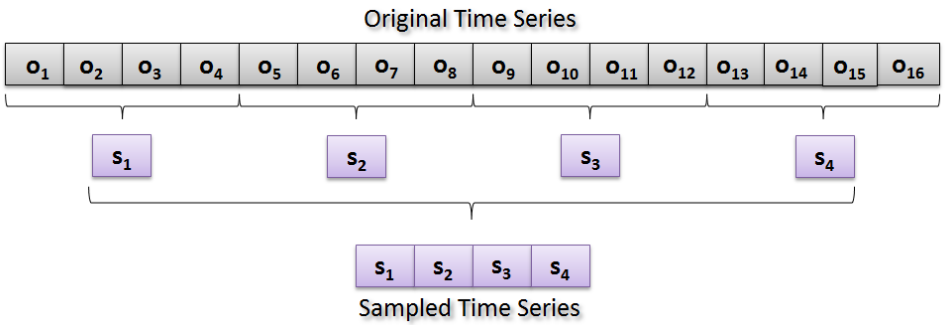


Figure 4.5 Sampling technique applied to a time-series in intervals of size 4. At each interval a sample is chosen by picking the minimum element, the maximum or by taking the average or the median of the elements in the interval.

If the intention is to use a proactive strategy, either the original signal (D) or the scale-space smoothed one (E) are sent to the forecasting module. There, we are able to apply the methods we use, SVR and ARIMA, to the whole signal or to chunks of it. Although we focus our attention to these two methods, any other can easily be attached to the architecture. *Sliding-window* technique may optionally be applied to obtain the forecasting (F). Should the ammount of points we gathered be too high, we may also perform sampling to it. We have implemented this technique in a *sampling module* (G) through the minimum or the maximum element in the interval, or by the median or the average of them, which may also be easily extended. Weather these auxiliary techniques are used or not, we are able to control the sizes of the observation and prediction windows in all the cases. Finally, an *error module* (H) provides us the implementeation of metrics that can be directly applied in order to measure the quality of the obtained predictions. Any other metric can be included here.

In the end, we are able to record the forecasting result to a file (I) or to a database (J). The result might be the time-series of predicted values or, optionally, information inferred from these predictions. Again, in a cloud environment this might be useful since instead of feeding a provisioning system with a time-series that needs to be processed, we are able to send specific information, for example concerning SLA vioations in a given time in the future.

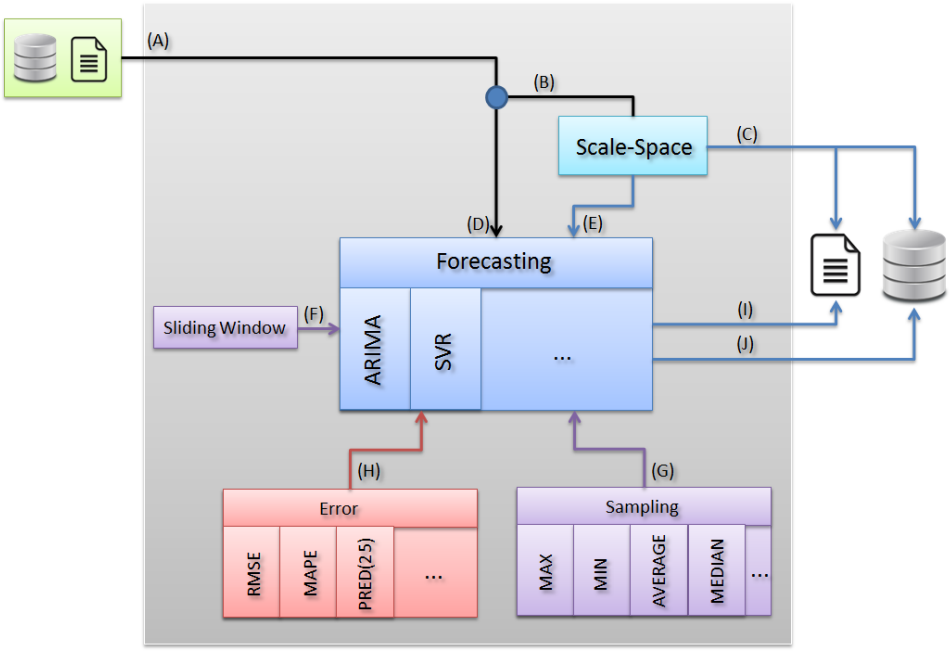


Figure 4.6 S-SWAP architecture.

4.4 IMPLEMENTATION

We implemented our framework in Python, a very popular interpreted programming language that combines remarkable power with very clear syntax. The reason we chose it is mainly due to the fact that it provided easy to use interfaces to necessary external libraries we needed and also made it very easy to perform string manipulation without compromising performance.

In our implementation of the scale-space algorithm, we pass as input parameters the original signal and *"s"*, which explicitly represents a threshold in time units under which peaks and depressions are considered irrelevant. The algorithm then obtains the Gaussian's standard deviation $s = \sqrt{2t}$ to perform operations and returns the smoothed signal. Interpolation is supported.

The *forecasting module* uses a python inteface to *LIBSVM* [57], a library for Support Vector Machine that supports various SVM formulations for classification, regression, and distribution estimation: C-SVM classification, one-class classification, ν-SV classification, ν-SV regression and ϵ-SV regression. In this work, we have used ϵ-SVR. Also we apply the Radial Basis Function (RBF) kernel. The module also adopts *RPy2*

[58], a python interface to the *R programming language* [59]. To generate ARIMA forecasts, we used the *auto.arima* implementation, with its default behavior, provided by the *Forcast* [60] R package. It returns the best found ARIMA model through by using a variation of the Hyndman and Khandakar algorithm which combines unit root tests, minimization of the Akaike Information Criterion (AICc), a measure of the relative quality of a statistical model, and Maximum-Likelihood Estimation (MLE), a method of estimating the parameters of a statistical model.

4.5 SVM PARAMETRIZATION

Contrary to ARIMA, to which we had available an implementation in R based on automatic parametrization capability, similar easiness did not apply to SVM, since performing grid search did not come up with good results. That lead us to try to understand the relationship between its parameters to be able to empirically determine a group of values that would give us good performance.

Training an SVM means to determine the parameters α_i and b for specific input values, which can be recalled from the SVM explanation in *Chapter 2*. SVR has another set of parameters called hyperparameters: The soft-margin constant, or cost paramater, C; the kernel parameters (γ, the inverse-width parameter in the RBF Kernel) and the ε value in Vapnik's ε-insensitive loss function. The biggest difficult in setting SVR generalization capacity resides in the complex interdependence relationship among these hyperparameters [61]. It is very hard to tell which are the best hyperparameters values for any dataset or any kind of problem. As [62] reinforces, it is not straightforward to select those hyperparameters properly and how to determine a set of proper hyperparameters is still suboptimal and computationally expensive, if not clumsy.

What we can specify are parameters regions that have higher probability to obtain better results. Parameters variate from problem to problem but, as these parameters change, one may realize certain behaviors that negatively affect the result or that will not interfere anymore [63]. The curve estimation problem also offers important *a priori* information that can be naturally used in the model selection task.

To conduct this study we used SVM-TOY, a simple graphical user interface from the LIBSVM package, which has gtk or qt graphical user interfaces, to display the obtained SVM decision/regression function on the $2-D$ plan. The trace we adopt has been

synthetically generated based on the workload used in [64].

In order to carry out the analysis of the hyperparameters, recall the objective function from Equation 2.12, which consists of two terms. As [65] points out, the first term, $\frac{1}{2}\|w\|^2$, captures the degree of complexity, which is proxied by the ε-insensitive region between lines $y_i = w^T \cdot \phi(x_i) + b + \varepsilon$ and $y_i = w^T \cdot \phi(x_i) + b - \varepsilon$. If $w = 0$, then complexity is minimal since the ε-insensitive region is biggest. All points $i = 1, 2, \ldots, l$ inside the ε-insensitive region have both $\xi_i = 0$ and $\xi_i^* = 0$. If a point i lies outside the ε-insensitive region, then either $\xi_i > 0$ and $\xi_i^* = 0$ or $\xi_i = 0$ and $\xi_i^* > 0$. So, ε, the width of the insensitive region of the cost function is the parameter that defines the sparseness of the SVR solution. It also influence on the smoothness of the mapping: the larger its value, the smoother the result, as we observe in *Figure 4.7*.

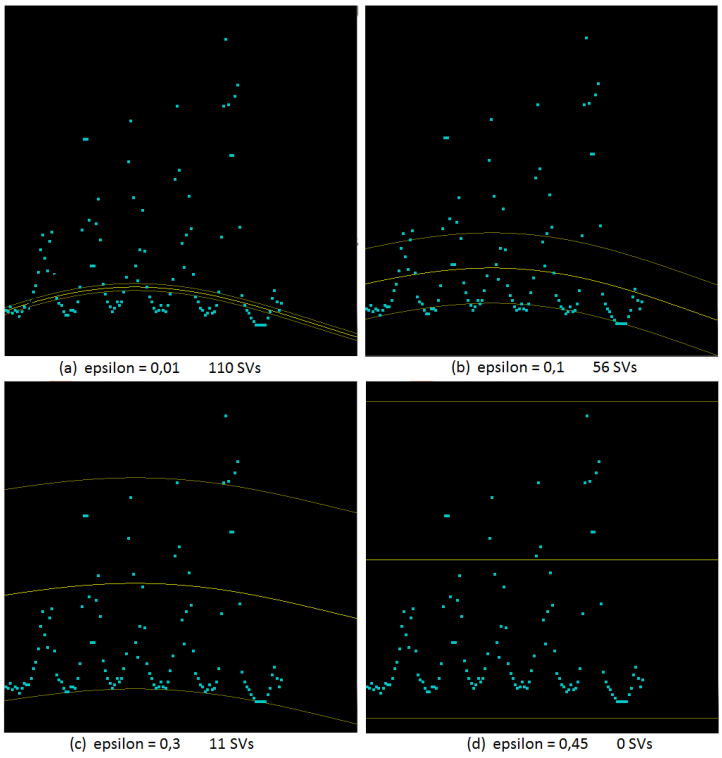


Figure 4.7 The influence of the hyper-parameter ε for fixed values of $C = 1$ and γ set to SVM-TOY's default value, $\frac{1}{k}$, where k is the number of attributes in the input. In our case, $\gamma = 0,5$. Note that, in (d), even increasing the constant C to very high values, the region remains unchanged, since loss occurs only if a point lies outside the ε-insensitive region.

The manually adjustable soft-margin constant, C, determines the trade-off between function's complexity, $\frac{1}{2}\|w\|^2$, and the overall loss associated with it, $\sum_{i=1}^{l}(\xi_i + \xi_i^*)$. Thus, the second term of the objective function, $C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$, stands for the actual

amount of loss associated with the estimated function, since loss occurs only if a point lies outside the ε-insensitive region. Notice, in *Figure 4.8*, that a large value of C will lead to a behavior similar to that of a hard-margin SVM (no misclassification). It can be a doubtful choice if the data are known to be noisy. Noisy data (i.e. outliers) are often better modelled with values of C, which allow for training errors. In the dual formulation, C defines the upper bound of the multipliers α_i and hence defines the maximal influence the sample can exert on the solution.

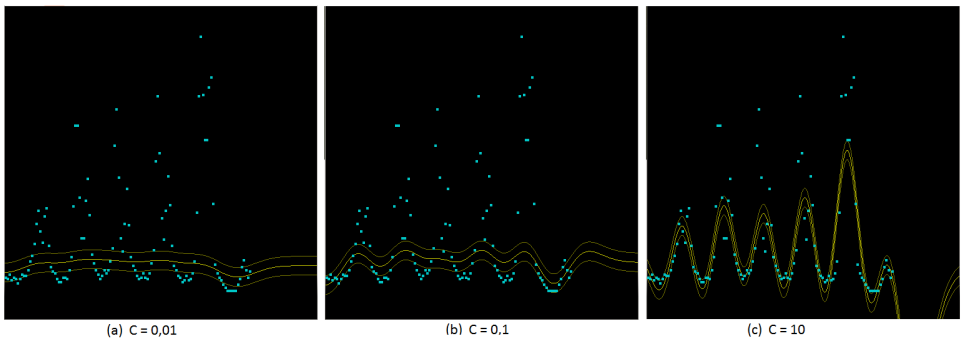


Figure 4.8 The influence of the hyper-parameter C for fixed values of $\varepsilon = 0,03$ and $\gamma = 100$.

Thus, What the SVR line aims for is to find a balance between the amounts of “flatness” (or complexity), and the amount of training mistakes (or fit). This is the fundamental idea behind SVR analysis: good generalization ability is achieved when the best trade-off between function’s complexity and function’s accuracy on the training data is being struck. The parameter ε in the ε-insensitive function and the soft-margin constant, C, are powerful means for regularization and adaptation to the noise in training data.

