Support Vector Regression for Link Load Prediction

Paola Bermolen and Dario Rossi – ENST Télécom Paris (France)
firstname.lastname@enst.fr

Abstract— **From weather to networks, forecasting techniques constitute an interesting challenge: rather than giving a faithful description of the current reality, as a looking glass would do, researchers seek crystal-ball models to speculate on the future.**

This work is the first to explore the use of Support Vector Machines (SVM) for the purpose of link load forecast. SVMs work well in many learning situations, because they generalize to unseen data, and are amenable to continuous and adaptive on-line learning, an extremely desirable property in network environments. Motivated by the encouraging results recently gathered by means of SVM on other networking applications, our aim is to enlighten whether SVM is also successful for the prediction of network links load at short time scales.

We consider the problem of link load forecast based only on its past measurements, which is referred to as "embedded process" regression in the SVM lingo, and adopt a hands-on approach to evaluate SVM performance. Our finding is that while SVM robustness is more than satisfactory, accuracy results are just close to be tempting, but not enough to convince. Based on the result of our experimental campaign, we then speculate on what directions can be undertaken to ameliorate the performance of SVM in this context.

I. Introduction

As network services and Internet application evolve, the network traffic is becoming increasingly complex and dynamic. On the one hand, transport networks are challenged by the current convergence trend of voice/video/data services on an all-IP network, and by the fact that user-mobility will likely translate into service-mobility as well. On the other hand, the explosion of Internet gaming, telephony and television applications implies that we may be forced to re-think what we mean by "data" traffic: the widespread usage of application layer overlays, directly translates into a much higher variability of the data traffic injected into the network.

Yet, despite it is widely accepted that the IP best effort paradigm is inadequate for the development of an effective multi-service next generation Internet built on sound commercial principles, the take up of QoS architectures (e.g., IntServ and DiffServ) has been deceiving with a very limited deployment. A *self-managing* network architecture, on which the management system delegates to the network control plane part of his tasks, is the key to cope with both the increasing traffic dynamism and the need for cost-effective solutions. The fundamental building blocks on top of which such an architecture can be built are represented by traffic classification, measurement and analysis: different actions have to be triggered depending on the type of traffic by a measurement-based decision process, aimed at counter the phenomenon that triggered the reaction in the first place. Generally speaking, decisions can be based on past snapshots of the system status, or may forecast the likely system evolution as well.

In this paper, we focus on the evaluation of supervised forecast techniques based on Support Vector Machines (SVM), a set of related methods for classification and regression, introduced in the early nineties [1], that are grounded in the framework of statistical learning theory. Basically, SVMs use training data to build a forecast model which works well in many learning situations because it generalizes to unseen data and is amenable to continuous and adaptive on-line learning, an extremely desirable property in network environments. Initially bound to the optical character recognition context, the use of SVMs rapidly spread to other fields, including time series prediction [2] and, more recently, networking [3], [4].

Motivated by such encouraging results, we focus for the time being on link load forecast based only on past measurements, which is known as "embedded process" [2]. This problem is of great interest in networking for both capacity planning and self-management application (e.g., as bandwidth provisioning, admission control, backpressure mechanism). Though the SVM approach is well fit to longer time-scales as well, which are more of a concern for capacity planning, in this paper we focus on the estimation of load variation at short time scales, envisaging thus the self-management as context for its application. Adopting a hands-on approach to the SVM regression, we evaluate the effectiveness of SVM for link load forecast by exploring a rather extensive parameter space. Our aim is twofold: first, we want to evaluate the SVM accuracy and robustness and, second, we want to provide useful insights on the tuning of the SVM parameters, an aspect not always clear in previous work. Our results show that SVM based models are rather robust to parameter variation, which constitutes an undoubted positive aspect. However, despite a good accordance with the actual data, the gain over simple estimation technique is not entirely satisfactory: thus, the question which we try to answer is what can be done to ameliorate the performance of SVM in this context.

II. Related Work

Most of the work related to network load forecast is based on the analysis of time series properties. On this context, a number of very different models [5], [6], [7] have been proposed, ranging from very simple to very complex ones. However, the majority of these approaches relies on specific assumptions and underlying models for the network traffic (e.g., they are tailored to capture Long Range Dependence (LRD) [8] at short and long timescales, etc.). A first drawback is that such models will no longer be applicable if the assumption no longer holds (e.g., considering other timescales). A second drawback is that such models usually rely, beside fitting on the actual data, on the precise estimation of some traffic parameters, whose computation can be a very intensive and delicate task (e.g., Hurst parameter of the arrival time series).

Rather, as in [9], [10], we prefer to focus on techniques that, avoiding to make any assumptions on the phenomenon under observation, allow for intrinsically more robust and flexible prediction. A simple local Gaussian predictor is provided in [9] as a core tool to guide the bandwidth provisioning in the hose model: interestingly, the model is able (but not *forced*) to embed assumptions on the LRD properties of the traffic, by an appropriate tuning of the parameters. TCP throughput prediction is the object of [10], where authors compare formula-based versus history-based prediction schemes, showing that even simple moving-average models are able to yield satisfactory results.

The technique considered in this paper fall into the SV regression class [11]. Partly due to its relatively short existence, the network research community has only very recently [3], [4] started to grasp SVM huge potential. TCP throughput prediction on a given path is the object of [3]: authors show that in realistic scenarios, using active probes to collect the necessary "features" (such as queue size and available bandwidth), the use of SVM can bring a 60% improvement over standard technique (predictions error lower than 10% nearly 50% of the times) with a much lower impact on end-to-end paths. Authors in [4] focus instead on the prediction of the latency toward an unknown IP address, based on the latency knowledge toward other IP addresses: SVM regression on a large randomly collected data set of 30,000 (IP,latency) couples, yield a mean prediction error of 30 ms (25 ms) using only 6% (20%) of the samples for training.

In the context of SVM regression, the problem of forecast of future values of a series based only on previous observation of the same phenomenon is done through an "embedding process" [2]. However, its application has usually targeted domains other than the networking context, and the series that SVM has been fed with up to now are very much different from those representing the packet arrival process at a router queue: thus, our aim is to test whether SVM can prove to be a useful tool also for link load forecast.

III. Support Vector Machines

A. Overview

Suppose that we are given a *training* set $\{(x_1, y_1), \ldots, (x_S, y_S)\} \subset R^d \times R$, where S is the training set size and R^d is the space of the input features x_i and y_i is the phenomenon under investigation. In ϵ-SV regression [12] the goal is to find a function $f(x)$ whose deviation from each target y_i is at most ϵ, and at the same time, is as "flat" as possible. For the sake of clarity, we first consider the linear case i.e. $f : \mathbb{R}^d \to \mathbb{R}$,

$$f(x) = \langle w, x\rangle + b, \qquad \text{with} \qquad x \in \mathbb{R}^d, b \in \mathbb{R} \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denote the dot product in R^d. Flatness in the case of (1) can be ensured by minimizing the norm $\|w\|^2$ leading to the following convex optimization problem:

$$\begin{array}{l} \min \frac{1}{2}\|w\|^2 + C \sum\limits_{i=1}^{S} (\xi_i + \xi_i^*) \\ \text{s.t.} \left\{ \begin{array}{ll} y_i - \langle w, x_i\rangle - b & \leq \epsilon + \xi_i^* \\ \langle w, x_i\rangle + b - y_i & \leq \epsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq 0 \end{array} \right. \end{array} \tag{2}$$

In the above formulation, slack variables $\xi_i + \xi_i^*$ are included to cope with otherwise infeasible constraint of the optimization problem, whereas the constant $C > 0$ determines the trade off between the flatness of f and deviations from the target greater than ϵ. This formulation is equivalent to using the ϵ-insensitive loss function (3) in the theory of error risk minimization with regularization:

$$L(\xi) = \left\{ \begin{array}{ll} 0, & \text{if } |\xi| \leq \varepsilon \\ |\xi| - \varepsilon, & \text{otherwise} \end{array} \right. \tag{3}$$

Thus, a first SVM peculiarity is rooted in the risk minimization theory [12] – where, given an i.i.d. training set, the aim is to find a function f that minimizes an empirical risk based on a loss function. Moreover, we stress that the SVM problem can be seen as an *extension* of more traditional regression techniques: for instace, when $L(\xi) = |\xi|^2$ is used as loss function, then we fall into the case of a minimum square error regression problem.