Also, model complexity is (roughly) proportional to the value of γ. Intuitively, if $\phi(x)$ and $\phi(x)$ are close together, then we might expect $K(x, z) = \phi(x)\phi(z)$ to be large. Conversely, if $\phi(x)$ and $\phi(z)$ are far apart - say nearly orthogonal to each other - then $K(x, z) = \phi(x)^T\phi(z)$ will be small. So, we can think of $K(x, z)$ as some measurement of how similar are $\phi(x)$ and $\phi(z)$, or of how similar are x and z. The RBF-Kernel (or Gaussian) is a resonable measure of x and z’s similarity, and is close to 1 when x and z are close, and near 0 when x and z are far apart as depicted in *Figure 4.9* (consider x a point in the origin and z sliding in $(0, \pm 2)$).

Lower γ means that the kernel is a "flatter" Gaussian and so the regression function is "smoother" with a large sphere of influence for a given data point. Higher gamma makes it a "sharper" peak, and so the regression function is more flexible and

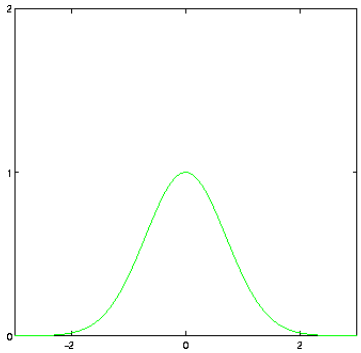


Figure 4.9 Gaussian RBF with centre c = 0 and radius r = 1

able to reproduce strange shapes if they're the right answer. In this case, the sphere of influence is much restricted. In the limit ($\gamma \to \infty$), a data point is not correlated with any other data point. See *Figure 4.10* for reference.

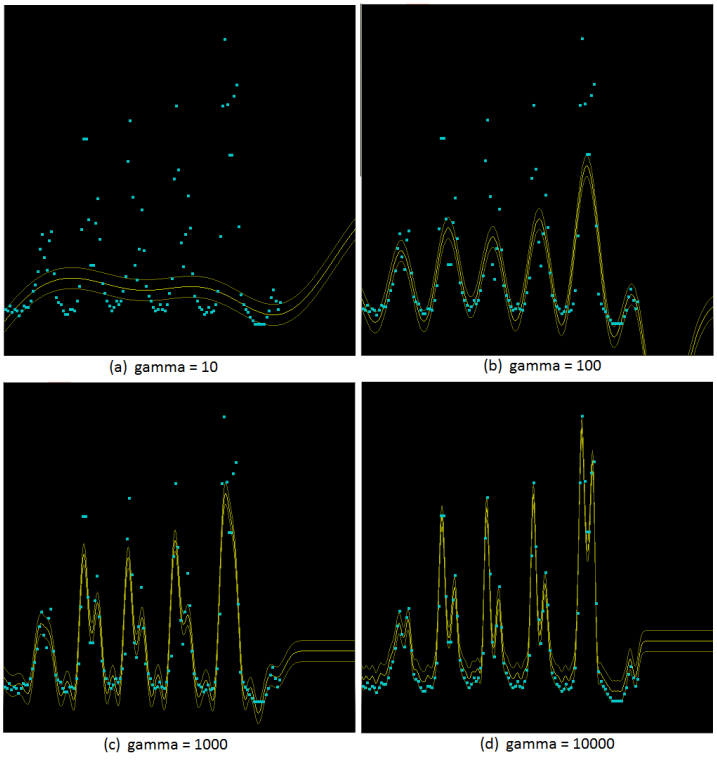


Figure 4.10 The influence of the hyper-parameter γ for fixed values of $C = 10$ and $\varepsilon = 0,03$.

To make the prior observations clearer, consider the study carried out by [66], which consists on the effect of the kernel parameter on the feature space. It states the following two lemmas:

- Lemma 1: Suppose $\Theta_{i,j}$ is the angle between two certain feature vectors $\phi(x_i)$ and $\phi(x_j)$ in the Hilbert space H, we have

$$cos\Theta_{i,j} = \frac{k(x_i, x_j)}{\sqrt{k(x_i, x_i)k(x_j, x_j)}} \quad 0 \leq \Theta_{i,j} \leq \frac{\pi}{2} \tag{4.1}$$

- Lemma 2: Suppose $D_{i,j}$ is the distance between two certain feature vectors $\phi(x_i)$ and $\phi(x_j)$ in the Hilbert space H, we have

$$D_{i,j} = \sqrt{k(x_i, x_i) + k(x_j, x_j) - 2k(x_i, x_j)} \tag{4.2}$$

Incorporate the RBF expression to equations (4.1) and (4.2) respectively, we obtain the angle and distance expressions in the RBF feature space:

$$cos\Theta_{i,j} = exp(-\gamma \left\| x_i - x_j \right\|^2) \quad 0 \leq \Theta_{i,j} \leq \frac{\pi}{2} \tag{4.3}$$

$$D_{i,j} = \sqrt{2 - 2exp(-\gamma \left\| x_i - x_j \right\|^2)} \tag{4.4}$$

We randomly choose two samples x_i and x_j in the sample space and consider two limit conditions:

1. When $\gamma \to 0$, $cos\Theta_{i,j} \to 1$ according to (4.3), then $\Theta_{i,j} \to 0$. From (4.4) we get $D_{i,j} \to 0$. It is reflected that the mapped feature space is a 0 dimension dot;

2. When $\gamma \to \infty$, $cos\Theta_{i,j} \to 0$ according to (4.3), then $\Theta_{i,j} \to \frac{\pi}{2}$. From (4.4) we get $D_{i,j} \to \sqrt{2}$. It is reflected that the dimension of the mapped feature space is l, and that all the vectors in the feature space are orthogonal and the distances between each two vectors are equal.

Thus, [66] draws three conclusions:

1. When γ is increased, the Feature Space Dimension (FSD) is increased monotonically, $\gamma \to 0$ leads to $FSD \to 0$, and $\gamma \to \infty$ leads to $FSD \to l$, where l is the number of samples in the sample sapace.

2. When γ is increased, the angle $\Theta_{i,j}$ is increased monotonically, $\gamma \to 0$ leads to $\Theta_{i,j} \to 0$, and $\gamma \to \infty$ leads to $\Theta_{i,j} \to \frac{\pi}{2}$.

3. When γ is increased, the distance $D_{i,j}$ is increased monotonically, $\gamma \to 0$ leads to $D_{i,j} \to 0$ and $\gamma \to \infty$ leads to $D_{i,j} \to \sqrt{2}$.

The optimal solutions provided by SVM are sparse. It means that a larger part of the weights α_i are zero, due to the specific cost functions, while the remaining nonzero weights contribute to the regression function. Those training data samples that correspond to $\alpha_i > 0$ are called the Support Vectors (SVs). The Support Vectors with $C > \alpha_i > 0$ are the closest to the ε-insensitive region boundary. Notice, if we remove all other points except the SVs from the training data set and train SVM on the SVs only, we will obtain the same boundary, i.e. SVs have the determinant meaning for the given regression task [67]. Sparsity is a very useful feature typically associated with interesting properties such as:

- fast evaluation of the model;
- fast optimization, which is exploited in SVM by many algorithmic approaches;
- statistical robustnes, since sparsity is usually associated with good statistical performance;
- or other computational advantages.

4.6 CONCLUSION

In this chapter we presented the objectives of our work. The architecture of *S-SWAP* was show and implementation details were provided along with the use of external libraries. Since we were not able, contrary to ARIMA, obtain automatic parametrization of SVR, a study was conducted to justify the parameter values we pick for the experiments in the chapter that follows. The obtained parametrization may not reflect the best possible choice, but we believe to be in a region in which good results can be obtained.

CHAPTER 5

EXPERIMENTAL EVALUATION

This chapter presents an evaluation concerning the use of the scale-space technique under two main scenarios, which aim to show its efficiency and weak points when combined with reactive or proactive provisioning approaches. The auxiliary techniques involved in each experiment setting are also explored here.

5.1 EXPERIMENTAL SETUP

Over the obtained workload, we performed two main sets of experiments, as shown in *Figure 5.1*.

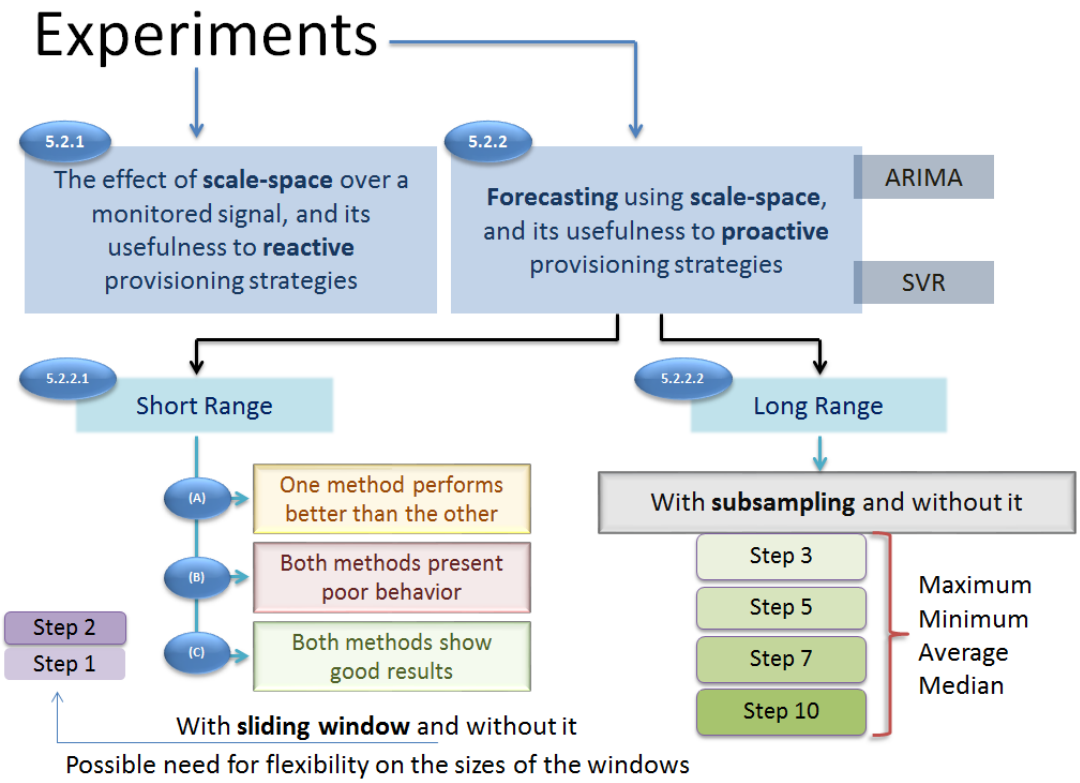


Figure 5.1 Experimental Setup

The first one, *section 5.2.1*, aims to analyse the effect of scale-space over a monitored signal and what advantages it may represent to a reactive scenario. It corroborates the hypothesis about the advantages of using it to deal with sudden workload variations, which may lead to the necessity of taking back recently added resources because of spikes that do not last long enough.

The second one, *section 5.2.2*, correspond to the proactive scenario, in which we perform forecasting with both ARIMA and SVR. With the second group of experiments, two cases are considered:

- *Subsection 5.2.2.1* shows the effectiveness of scale-space in improving the quality of short-range predictions. Here, three scenarios are considered:
  - (*A*): The one in which one method performs better than the other. We first have a look at a situation in which ARIMA provides the best results, and then we show an example that is better dealt by SVR;
  - (*B*): That in which both methods present poor behavior;
  - (*C*): Both methods show good results.

  In order to take the most out of the forecasting methods employed in this case, we also explore the use of sliding-windows, as described in *Chapter 4*. In this experiment we consider sliding-window steps of size one and two. During the experiments, it is important to notice that in some of the mentioned scenarios we had to relax some of the demands of windows sizes in order to obtain reasonable results. This is due to limitations of the methods for given situations. This shows that no method is always 100% good, and that opens a plethora possibilities for future investigations. To finish, we show the execution of short-range predictions over the period of one day.

- In *subsection 5.2.2.2*, we demonstrate that scale-space can also be very effective to aid ARIMA and SVR to obtain long-range predictions, specially if combined with a sampling approach. In this experiment we consider all the sampling methods listed in *Chapter 4*, i.e., minimum, maximum, median and average, each for intervals of size three, five, seven and ten. Results without sampling are also obtained.

In each scenario, whenever relevant, we show the obtained performance levels by means of the error metrics *RMSE*, *MAPE* and *Pred(25)* and also the methods training times.