B. Support Vectors

The training problem (2) can be solved more easily in its *dual* formulation, obtained by constructing a Lagrange function from the objective and the constraints: indeed, the dual formulation yield a quadratic optimization problem with a unique solution, thus avoiding the local minimum problem. The solution of (2) yield to a model which can be used to forecast future values. The function $f(x)$ can be written as a linear combination of the training data, the Lagrange multipliers α_i, α_i^*, and a constant term b whose computation stems from the Karush-Kuhn-Tucker (KKT) condition:

$$f(x) = \sum_{i=1}^{n} (\alpha_i - \alpha_i^*)\langle x_i, x\rangle + b \tag{4}$$

The Lagrange multipliers verifies the constraints $\sum_{i=1}^{n}(\alpha_i - \alpha_i^*) = 0$ and $\alpha_i, \alpha_i^* \in [0, C]$. The KKT conditions also implies that if $\alpha_i, \alpha_i^* \neq C$ and $|f(x_i) - y_i| < \varepsilon$, then α_i, α_i^* must be zero. Intuitively, as errors lower than ϵ are tolerated, training-data lying inside the so called "ε-tube" will not contribute to the problem solution (nor to its cost). In other words, not all x_i are needed to calculate $f(x)$, but only the $S_V < S$ training points x_i whose $\alpha_i, \alpha_i^* \neq 0$, which are referred to as *support vectors*.

C. Kernel Trick

The dual formulation also provides the key to its non linear extension, which is a second very important peculiarity of SVM, by means of the so called "kernel trick" [13]. The idea is to map the input data into a higher-dimensional space $\mathcal{F}$ by a function $\phi : \mathbb{R}^d \to \mathcal{F}$. Then, a linear regression in this new space $\mathcal{F}$ is equivalent to a non-linear regression in the original space. Observing that $\langle x, x' \rangle = \langle \phi(x), \phi(x') \rangle$ and that the SVR algorithm only depends on the dot product between the data, it is sufficient to know how to compute the function $k(x, x') = \langle \phi(x), \phi(x') \rangle$ rather than knowing the mapping function $\phi(x)$ explicitly. More formally, a function $k(x, x')$ is called a *kernel* if it corresponds to a dot product in some feature space $\mathcal{F}$. By restating the optimization problem in terms of the kernel, it follows that the function f can be written as:

$$f(x) = \sum_{i=1}^{S_V} (\alpha_i - \alpha_i^*) k(x_i, x) + b \tag{5}$$

Interestingly, it can be shown [14] that using a linear kernel is equivalent to perform an ARMA regression [5], with the advantages of a simpler fitting procedure and robustness in presence of outliers. In this work, we use a radial basis kernel (to which corresponds an infinite dimensional mapping space), due to the good performance shown in both the specific time series prediction [14] and more general network [3], [4] contexts:

$$k(x, x') = e^{-\gamma \|x - x'\|^2} \tag{6}$$

D. Time series prediction

In the context of SVM regression, there is no a priori restriction on the type and number of features. There is, however, one special case: whenever future values of a phenomenon are to be predicted based only on past observation of the same phenomenon. More formally, we select the SVM features as done in the embedding process [2] context:

$$\{\lambda(t), \lambda(t - \tau), \ldots, \lambda(t - (d - 1)\tau)\} \tag{7}$$

For what concerns our link load forecast application, $\lambda(t)$ represents the average traffic load measured in the time interval $[t - \tau, t)$. More on details, we quantize the time in multiples of τ: denoting for the ease of notation by $\lambda(t) = \lambda(k\tau) = \lambda_k$ the average traffic load measured in the time interval $[(k - 1)\tau, k\tau)$, the features (7) take the form:

$$\{\lambda_k, \lambda_{k-1}, \ldots, \lambda_{k-(d-1)}\} \tag{8}$$

In general terms, τ can be though as the timescale of the observation, the dimension d as the minimum number of state variables required to describe the system and their product $d\tau$ as the average system memory length.