5.2 EXPERIMENTS

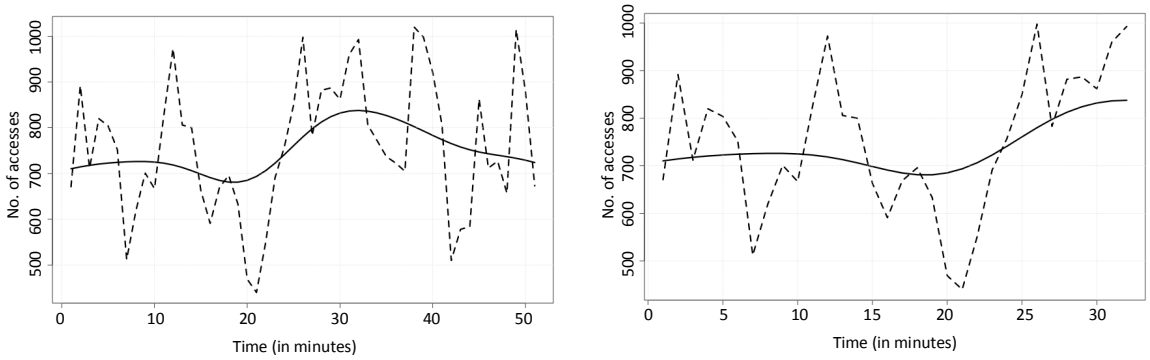
5.2.1 The Effect of Scale-Space Over a Monitored Signal

In order to conduct this experiment, we must first define a time interval under which all information should be considered irrelevant to a reactive provisioning strategy and should thus be removed from the observed signal. Based on the time of observation of related works, we chose this time interval to be of five minutes and, as such, we run the scale-space algorithm in our original signal with $s = 5$. This is because, as we mentioned before, most of the structures in the signal with a characteristic length less than *"s"* units are removed after convolving a signal by $g(\cdot; s)$. Thus we are able to concentrate on the features shown at this specific scale, which prevents us from detaining our attention on irrelevant information. In other words, all peaks and depressions over this interval of five minutes are to be considered as irrelevant, thus avoiding unnecessary provisioning due to transient peaks.

In *Figure 5.2* we represent data obtained from Wednesday, January $25th$. In (a) we ran the scale-space algorithm over the interval from 12 : 27 PM to 1 : 17 PM. Observe that those peaks and depressions that last less than five minutes are no longer represented in the obtained signal. Also notice in (b), which represent data from 12 : 27 PM to 12 : 58 PM, that the behavior of the obtained curve does not change as we execute the algorithm, with the same parameter, over a shorter range of a specific region. This behavior has been observed in every experiment and it guarantees that, if we run the algorithm on-line, over the original signal, irrelevant datapoints will not appear in the obtained scale over which we are focusing our observations. Thus, any variation that appears will be meaningful and prompt action should be considered by a reactive solution.

If we apply another smoothing technique, like the moving average, in the same situation presented in *Figure 5.2*, we may also notice that the behavior of the curve does not change in the shorter region, as we keep the same window size of five in both of them. This is depicted in *Figure 5.3*.

Observe, however, two interesting situations. By choosing a window size equal to five in the moving average, the same value used as the scale-space *"s"* parameter, we are not able to conclude anything related to the type of information that is removed from the signal. Thus, there is no relationship between its parameter and a unit of time that may lead us to have considerable control over the type of information we are



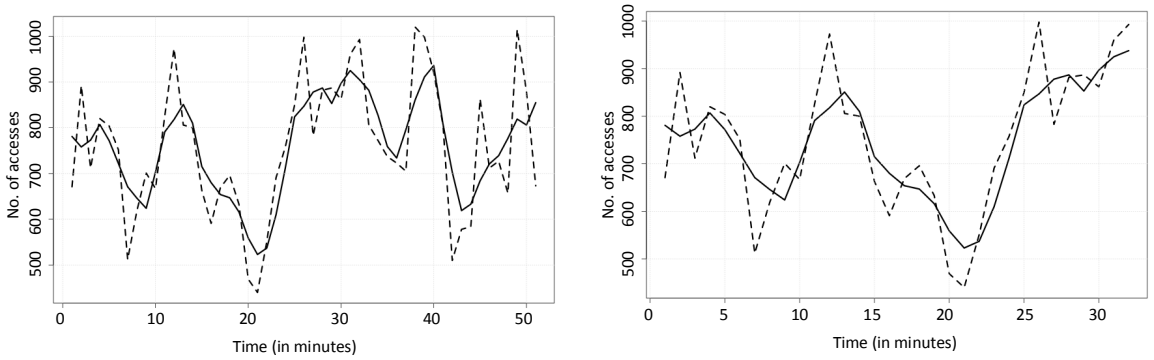
(a) (b)

Figure 5.2 Application of scale-space algorithm over a region of the workload (a) and a shorter range of it (b). The shorter timespan in (b) is obtained by removing samples from the right of the interval shown in (a).

supressing from the signal, as we do with scale-space. Besides, and most importantly, notice that there are some regions where peaks (depressions) appear where before there was a depression (peak). Thus, structures are being added to the obtained signal, and that might compromise resource provisioning strategies. With scale-space, however, due to the *Scaling Theorem*, as we mentioned before, this never happens in scale-space smoothing.

Figure 5.4 clearly shows the difference in the behavior of obtained signals in different scales after applying the scale-space algorithm to the interval from Wednesday, $25th$, $12:27$ PM to $12:39$ PM. If we change the value of *"s"*, from (b) $s = 5$ to (a) $s = 1.5$, the obtained signal will consider relevant those peaks and depressions with more than one and a half minutes.

This shows us that the mathematical properties of scale-space algorithm gives us a strict control over the type of information we want a signal to preserve, by means of the standard deviation parameter, with the guarantee, by the *Scaling Theorem*, that no additional information will be added. Also, the scale-space filtering, by definition, is the convolution of the input curve (length n) with a Gaussian filter of a chosen scale (size s). Thus, the complexity for computing each scale is in the order of O(ns), which is not computationally expensive.



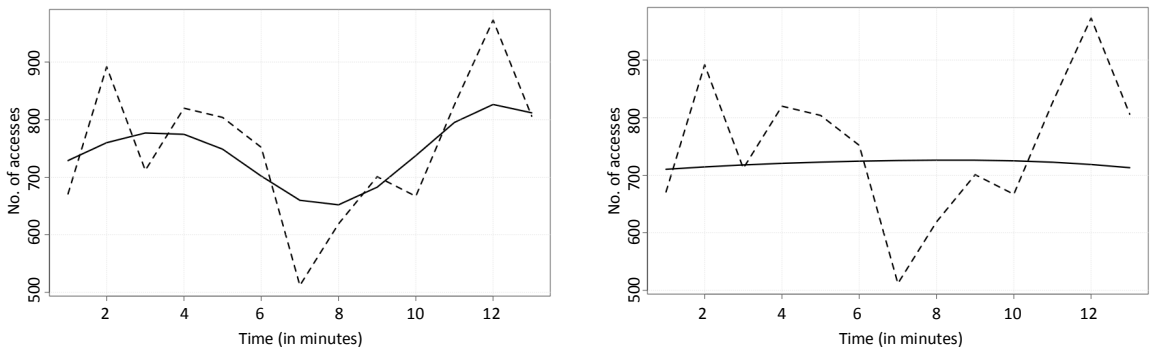
(a) (b)

Figure 5.3 Application of moving average algorithm over a region of the workload (a) and a shorter range of it (b). The shorter timespan in (b) is obtained by removing samples from the right of the interval shown in (a).

5.2.2 Forecasting

For these experiments, we applied ARIMA and SVR regression over both the original signal and the result of scale-space filtering in order to evaluate how would proactive strategies benefit from our approach. We empirically determined a set of values to SVR hyper-parameters which led us to obtain acceptable generalization levels. These values were $C = 175$, $\gamma = 0.010325$ and $\varepsilon = 0.005$. ARIMA, on the other hand, is executed by fitting a model to each observed region by automatically selecting parameters, as described in the implementation details on chapter 4, since this computation is not so expensive.

5.2.2.1 Short-Range Forecasting The experiments in this section test ARIMA and SVR performance over the signal obtained from our workload after applying scale-space filtering with means to perform short-range forecasting. Scale-space *"s"* parameter was kept at $s = 5$. Both the size of the observation and prediction windows were empirically obtained with the goal to maximize the prediction horizon with as few observations as possible. With the considered workload, we were able to obtain a prediction window of size $24min$ for an observation window of size $40min$. As expected, both methods react differently to the observed data. In some situations one performs better than the other, sometimes both perform equally bad or show good performance. There are scenarios,



(a) (b)

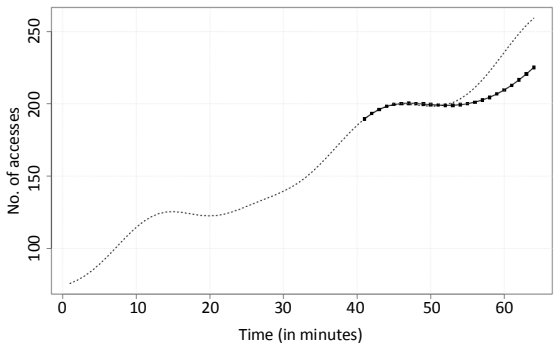
Figure 5.4 Application of scale-space algorithm with different values of *"s"* for the same signal. In (a) we have s = 1.5 and in (b) s = 5.0.

however, in which we must flexibilize the size of our windows if we want to obtain good prediction results. In all the cases, prediction with scale-space performs better than without it.

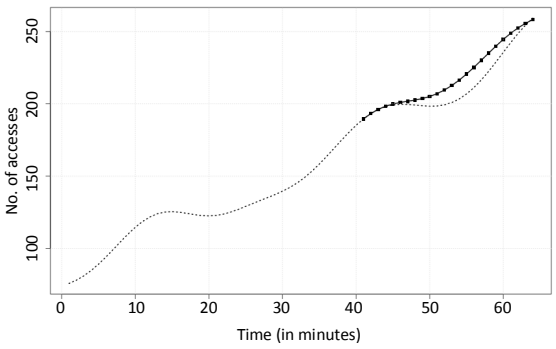
(A) **One method performs better than the other**

For these windows sizes just mentioned, ARIMA proved to be more stable for a wider range of situations when aided by the sliding-window technique. In *Figure 5.5* we depict one such case. Notice how *step 1* sliding-window, (b), is able to reach the peak of the interval, obtaining the best result. In (a) and (c), respectively with no-sliding window and *step 2* sliding-window, ARIMA is not able to reach the peak height.

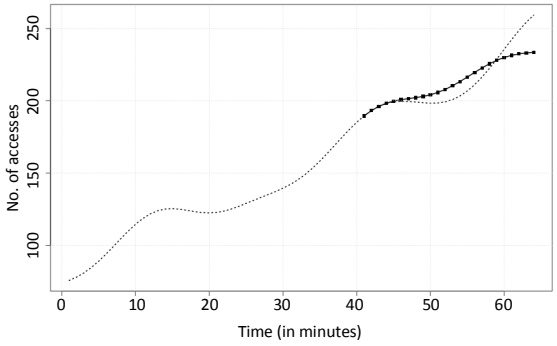
In order to verify whether increasing the size of the observation window would take ARIMA to reach the peak height without the need to apply sliding-window technique and how this technique influentiates ARIMA's behavior in this scenario, we repeated the experiment keeping the size of the prediction window to $24min$ and increasing the size of the observation window to $50min$, *Figure 5.6*, and to $60min$, *Figure 5.7*. In *Table 5.1*, we present ARIMA's training times in millisecond under all the considered settings. As expected, *step 1* sliding-window demands a higher training time since, for the case considered, 24 iterations are executed. *step 2* sliding-window performs 12 iterations and, using no sliding-window, it takes only 1 iteration for the model to be trained, thus demanding less time. Still, in general the training times



(a) ARIMA (no sliding-window)



(b) ARIMA (*step 1* sliding-window)



(c) ARIMA (*step 2* sliding-window)

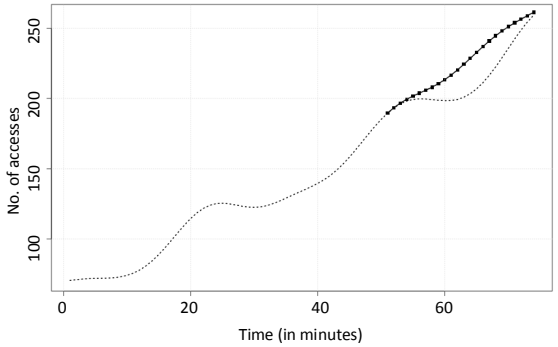
Figure 5.5 ARIMA behavior over a signal with scale-space applied to and considering both no sliding-window and *steps 1 and 2* sliding-windows of size 40

are low, which renders any of the techniques fit to be used by resource provisioning algorithms.

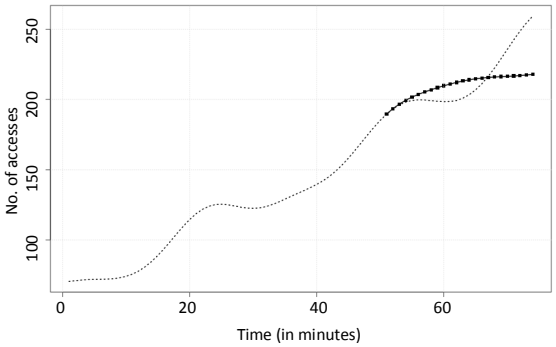
	Observation Window Size		
	40min	50min	60min
NSW	976	517	544
SW1	7221	7794	9250
SW2	5872	4589	5200

Table 5.1 ARIMA's total training time in milliseconds for observation windows of size 40, 50, 60 with no sliding-window (NSW), *step 1* sliding-window (SW1) and *step 2* sliding-window (SW2).