Usually the above parameters are obtained by running (computationally intensive) geometric heuristics on input data. Indeed, our aim being to build a robust engine for the estimation of traffic load on *arbitrary* timescales, we would rather have no constraint on the selection of the operation point (d, τ) – or at least on the operation timescale τ. Therefore, we prefer to cross-check the impact of the embedding parameters choice *a posteriori*, based on the empirical results of the regression, rather than developing an ad-hoc methods to extract them from the data we consider: as a side effect of this choice, we will extend the empirical evaluation of the SVM engine robustness to a wider range of parameters.

IV. Experimental results

SVM embedded process regression is affected by many parameters, pertaining to two different semantic areas: a first set is related to the SVM regression:

- the training size S and the smoothing factor C of (2);
- the tolerance ϵ of the loss function (3);
- the parameter γ of the kernel function (6);

whereas the second set is related to the embedded process:

- the timescale τ and dimension d of (7);

However, prior to inspect the impact of the above parameters in the effectiveness of SVM for link load forecast, we need to provide details on the input data, and to overview the unsupervised forecast methods that we compare SVM with.

A. History-Based Forecast Methods

The one-step d-order Moving Average (d-MA) predictor can be defined as:

$$\hat{x}_{i+1} = \frac{1}{d} \sum_{k=i-d+1}^{i} x_k \tag{9}$$

as a general remark, if n is too small the predictor cannot smooth out the noise in the underlying measurements, whereas a too large value of n makes it slow to adapt to non-stationarity properties of the data. The predictor (9) is the simplest among the unsupervised forecast methods. Yet, in a slightly different context, [10] shown that, despite its simplicity, d-MA it is able to provide accurate results provided that it copes with the two major error sources: Level-Shift and Outliers. We implement the LSO heuristics as in [10], to which we refer the reader for further details, and denote with d-LSO the corresponding predictor: basically, outliers are just ignored, whereas the detection of level-shift triggers a filter restart

It can be argued that the extreme simplicity of d-LSO will unfairly bias the comparison. However, we believe that a more thorough comparison with other, more sophisticated, forecast techniques will be necessary only whether the SVM engine is able to provide a significant improvement over simple techniques, so to justify the burden of its deployment.

Moreover, we point out that we deliberately avoid comparison with other, more sophisticated, unsupervised predictor such as the Local Gaussian Predictor (LGP) of [9], as we experimentally verified that in the short timescales considered in this paper they *systematically* overestimate the incoming traffic rate. While in some application this is actually a desirable feature (e.g., as in VPN bandwidth provisioning, where over-provisioning translate in fewer losses and thus in a greater service QoS), in many other context it is not (e.g., when admission control is performed, over-estimating the incoming load unjustifiably increases the flow reject ratio).

For the time being, out aim is thus to evaluate the accuracy of the prediction, without introducing any a priori "preference" toward a specific type of error (i.e., over- or under- estimation), as this would implicitly mean to focus on a specific application (i.e., bandwidth provisioning and admission control respectively).

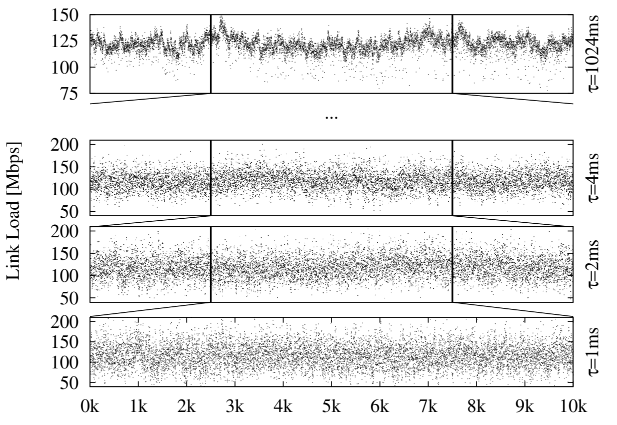

Fig. 1. Subset choice at different timescales (daily dataset)