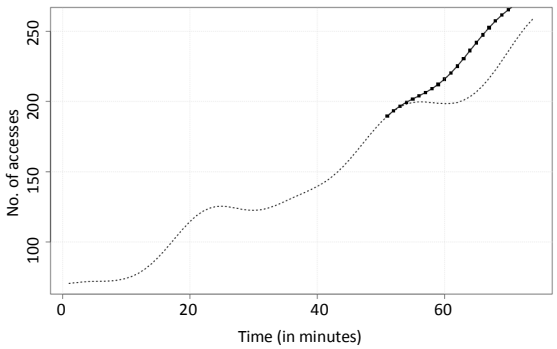
In *Table 5.2*, we present the errors obtained from these executions. An interesting thing to notice from these results is that it is not always that a better error value



(a) ARIMA (no sliding-window)



(b) ARIMA (*step 1* sliding-window)

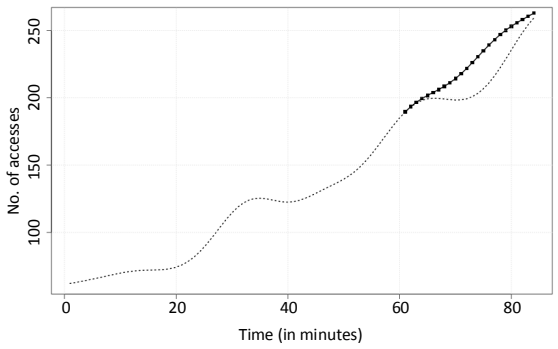


(c) ARIMA (*step 2* sliding-window)

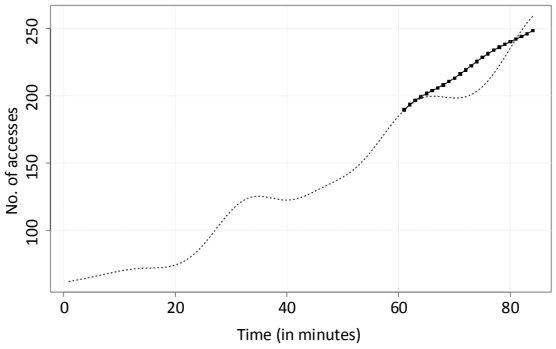
Figure 5.6 ARIMA behavior over a signal with scale-space applied to and considering both no sliding-window and *steps 1 and 2* sliding-windows of size 50

implies better practical results for a given objective. Ours is detecting peaks with the best possible curve fitting. Notice that the best result is obtained by *step 1* sliding-window with an observation window of $40min$, which happens to coincide with the best error value we got. However, suppose we had not reached this scenario. Visually, not applying sliding-window and taking as observatioon intervals $50min$ and $60min$ would be our best choices, since they both reach the peak height. However, in this scenario, the best error values would come from *step 2* sliding-window with an observatioon interval of $40min$, which is clearly worse to our purposes.

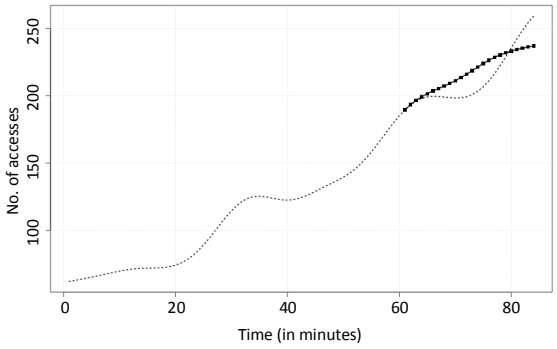
For the scenario under consideration, as exemplified in *Figure 5.8* for an osbservation window of size 40 and prediction window of size 24, SVR was not able to perform well. In (a), no sliding-window is applied and, in (b), *step 1* sliding-window. Also, both methods perform poorly when scale-space is not applied to the signal, as depicted in



(a) ARIMA (no sliding-window)



(b) ARIMA (*step 1* sliding-window)



(c) ARIMA (*step 2* sliding-window)

Figure 5.7 ARIMA behavior over a signal with scale-space applied to and considering both no sliding-window and *steps 1 and 2* sliding-windows of size 60

Figure 5.8 (c). Here, we also do not apply sliding-window. By empirically changing its parameters, we were not able to improve SVR performance for those cases. The obtained error values are presented in *Table 5.3*.

Observe, mainly through *RMSE* and *Pred(25)*, that in the case in which we apply scale-space and *step 1* sliding-window to SVR, error values are considerably worse than the situation in which we do not adopt the sliding-window technique. We have noticed that applying sliding-window to SVR leads to poorer results than forecasting without it. In other words, if the prediction obtained with no sliding-window is not qualitatively satisfactory, applying sliding-window to it will not in general come up with better results.

Sometimes, to a given parametrization, it is necessary to provide SVR with more training points in order for it to perform better for a specified prediction window

		Observation Window Size		
		40min	50min	60min
NSW	RMSE	15.88224	15.37646	16.87352
	MAPE	0.04191	0.05834	0.06432
	PRED(25)	1.0	1.0	1.0
SW1	RMSE	8.05982	16.21801	12.68807
	MAPE	0.02928	0.05084	0.04818
	PRED(25)	1.0	1.0	1.0
SW2	RMSE	9.39771	22.92136	11.82179
	MAPE	0.02963	0.08772	0.04497
	PRED(25)	1.0	1.0	1.0

Table 5.2 Errors obtained for ARIMA prediction without sliding-window, (NSW), and with *step 1* sliding-window, (SW1), and *step 2* sliding-window, (SW2) for observation windows of sizes 40min, 50min and 60min.

		RMSE	MAPE	PRED(25)
SS NSW	ARIMA	15.88224	0.04191	1.0
	SVR	69.08000	0.24563	0.5
SS SW1	ARIMA	8.05982	0.02928	1.0
	SVR	87.05221	0.34165	0.375
OS NSW	ARIMA	100.21736	0.32429	0.33333
	SVR	114.16501	0.39578	0.29166

Table 5.3 Errors obtained for both ARIMA and SVR prediction with an observation window of size 40min and prediction window size of 24. Values are presented for three different cases: with scale-space applied and no sliding-window (SS, NSW); with scale-space and *step 1* sliding-window (SS, SW1) and finally with no scale-space and no sliding-window (OS, NSW)

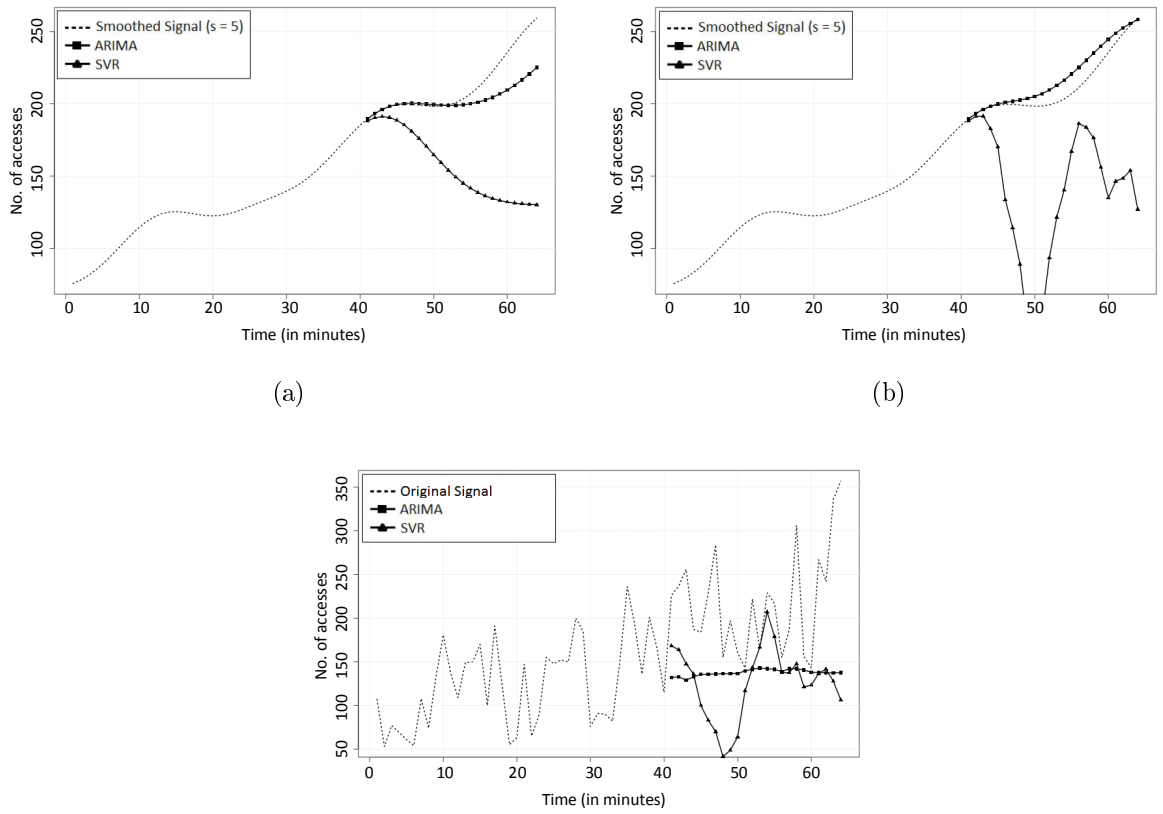


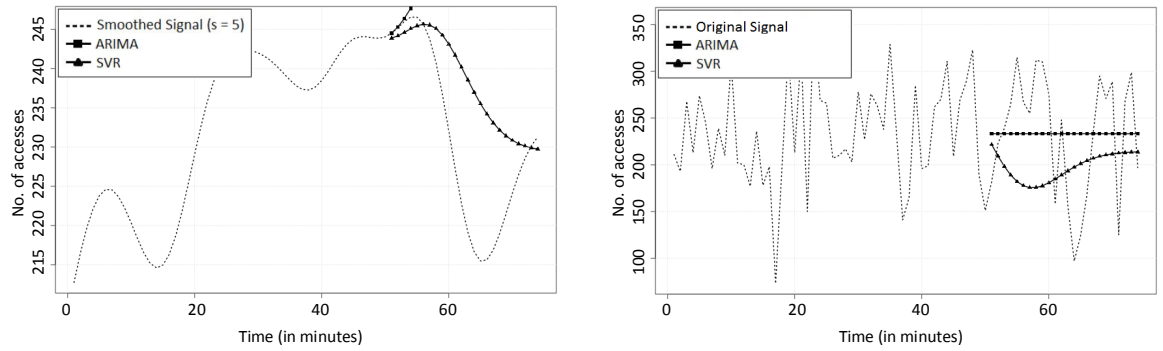
Figure 5.8 In (a), we show SVR behavior compared to ARIMA over a signal with scale-space applied to. No sliding-window is applied, and we consider an observation window of size 40 and prediction window of size 24. In (b), we include *steps 1* sliding-window. In (c), no scale-space was applied to the signal and also no sliding-window.

size. By increasing the observation window $10min$, thus considering now $50min$, and still keeping the prediction at $24min$, we were able to find situations in which SVR is able to make good guesses to find the peaks heights, as depicted in *Figure 5.9*.

Notice in the scale-space smoothed signal that, although SVR is not able to fit well to the curve, and error values are high, as shown in *Table 5.4*, it might still be useful if window sizes must be kept, and ARIMA is not accurate at all.

(B) **Both methods present poor behavior**

There are situations, however, in which even after applying scale-space to the signal and *n-step* sliding-window, neither ARIMA nor SVR, for a given parametrization, will make it possible to obtain accurate forecasting results. This is exemplified in



(a) (b)

Figure 5.9 SVR and ARIMA forecasting for an observation window of size 50 and prediction window of size 24. In (a) we have scale-space smoothed signal, while in (b) we have the original one.

	ARIMA			SVR		
	RMSE	MAPE	PRED(25)	RMSE	MAPE	PRED(25)
SS	28.86917	0.10844	1.0	10.91709	0.03641	1.0
OS	64.38914	0.30212	0.625	77.83718	0.32144	0.45833

Table 5.4 Errors obtained for both ARIMA and SVR prediction with an observation window of 50min and prediction window of 24min. Values are presented for predictions over scale-space smoothed signal (SS), and the original one (OS).

Figure 5.10.

A curious behavior was observed when SVR performs erroneous forecasts, as we can also notice in *Figure 5.10*: it exceeds the value actually observed in the prediction horizon, but this growth in the workload is observed a little ahead. Thus SVR would lead to a pessimist proactive provisioning algorithm that will tend to use a bit more resources than necessary.

(C) **Both methods show good results**

Finally, there are cases in which the only viable solution in order to obtain good forecasting will be to diminish the prediction window size. Other than that, whther increasing the observation window, reducing it or applying sliding-window technique over the scale-space smoothed signal, will be of no efficiency. Two situations exemplifying it are presented in *Figure 5.11*, in which we have direct forecasting applied over a scale-space smoothed signal, with observation windows of size 40 and 50. The prediction window has been reduced to $12min$. Comparison to forecasting over the

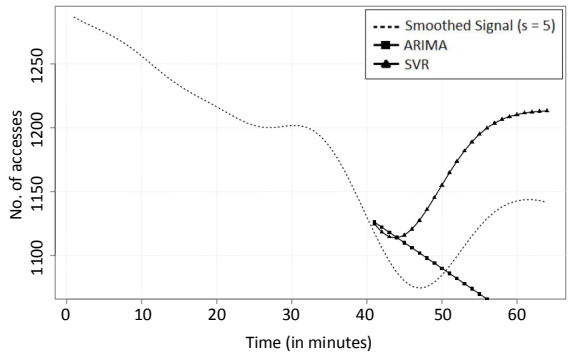


Figure 5.10 Example of a situation in which none of the forecasting methods perform accurate time-series forecasting.

original signal is provided for the second situation. Obtained errors are show in *Table 5.5*, and training times are presented in *Table 5.6*.

	ARIMA			SVR		
	RMSE	MAPE	PRED(25)	RMSE	MAPE	PRED(25)
SS/40min	5.17346	0.01751	1.0	1.31913	0.00445	1.0
SS/50min	0.69494	0.01044	1.0	0.96842	0.01261	1.0
OS/50min	13.05224	0.30035	0.5	26.94465	0.65464	0.16666

Table 5.5 Errors obtained for both ARIMA and SVR prediction with observation windows of 40min and 50min and prediction window of 12min. Values are presented for predictions over scale-space smoothed signal (SS), and the original one (OS).

	Signal/Observation Window Size		
	SS/40min	SS/50min	OS/50min
ARIMA	674	955	438
SVR	9	18	2

Table 5.6 ARIMA and SVR training times in milliseconds for observation windows of size 40 and 50. In the first case results are show over the scale-space smoothed signal (SS), in the latter both scale-space smoothed (SS) and original signal (OS) values are shown for comparison.

We also applied the error metrics in a set of predictions over the period of one day, on January $24th$, in order to evaluate the suitability of each SVR and ARIMA to our workload on a continuous space of time over both the original signal and the scale-space smoothed one. These predictions were run every ten minutes, based on the last 30 obtained samples, thus representing observation windows of $30min$ each, with the

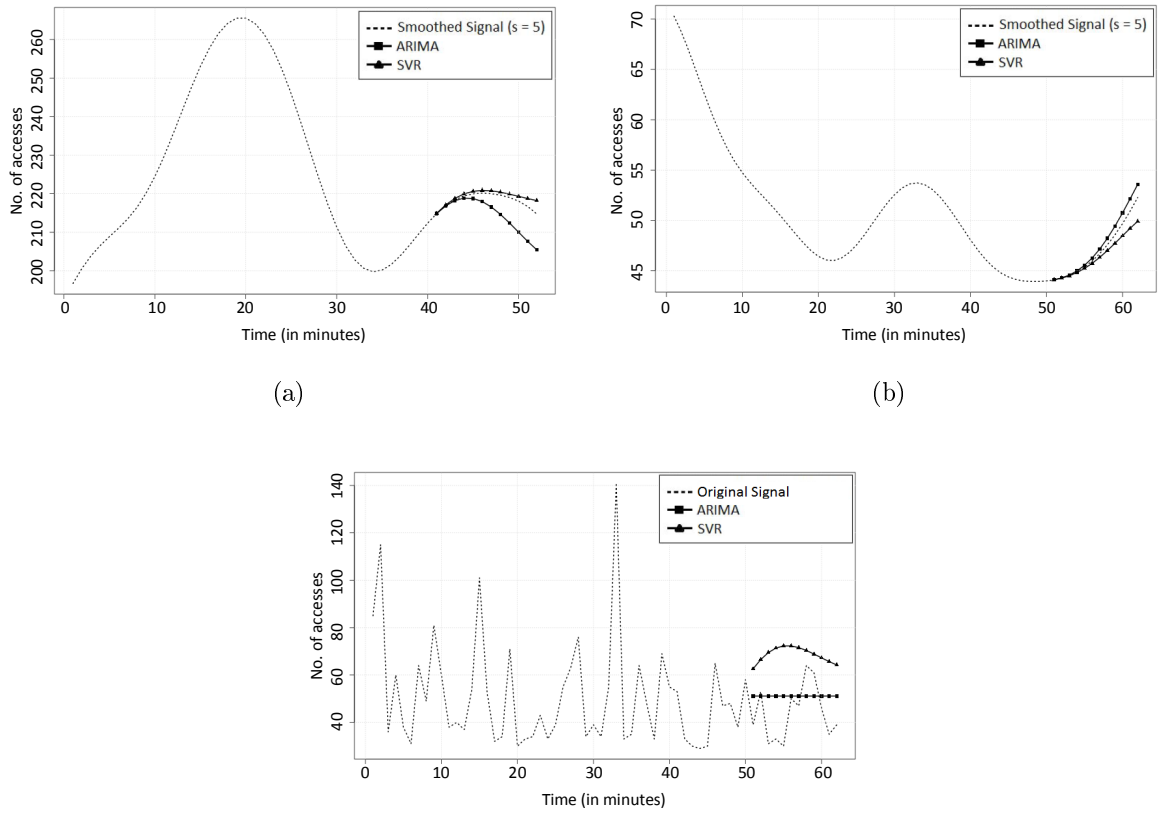


Figure 5.11 Example of a case in which both methods show good accuracy for a diminished prediction window of 12min. In (a), forecasting is performed over the scale-space smoothed signal and based on an abservation window of 40min. In (b), observation window is increased to 50min and the result obtained is, in (c), compared to forecasting over the original signal.

objective to predict the following $10min$. The values are presented in *Table 5.7*. With the application of scale-space, we were able to considerably reduce the error observed in our predictions, as we can observe in all the metrics. Notice by *PRED(25)* that almost all predicted values fell within a margin of 25% of the actual ones.

	SVR			ARIMA		
	RMSE	MAPE	PRED(25)	RMSE	MAPE	PRED(25)
SS	34,76	0,040	0,994	28,51	0,021	0,997
OS	128.71	0,203	0,738	129,45	0,197	0,762

Table 5.7 Comparison SVR x ARIMA in both the Original Signal (OS) and Scale-Space smoothed signal (SS) for an interval of one day of short-range predictions.

In *Figure 5.12* we present forecasting results described in *Table 5.7* with SVR and ARIMA over both the scale-space signal and the original one for the considered region.

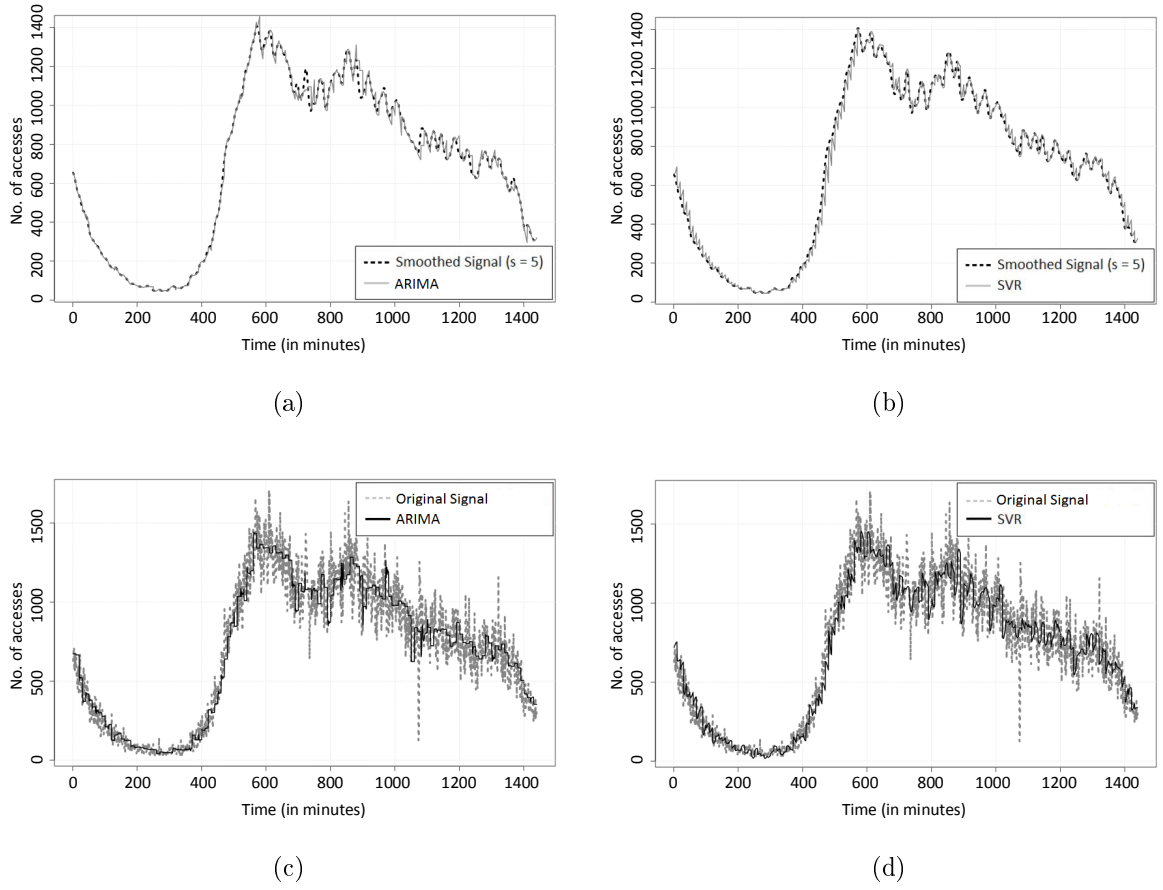


Figure 5.12 Sequence of short-range forecasting over a period of one day. In (a) and (b), ARIMA and SVR applied over the scale-space smoothed signal. In (c) and (d), both methods are applied over the same region in the original signal.

5.2.2.2 Long-Range Forecasting With a simple change we can make our signal suitable, for example, to a daily based analysis. By appling another value to the *"s"* parameter, we are able to elimante information that may be considered irrelevant when we want to predict the behavior of the signal on a specific day based on the observation of previous ones. As such, the experiments in this section test ARIMA and SVR performance over the signal obtained from our workload after applying scale-space filtering with means to perform long-range forecasting. For the tests conducted we considered an observation window of $5829min$, which correspond to about 4 days and a prediction win-

dow of 700min. The size of the prediction window was chosen based on the highest peak of the next day following the observtion window. Our objective was to verify whether the adopted methods would be able to predict the peak's height. In order to obtain the best results among empirically tested SVR parameters, we used the following values for the scale-space smoothed signal: $C = 200$, $\gamma = 0.0005$ and $\varepsilon = 0.5$. Scale-space *"s"* parameter was set to $s = 120$ since for that scale of days, information of a few hours are irrelevant. After applying it, we interpolated the obtained signal in order not to lose infomation on the peaks heights. This is show in *Figure 5.13*.

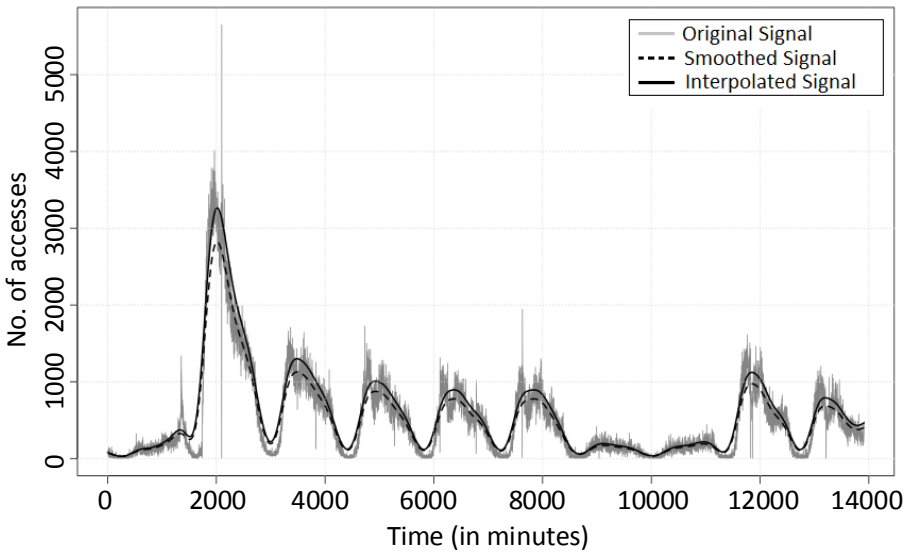
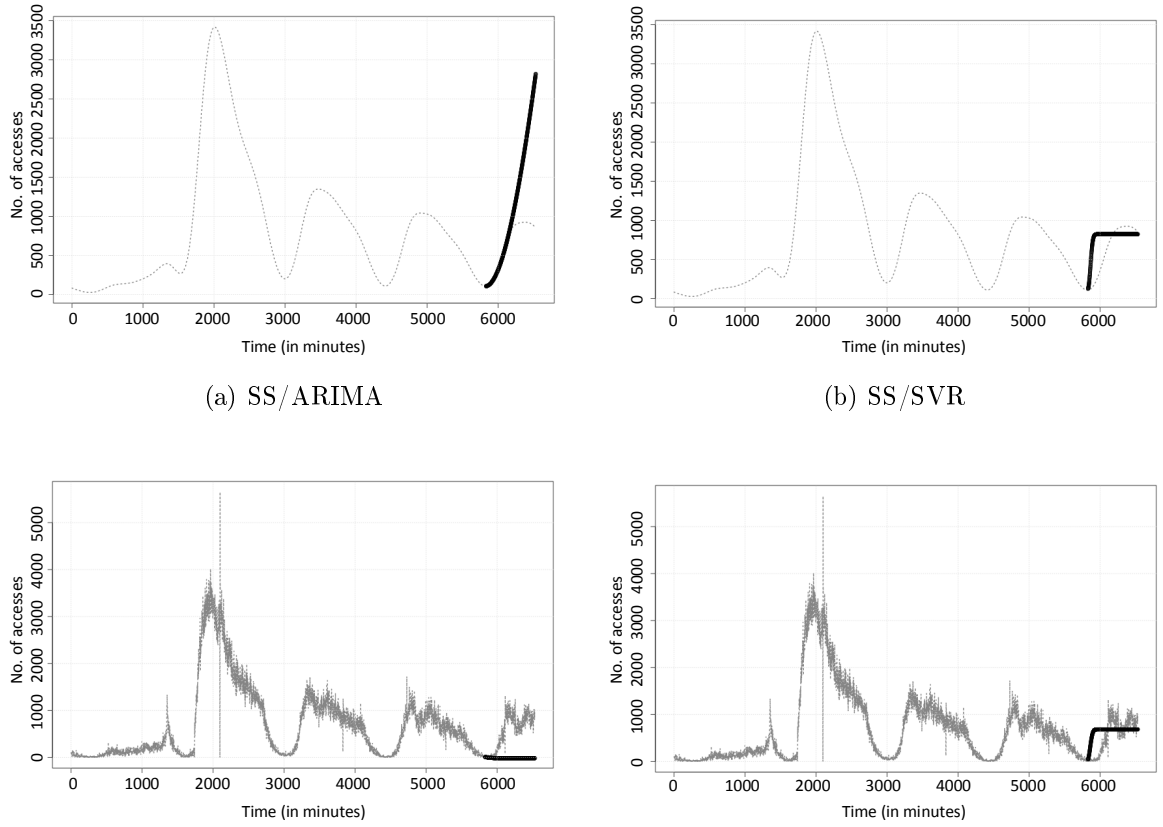