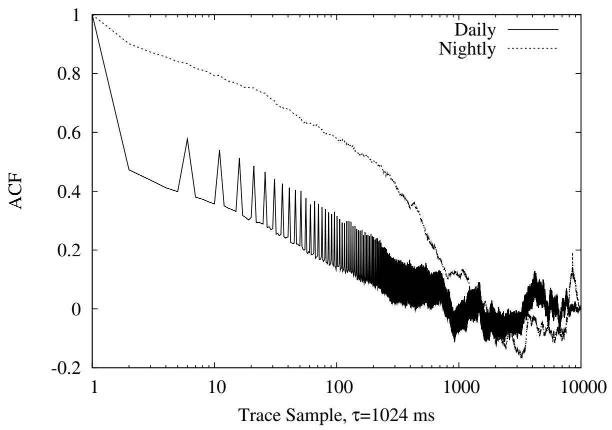

Fig. 2. Autocorrelation function of the Daily and Nightly trace datasets

TABLE I
INPUT TRACE: LINK LOAD AT DIFFERENT TIMESCALES

	i		0	2	4	6	8	10
D	μ	[Mbps]	116.4	118.3	120.4	119.5	118.9	121.3
	σ	[Mbps]	28.9	20.7	16.4	12.7	10.3	7.9
N	μ	[Mbps]	60.8	62.4	68.1	69.6	68.2	66.2
	σ	[Mbps]	32.6	21.2	15.4	9.6	6.8	8.7

B. Input Data

To benchmark the effectiveness of the SVM for traffic load forecast, we collect the traffic at the POP of a major Italian ISP. This dataset is very peculiar, since it refers to an innovative ISP which is providing end users (residential, SOHO or large companies) with data, voice and video over IP by means of either an ADSL or a FTTH link (no PSTN link is offered). Traffic is therefore composed of data transfers over TCP, VoIP and VideoIP traffic over RTP/UDP. Moreover, as users make extensive use of P2P applications, VPN services, etc., the resulting traffic mix is therefore very heterogeneous. We sniffed a one-day long trace on Monday the 15th of May 2007, and consider a single traffic direction. We then extracted several $N = 10000$ second long (about 2h45) subsets of the trace: here, we report results referring to a daily-busy subset and nightly-idle periods. In the daily subset (D), average link load is 122.5 Mbps whereas in the nightly subset (N), average load was 65.4 Mbps. For each subset, we consider different timescales $\tau_i = 2^i$ ms with $i \in [0, 10]$, and for the sake of brevity, in the description we approximate $\tau_{10} = 1024$ ms with $\tau = 1$ s. We then compute the average arrival rate, building N elements long datasets: details relative to the mean and standard deviation of the load at different timescales are reported in Tab. I for even values of i: it can be clearly seen that the lower the timescale, the higher the load variance. The partitioning criterion for different timescales is depicted in Fig. 1: the subset for the (k-1)-th timescale corresponds to the central portion of the k-th one.

The daily and nightly datasets present an interesting difference. Fig. 2 depicts the autocorrelation function of the load at timescale $\tau = 1$ s: the peak of the busy trace exhibit a periodic fluctuation on the range of 5 s (and multiples of 5 s) which is absent in the nightly trace. Clearly, this dependence will affect any 5-lag samples when $\tau = 1s$ (i.e., λ_k and λ_{k-5}) and more generally any two samples that are $5\tau_i$ seconds apart.

C. SVM Parameter Tuning

To tune the performance of the SVM, we start by selecting the combination $(C^*, \epsilon^*, \gamma^*)$ of SVM parameters that minimize the root square mean error (RSME) of the prediction, which assess the quality of the estimator in terms of its variation and unbiasedness, where RMSE $= \sqrt{\sum_i^n (y_i - \hat{y}_i)^2/n}$. Results shown in this section were gathered using the JMySVM implementation of the Rapidminer tool [15].

Fixing $d = 5$ and $\tau = 1$ s, we selected 20% of the dataset at random for training the SVM, and tested the prediction accuracy for the remaining 80% of the set. We sampled 10 values for each of the SVM parameters, for a total of 1000 combinations. To select the boundary of the parameter space to be explored, we apply the following reasoning. A prescription for the regularization parameter C follows from (5): if we consider that, $|\alpha_i - \alpha_i^*| \leq C$ and $|k(x_i, x_j)| \leq 1$, we have that $|f(x)| \leq S_V C$, which yield to $C \leq |f(x)|/S_V$. While for $|f(x)|$ a reasonable choice is $\max(|y + 3\sigma_y|, |y - 3\sigma_y|)$ (to avoid outliers influence), since the number of support vector cannot be known a priori we consider the boundary cases where either i) all training data are support vectors $S_V = S$, or ii) only a single data point is a support vector $S_V = 1$. About ϵ, it is suggested in [16] that it should be proportional to the input noise level: however, as the definition of link load "noise" is questionable, we are forced to resort to an empirical choice – and we proceed similarly for γ.