Figure 5.13 Interpolating the signal obtained with the scale-space *"s"* parameter set to s = 120 helps bringing the extrema close to their original amplitude. This is very helpful, since the detection of extrema plays a key role in resource provisioning.

For the original signal, SVR parameters were set to: $C = 200$, $\gamma = 0.00043$ and $\varepsilon = 0.05$. With these parameters set, we run the algorithms for both cases. Results are presented in *Figure 5.14*. The errors for both situations are show in *Table 5.8*, and training times in *Table 5.9*

Notice, in *Figure 5.14*, that the sole application of scale-space is already able to obtain a result from ARIMA, although a very bad one, while with the original signal it has shown no reaction. SVR, although in both cases it presents a behavior typical to overfitting, it is able to guess the height of the peak with a good precision. With the original signal it reaches the value of 684.71, while with scale-space it goest up to 822.04.



(a) SS/ARIMA (b) SS/SVR

(c) OS/ARIMA (d) OS/SVR

Figure 5.14 SVR and ARIMA forecasting for an observation window of size 5829 and prediction window of size 700. In (a) and (b) we have scale-space smoothed signal (SS) with the *"s"* parameter set to s = 120, while in (c) and (d) we have the original one (OS).

	ARIMA			SVR		
	RMSE	MAPE	PRED(25)	RMSE	MAPE	PRED(25)
SS	470.21749	0.35138	0.64428	281.63054	0.74556	0.61571
OS	686.16672	1.12503	0.00857	340.50752	10.57002	0.38857

Table 5.8 Comparison SVR x ARIMA in both the Original Signal (OS) and Scale-Space smoothed signal (SS) for an observation window of 5829min and a prediction window of 700min.

	ARIMA	SVR
SS	709	69703
OS	1492	2347

Table 5.9 Comparison SVR x ARIMA in both the Original Signal (OS) and Scale-Space smoothed signal (SS) for an observation window of 5829min and a prediction window of 700min.

In the latter case, since interpolation has brought the signal up to reflect a more accurate peak detection, obscured by the noise in the original signal, chances are that it might be less prone to cause a situation of underprovisioning.

Notice that the errors are improved with the use of scale-space. SVR training time however has considerably increased with the application of scale-space. Recall that the parameters values for both the scale-space smoothed signal and the oringinal one are different. In the former case, we noticed that if we increased the value of the ε we would slowly reduce training time. The same effect would be obtained more quickly by reducing the value of the C parameter. However, since both these actions also contribute to reduce the method's accuracy, we decided to keep the chosen values, after all a training time of a few minutes does not incurs negative effects on long-range predictions for the following day.

Still, none of the above results are desirable. We wish to obtain good results from ARIMA and avoid an overfitting situation with SVR. Given our needs, and since we are only dealing with these two forecasting methods, we decided to verify the influence of the ammount of points considered to the quallity of our predictions through sampling. Performing sampling is admissible given that the error introduced by this process is not very significant in the approximation of the original signal. As the number of samples is reduced, calculations are in general simplified and the overall method is expected to perform faster.

Thus, we performed some tests with both ARIMA and SVR considering $\frac{1}{3}rd$, $\frac{1}{5}th$, $\frac{1}{7}th$, $\frac{1}{10}th$ of the points from the original signal, which correspond respectively to sampling intervals *step 3*, *step 5*, *step 7* and *step 10*. The choice of which points to consider were based on: the maximum, the minimum, the median and the average of points for each interval in the considered signal. Results from these experiments are presented, in terms of the obtained errors, in *Figure 5.15*, in which we show *RMSE* values for all the cases, and *Figure 5.16* and *Figure 5.17*, which contain respectively *MAPE* and *PRED(25)* errors. Training times are also presented in *Figure 5.18*

By the analysis of the errors in *Figures 5.15, 5.16 and 5.17* a few things should be noticed: scale-space leads to the best results; by looking at the three graphics, SVR, with $\frac{1}{5}th$ of the points, has obtained the best performance together with ARIMA in the case of median sampling with $\frac{1}{3}rd$ and average with $\frac{1}{5}th$ of the points. From *PRED(25)*, in these cases it can be seen that all predicted values fall within a 25% margin of the

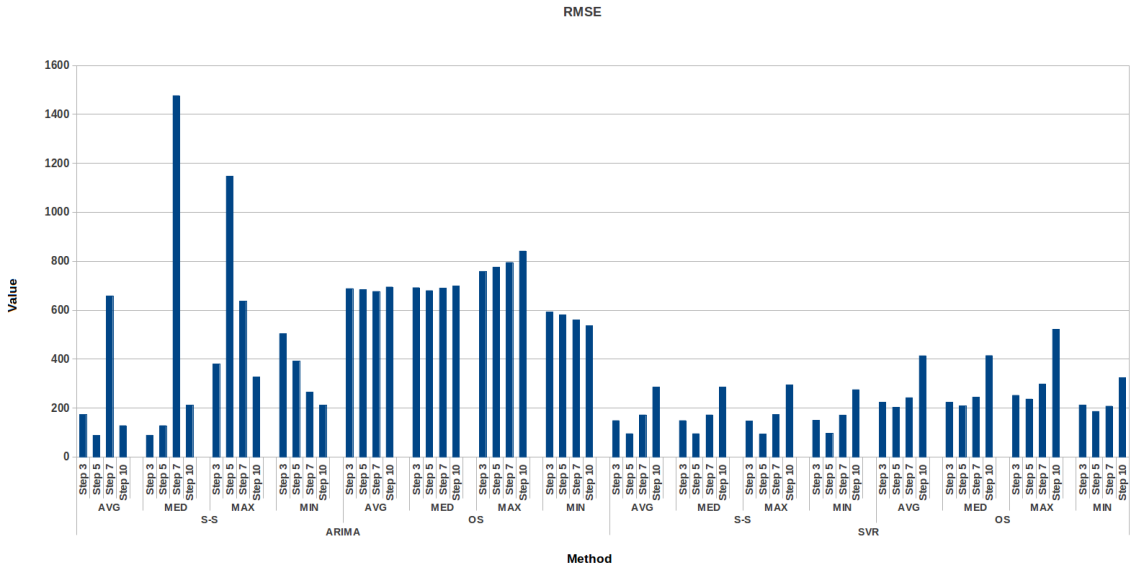


Figure 5.15 *RMSE* values for long-range predictions with ARIMA and SVR considering min, max, med and avg sampling methods for 1/3rd, 1/5th, 1/7th and 1/10th of the total points. Results for both the scale-space smoothed signal and the original one are presented.

actual ones; SVR shows a more behaved pattern given its fixed parameters in each OS and SS cases. Observed that reducing the ammount of points is only advantageous up to a certain ammount, in our case $\frac{1}{5}th$ of the points, from which forecasting accuracy starts to fall. Such an analysis is not possible to be extracted from ARIMA, which is mainly due to the fact that the method tries to obtain the best parameters for every different scenario, thus changing the model. That allows ARIMA to obtain varied results out of different combinations of sampling size and method; for the same reason, notice that SVR is more indifferent to changes in the sampling method, while for ARIMA that may produce more perceptible variations; finally, in the case of SVR and ARIMA with sampling by the minimum method, both in the original signal, we may observe that metrics do not vary in unison, i.e. the model that is best on one of them is not on the other which indicates that they are probably similar in terms of their average errors.

Also, observe that with SVR, as the number of samples is reduced, the overall method performs faster, indicating that, as we expected, calculations are simplified given that the ammount of support vectors is also reduced. In the case of ARIMA, this is not observed. Again, this is a reflex of the fact that, in each situation, as the model is automatically calculated in order for the best one to be obtained for a specific situation, different parameters are picked, thus leading to varied infuence over the method's training times. SVR on the other hand, has fixed parameters for the original signal and for the

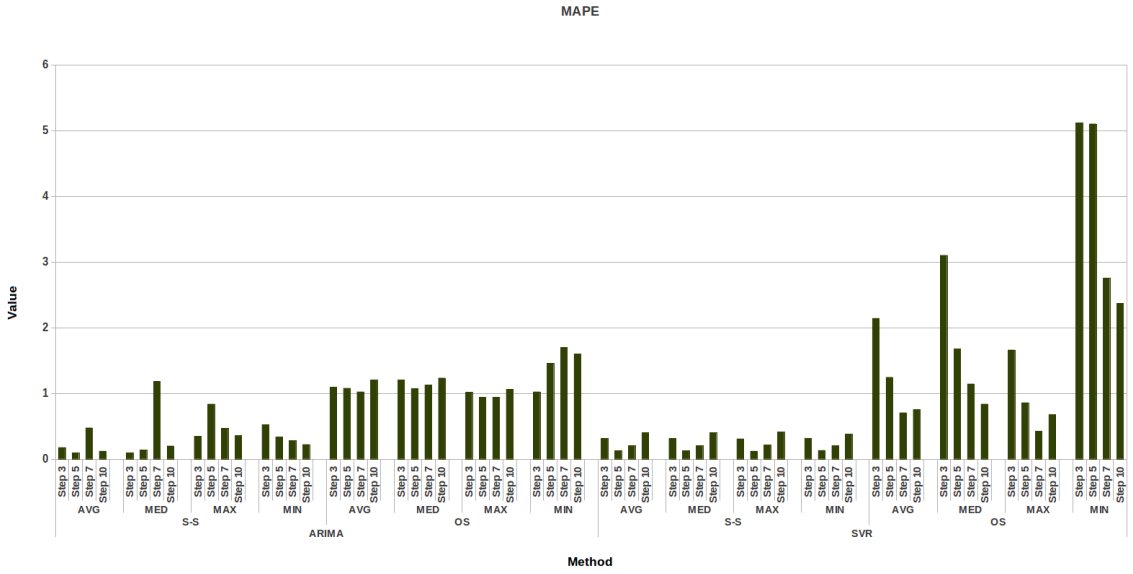


Figure 5.16 *MAPE* values for long-range predictions with ARIMA and SVR considering min, max, med and avg sampling methods for 1/3rd, 1/5th, 1/7th and 1/10th of the total points. Results for both the scale-space smoothed signal and the original one are presented.

scale-space smoothed one. Notice as well that the higher the ammount of points, training time for the scale-space smoothed data is higher than that for the original one, which also concerns to parameters values, as we explained when it was as well observed when we ran the method without sampling.

In *Figure 5.19*, we compare both SVR and ARIMA with and without scale-space for the sampling case in which we consider $\frac{1}{5}th$ of the points obtained through average. Notice how ARIMA obtains superior prediction accuracy and how SVR better fits the curve compared to the previous case in which no sampling was performed. Errors are shown in *Table 5.10* and training times in table *Table 5.11*. Notice the influence that the ammount of points has over SVR training times in comparison to *Table 5.9*.

	ARIMA			SVR		
	RMSE	MAPE	PRED(25)	RMSE	MAPE	PRED(25)
SS	88.34989	0.09460	1.0	94.98720	0.12702	1.0
OS	682.84291	1.07812	0.01428	203.30363	1.24449	0.4

Table 5.10 Comparison SVR x ARIMA in both the original ignal (OS) and scale-space smoothed signal (SS) for 1/5th of the all the points, which are obtained through average.

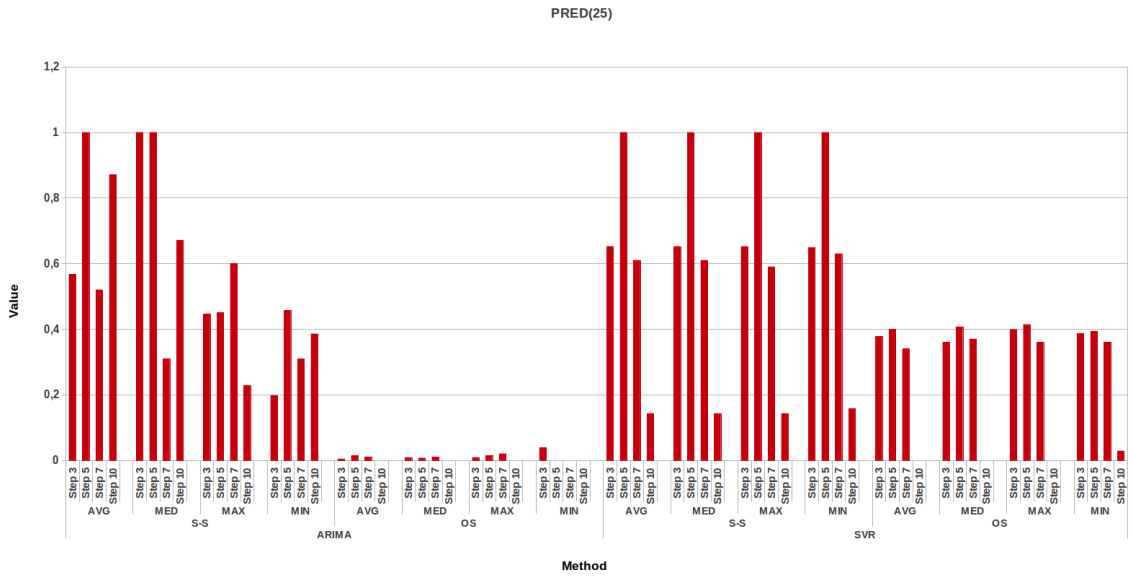


Figure 5.17 *PRED(25)* values for long-range predictions with ARIMA and SVR considering min, max, med and avg sampling methods for 1/3rd, 1/5th, 1/7th and 1/10th of the total points. Results for both the scale-space smoothed signal and the original one are presented.

	ARIMA	SVR
SS	644	519
OS	1205	160

Table 5.11 Comparison SVR x ARIMA in both the Original Signal (OS) and Scale-Space smoothed signal (SS) for 1/5th of the all the points, which are obtained through average.

5.3 CONCLUSION

In this chapter we have presented the evaluation of the scale-space technique with means to be used together with a reactive provisioning strategy and also its efficiency when combined with ARIMA and SVR, two widely used forecasting techniques which are commonly used with proactive provisioning approaches. Also, we evaluated the use of two auxiliary techniques with the objective to improve the quality of our results. In the following chapter, we add the conclusions obtained from this work and also discuss about the future ones that may be derived from it.

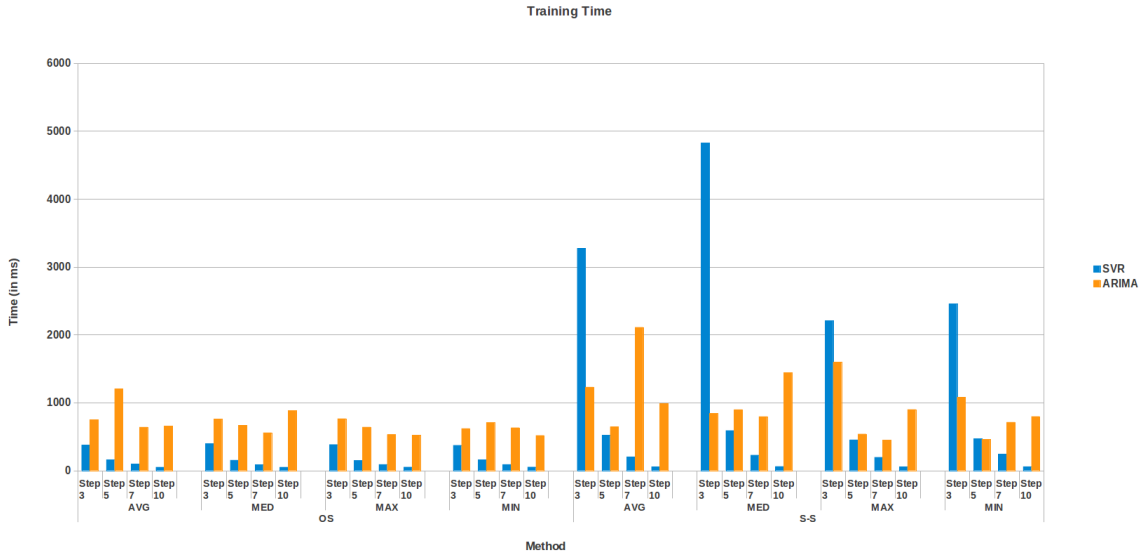
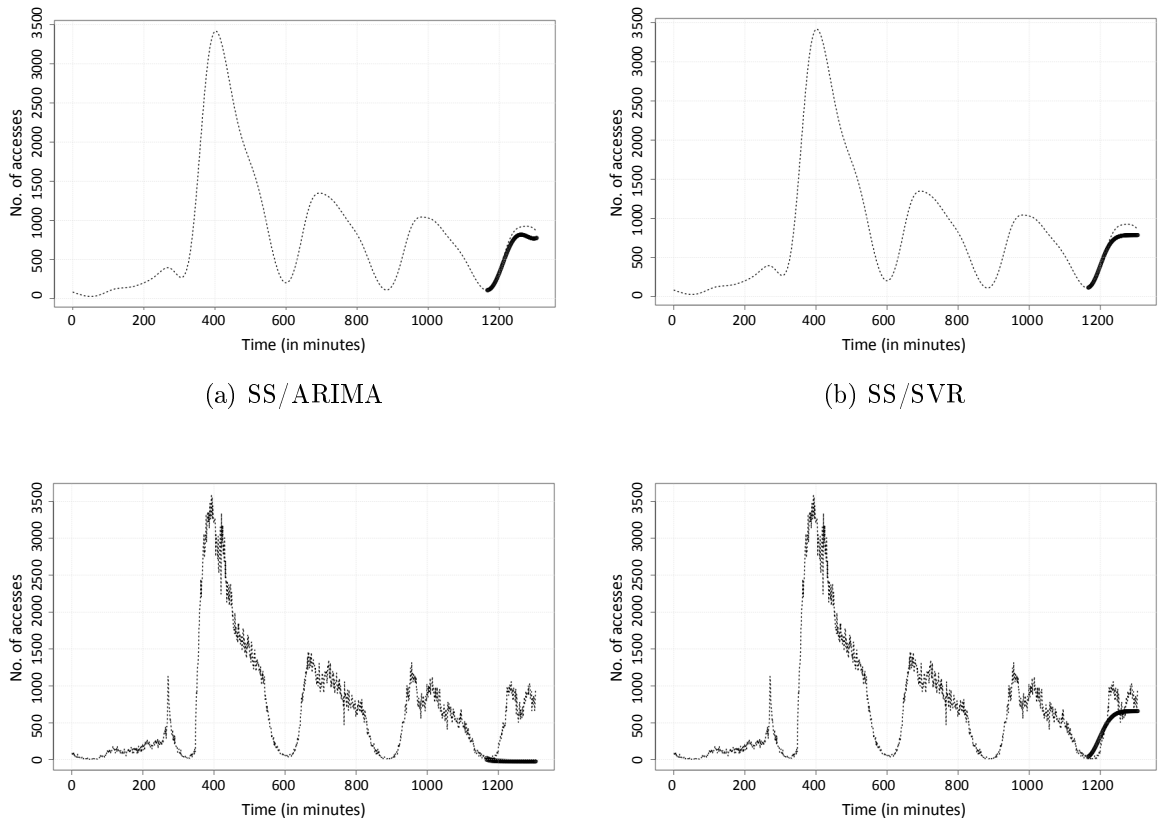


Figure 5.18 Training times obtained with ARIMA and SVR considering min, max, med and avg sampling methods for 1/3rd, 1/5th, 1/7th and 1/10th of the total points. Results for both the scale-space smoothed signal and the original one are presented.



(a) SS/ARIMA

(b) SS/SVR

(c) OS/ARIMA

(d) OS/SVR

Figure 5.19 SVR and ARIMA forecasting for an observation window of size 5829 and prediction window of size 700. In (a) and (b) we have scale-space smoothed signal (SS) with the *"s"* parameter set to s = 120, while in (c) and (d) we have the original one (OS). In all the cases, we have 1/5th of the points obtained through average

CHAPTER 6

CONCLUSION

This research work presented *S-SWAP*, an approach that uses time-series analysis for resource usage prediction in a dynamic environment. It is based on scale-space theory and forecasting and constitutes an independent module that can be adopted by reactive and proactive algorithms in different computational environments, like web, distributed or cloud applications. We compared the performance obtained by two widely used forecasting methods, SVR and ARIMA, which are respectively representatives from machine learning and from statistical time-series analysis. Further, we evaluated our proposed approach under the light of a real-world workload.

The implemented architecture is flexible, allowing its components to be easily extended to meet specific needs. We believe that the characteristics covered make it very attractive to cloud environments, in which elasticity plays a key role.

6.1 RESULTS

To the best of our knowledge, scale-space theory in the context of dynamic provisioning of resources has never been experimented. Because of the *Scaling Theorem*, we have shown that this smoothing technique, unlike others, can be very useful to our purposes. The control it provides over a time-series to remove irrelevant information is very powerful, specially when combined with interpolation. The multi-scale representation of a signal allows an easy interpretation of its behavior at different granularity levels. With these characteristics we have shown that scale-space smoothing may be of great use to reactive algorithms.

More than that, given the simplified representation of the signal while still being able to maintain the extent of relevant extreme points, and the characteristic of not creating new ones, it has proven to be of great utility to increase forecasting accuracy, as we confirmed in the performed experiments. Thus scale-space may also be of great

interest to proactive algorithms. Again, the multi-scale representation of a signal made it easy to turn a time-series suitable to forecasting in a given granularity to be also suitable in different ones, preserving the important qualities in each case. Also, as we have shown, training times were not high for the scenarios we considered.

The provided evaluation has corroborated both hypothesis and, together with that, the application of auxiliary techniques like sliding-window and sampling increased even more chances of obtaining good results as we have also presented in the experiments.

Although our strategy can be used with other types of time-series, it is important to have in mind that scale-space might not be useful if too small values of *"s"*, are applied, since, in the case of a noisy workload, it would not be able to extract the uderlying behavior. Thus, it is necessary to keep a trade-off between data collection interval (CI), and the size of the interval to be considered irrelevant (II). It is important that $CI < II$. How much will depend on the application being considered, and must be empirically determined. Only then will scale-space be able to extract meaningful information from the signal.

After evaluating the approach, one thing however is still challenging. In the cases in which none of the techniques are able to lead to good predictions from certain observed signal, we are not able to figure the reason out, and we are not able to identify, among the tested strategies, what signal characteristics pressuposes which method would be more suitable for that specific situation. We have seen that starting from a good parametrization, in the short-range case, if the obtained result is not good for the observation and prediction windows sizes adopted, applying sliding-window would be the first atempt to obtain better errors for ARIMA, since SVR does not respond well to it. Then, if that didn't improve much, increasing the size of the observation window around ten units would be a good choice. If that, with and without sliding-window, does not lead to better results, it is necessary to reduce the prediction window. Many situations could be solved following these steps. Still many could not. That opens many possibilites for future works, as we discuss in the following section.

6.2 FUTURE WORKS

As we have seen, some observations do not lead to accurate predictions even after application of scale-space and of auxiliary methods to different observation and prediction

windows sizes.

It is important to have in mind that procedures for forecasting vary greatly in complexity, basic assumptions, and accuracy. A few are simple to use, but also provide little information; others yield substantial quantities of information, but also take more time and effort. In between these extremes are a variety of forecasting methodologies. Selection of the methodology depends on the importance of the forecast. There is no single technique widely agreed to be the best one [68], and, as such, many authors consider the forecasting process both an art and a science. It's an art because it is estimation in unknown situations, and the accuracy of the forecast will be due in some part to judgment and experience. But it's also a science as it can be seen as a step-by-step mathematical process that takes past history and uses it to predict future events.

In the light of that, we believe that although ARIMA and SVR have proven to be two powerful forecasting methods, we can increase the general accuracy of our approach by adding to our framework more prediction algorithms with different characteristics, with means to obtain good results in situations uncovered by other methods. Learning is an ill-posed problem, and with finite data, each algorithm converges to a different solution and fails under different circumstances. The performance of a learner may be fine-tuned to get the highest possible accuracy on a validation set, but this fine-tuning is a complex task and still there are instances on which even the best learner is not accurate enough [43]. The idea is that there may be another learner that is accurate on these. By suitably combining multiple learners then, accuracy can be improved. Performing this study and considering that we may execute predictions in parallel, would be a great addition to our approach.

It is important, though, to remember to treat forecasts as what they really are: educated guesses that need constant refinement. Being able to comprehend the limitations of a method but still taking advantage of its strength is crucial in developing stronger models by the combination of different techniques.

However, dynamic environments deal with a wide range of applications with different QoS requirements. The intrinsic differences among these workloads further make the resource provisioning a challenging task. It is quite difficult to match the capacities of various services without the knowledge of their behaviors. As such, effective capacity planning requires an accurate understanding of application behavior [54]. Thus, we intend to classify the workloads we receive in oder to obtain groups with specific characteristics.

Such workload characterization will allow us to select a set of prediction strategies to act in each group, allowing executions to adaptively change to each group's specificities.

It is also our objective to incorporate to our solution other variables such as CPU, memory, response time, I/O, and perform multivariable analysis in order to be more efficient in detecting bottlenecks that lead to QoS decreasing.

Finally, it is our intention to integrate our approach to different provisioning modules and test it in the cloud environment, and to study other error metrics with means to come up with more meaningful analysis in our context.

BIBLIOGRAPHY

[1] J.J. Prevost, K. Nagothu, B. Kelley, and M. Jamshidi. Prediction of cloud data center networks loads using stochastic and neural models. In *SoSE*, pages 276 –281, 2011.

[2] Zhenhuan Gong, Xiaohui Gu, and John Wilkes. Press: Predictive elastic resource scaling for cloud systems. In *CNSM*, pages 9–16. IEEE, 2010.

[3] Wei Fang, ZhiHui Lu, Jie Wu, and ZhenYin Cao. Rpps: A novel resource prediction and provisioning scheme in cloud data center. In *SCC '12*, pages 609–616, 2012.

[4] Brian Guenter, Navendu Jain, and Charles Williams. Managing cost, performance, and reliability tradeoffs for energy-aware server provisioning. In *INFOCOM*, pages 1332–1340. IEEE, 2011.

[5] Juan M. Tirado, Daniel Higuero, Florin Isaila, and Jesú Carretero. Predictive Data Grouping and Placement for Cloud-Based Elastic Server Infrastructures. In *CCGrid*, pages 285–294, 2011.

[6] Pengcheng Xiong, Yun Chi, Shenghuo Zhu, Hyun Jin Moon, Calton Pu, and Hakan Hac. Intelligent management of virtualized resources for database systems in cloud environment. *Response*, 12(1):87–98, 2011.

[7] Sadeka Islam, Jacky Keung, Kevin Lee, and Anna Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Comp. Syst.*, 28(1):155–162, 2012.

[8] S. Sakr and A. Liu. Sla-based and consumer-centric dynamic provisioning for cloud databases. In *IEEE 5th International Conference on Cloud Computing (CLOUD), 2012*, pages 360 –367, june 2012.

[9] Flavio R. C. Sousa and Javam C. Machado. Towards elastic multi-tenant database replication with quality of service. In *UCC '12*, 2012.

[10] S Nandi. Hybrid process modeling and optimization strategies integrating neural networks/support vector regression and genetic algorithms: study of benzene isopropylation on hbeta catalyst. *Chemical Engineering Journal*, 97(2-3):115–129, 2004.

[11] John Allspaw. *The Art of Capacity Planning: Scaling Web Resources*. O'Reilly Media, Inc., 2008.

[12] Saurabh Kumar Garg, Srinivasa K. Gopalaiyengar, and Rajkumar Buyya. Sla-based resource provisioning for heterogeneous workloads in a virtualized cloud datacenter. In *ICA3PP (1)*, pages 371–384, 2011.

[13] Smita Vijayakumar, Qian Zhu, and Gagan Agrawal. Dynamic resource provisioning for data streaming applications in a cloud environment. In *Proceedings of the 2010 IEEE Second International Conference on Cloud Computing Technology and Science*, CLOUDCOM '10, pages 441–448, Washington, DC, USA, 2010. IEEE Computer Society.

[14] Accenture. Cloud computing and sustainability: the environmental benefits of moving to the cloud, november 2010.

[15] S. Ghanbari. *Cost-aware Dynamic Provisioning for Performance and Power Management*. Canadian theses. University of Toronto, 2008.

[16] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *Comput. Netw.*, 53(17):2923–2938, 2009.

[17] N. Roy, A. Dubey, and A. Gokhale. Efficient autoscaling in the cloud using predictive models for workload forecasting. In *IEEE International Conference on Cloud Computing (CLOUD), 2011*, pages 500 –507, july 2011.

[18] Emmanuel Cecchet, Rahul Singh, Upendra Sharma, and Prashant Shenoy. Dolly: virtualization-driven database provisioning for the cloud. In *Proceedings of the 7th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, VEE '11, pages 51–62, New York, NY, USA, 2011. ACM.

[19] Andrew P. Witkin. Scale-space filtering. In *Proceedings of the Eighth international joint conference on Artificial intelligence - Volume 2*, IJCAI'83, pages 1019–1022, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.

[20] M. Sato, T. Wada, and H. Kawarada. A hierarchical representation of random waveforms by scale-space filtering. In *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP '87.*, volume 12, pages 273 – 276, apr 1987.

[21] Tevfik Metin Sezgin and Randall Davis. Scale-space based feature point detection for digital ink. In *ACM SIGGRAPH 2006 Courses*, SIGGRAPH '06, New York, NY, USA, 2006. ACM.

[22] Vladimir N. Vapnik. *Statistical Learning Theory.* Wiley-Interscience, 1998.

[23] G.E.P. Box, G.M. Jenkins, and G.C. Reinsel. *Time Series Analysis: Forecasting and Control.* Wiley Series in Probability and Statistics. Wiley, 2008.

[24] Tony Lindeberg. *Scale-Space Theory in Computer Vision.* Kluwer Academic Publishers, Norwell, MA, USA, 1994.

[25] Lide Wu and Zhaohui Xie. Scaling theorems for zero-crossings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):46 –54, jan 1990.

[26] Ronald L. Allen and Duncan Mills. *Signal Analysis: Time, Frequency, Scale, and Structure.* Wiley-IEEE Press, 2004.

[27] H. Xiong, Gaurav Pandey, M. Steinbach, and Vipin Kumar. Enhancing data analysis with noise removal. *IEEE Transactions on Knowledge and Data Engineering*, 18(3):304 – 319, march 2006.

[28] G.M. Phillips. *Interpolation and Approximation by Polynomials.* CMS Books in Mathematics. Springer, 2011.

[29] Prashant Pant. Data mining and predictive analytics. `http://www.information-management.com/newsletters/analytics-data-mining-predictive-ROI-value-10021815-1.html`, January 2012.

[30] Mausumi Debahuti Mishra, Asit Kumar Das and Sashikala Mishra. Predictive data mining: Promising future and applications. *International Journal of Computer Communication and Technology (IJCCT)*, 2(1):20–28, 2010.

[31] Data Warehousing Investment. Extending the value of your data warehousing investment. *Advances*, 5:1–36, 2007.

[32] Georg Dorffner. Neural Networks for Time Series Processing. *Neural Network World*, 6(4):447–468, 1996.

[33] Stefan Kok. Liquid state machine optimization, 2007.

[34] Prashant Pant. Time series analysis and forecasting with weka. `http://wiki.pentaho.com/display/DATAMINING/Time+Series+Analysis+and+Forecasting+with+Weka`.

[35] P.J. Brockwell and R.A. Davis. *Time Series: Theory and Methods*. Springer Series in Statistics. Springer, 2009.

[36] G. Kitagawa. *Introduction to Time Series Modeling*. Monographs on Statistics and Applied Probability. Chapman & Hall/CRC, 2009.

[37] Josefina L. Herrera. *Time Series Prediction Using Inductive Reasoning Techniques*. PhD thesis, Instituto de Organizacion y Control de Sistemas Industriales, March 1999.

[38] NIST/SEMATECH. e-handbook of statistical methods. `http://www.itl.nist.gov/div898/handbook/`, April 2012.

[39] A. Lendasse, V. Wertz, G. Simon, and M. Verleysen. Fast bootstrap applied to ls-svm for long term prediction of time series. In *IEEE International Joint Conference on Neural Networks, 2004. Proceedings.*, volume 1, pages 4 vol. (xlvii+3302), july 2004.

[40] E.A. Plummer. *Time series forecasting with feed-forward neural networks: guidelines and limitations*. University of Wyoming, 2000.

[41] G. E. P. Box and G. M. Jenkins. *Time Series Analysis, Forecasting, and Control*. Holden-Day, San Francisco, 1976.

[42] O.D. Anderson. *Time Series Analysis and Forecasting: The Box-Jenkins Approach*. Time series and forecasting. Butter Worths, 1976.

[43] E. Alpaydin. *Introduction to machine learning*. Adaptive Computation and Machine Learning. MIT Press, 2010.

[44] Guoqiang Zhang, B. Eddy Patuwo, and Michael Y. Hu. Forecasting with artificial neural networks:: The state of the art. *International Journal of Forecasting*, 14(1):35–62, March 1998.

[45] Bernhard Schölkopf, Alex J. Smola, Robert C. Williamson, and Peter L. Bartlett. New support vector algorithms. *Neural Comput.*, 12(5):1207–1245, May 2000.

[46] Wang Ling, Fu DongMei, and Li Qing. Samples selection based on svr for prediction of steel mechanical property. In *Second International Conference on Intelligent System Design and Engineering Application (ISDEA), 2012*, pages 909 –912, jan. 2012.

[47] Alex J. Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14(3):199–222, August 2004.

[48] S. Mukherjee, E. Osuna, and F. Girosi. Nonlinear prediction of chaotic time series using support vector machines. In *Neural Networks for Signal Processing [1997] VII. Proceedings of the 1997 IEEE Workshop*, pages 511 –520, sep 1997.

[49] Wenjian Wang and Zongben Xu. A heuristic training for support vector regression. *Neurocomputing*, 61:259–275, 2004.

[50] A. Shabri R. Samsudin and P. Saad. A comparison of time series forecasting using support vector machine and artificial neural network model. *Journal of Applied Sciences*, 10:950–958, 2010.

[51] ÃSTÃN B.; MELSSEN W. J.; BUYDENS L. M. C. Facilitating the application of support vector regression by using a universal pearson vii function based kernel. *Chemometrics and intelligent laboratory systems*, 81(1):29–40, 2006.

[52] Robert F. Nau. What's the bottom line? how to compare models.

[53] David J. Lilja. *Measuring Computer Performance: A Practitioner's Guide*. Cambridge University Press, 2005.

[54] Nilabja Roy, Abhishek Dubey, Aniruddha Gokhale, and Larry Dowdy. A capacity planning process for performance assurance of component-based distributed systems. In *International Conference on Performance Engineering (ICPE) 2011 - A Joint Meeting of WOSP/SIPEW*, pages 259–270, Karlsruhe, Germany, March 2011 2011. ACM, ACM.

[55] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal, and Timothy Wood. Agile dynamic provisioning of multi-tier internet applications. *ACM Trans. Auton. Adapt. Syst.*, 3(1):1:1–1:39, March 2008.

[56] Christophe Taton, Sara Bouchenak, Noël De Palma, and Daniel Hagimont. Self-Optimization of Internet Services with Dynamic Resource Provisioning. Research Report RR-6575, INRIA, 2008.

[57] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[58] Moriera W and Warnes GR. Rpy, a robust python interface to the r programming language. Available at `http://rpy.sourceforge.net/`.

[59] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0.

[60] 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.

[61] André d'Almeida Monteiro. Interest rate curve estimation: a financial application for support vector regression. Technical report, Princeton University, 2004.

[62] Chen-Chia Chuang, Shun-Feng Su, Jin-Tsong Jeng, and Chih-Ching Hsiao. Robust support vector regression networks for function approximation with outliers. *IEEE Transactions on Neural Networks*, 13(6):1322 – 1330, nov 2002.

[63] Gilson Medeiros de Oliveira Junior. Máquina de vetores suporte: estudo e análise de parâmetros para otimização de resultado, 2010.

[64] Martin Arlitt and Tai Jin. Workload characterization of the 1998 world cup web site. Technical report, IEEE Network, 1999.

[65] Patrick J.F. Groenen G.Nalbantov and Jan C. Bioch. Support vector regression basics. `http://www.ectrie.nl/met/index.php?view=publications&year=13&edition=1&lang=en`, 2010.

[66] Song Xiaoshan, Jiang Xiaoyu, Han Chongzhao, and Luo Jianhua. Inter-class distance based kernel parameter evaluating method for rbf-svm. In *International Conference on Digital Manufacturing and Automation (ICDMA), 2010*, volume 1, pages 853 –858, dec. 2010.

[67] Mikhail Kanevski Alexei Pozdnoukhov. Multi-scale support vector regression for hot spot detection and modeling, 2006.

[68] Xiaoqiao Meng, Canturk Isci, Jeffrey Kephart, Li Zhang, Eric Bouillet, and Dimitrios Pendarakis. Efficient resource provisioning in compute clouds via vm multiplexing. In *Proceedings of the 7th international conference on Autonomic computing*, ICAC '10, pages 11–20, New York, NY, USA, 2010. ACM.