The grid optimization process for the busy trace, which yielded to a minimum RMSE=5.88 when $(C^*, \epsilon^*, \gamma^*) = (30, 5, 0.05)$, is depicted in Fig. 3. The picture show the whole (C, ϵ, γ) parameter set explored, conditioning over each of the three parameters. For the sake of clarity, let consider the leftmost plot of Fig. 3, whose x-axis represent the C parameter values. Each point in the plot represent a single experiment, and for any value of the C parameter on the x-axis, 100 points

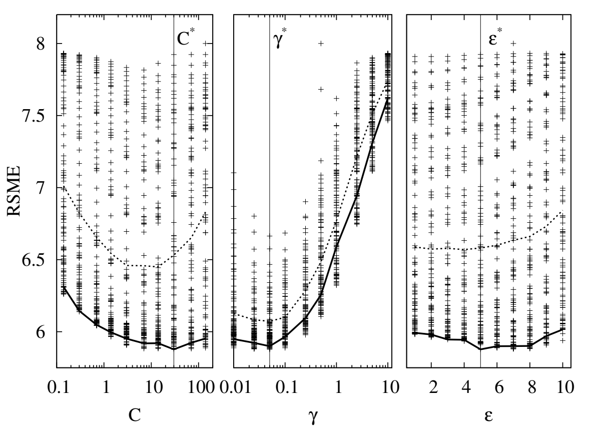

Fig. 3. Grid optimization process for the selection of $(C^*, \epsilon^*, \gamma^*)$

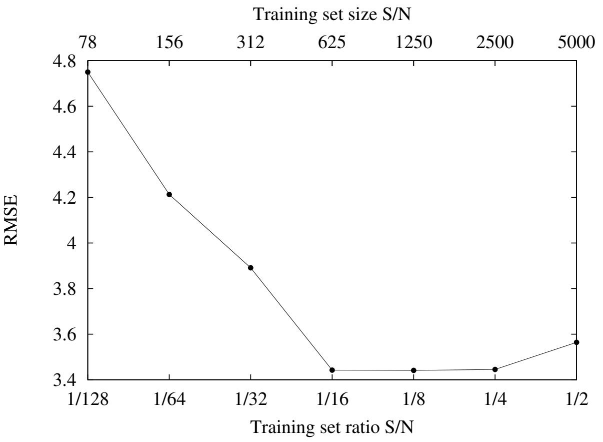

Fig. 4. Impact of training size on prediction accuracy

are plotted that correspond to the 100 combination of the other two parameters ϵ and γ. The plot also reports some reference lines: the vertical thin line refers to the best value C^*; the dotted thick line represents, for any given C, the average of the RMSE achieved for the 100 possible combination of ϵ and γ; the solid thick line refers instead to the RMSE achieved as a function of C when the other parameters are set to their best values (i.e., ϵ^* and γ^*). From the Fig. 3, it can be gathered respectively that i) as the RMSE is convex in C, the value of C should be neither too big nor too small, ii) the prediction error exhibit a (roughly exponential) increase with γ value, and ii) that the impact of ϵ is less significant with respect to C and γ. A similar operation on the nightly trace yielded to a different best combination of parameters, namely $(25, 0.1, 0.01)$, that we will therefore use for the nightly subset. At the same time, the daily parameters $(C^*, \epsilon^*, \gamma^*)$ ranked 50th in the night trace, with an RMSE increase of 4%: thus, even in extremely different load conditions, the SVM prediction is rather robust to the parameter choice.

Next, we explore the impact of the training set size on the accuracy of the prediction. For different sizes S of the training set, we perform the so-called "cross validation": we build a training set with randomly chosen S data within the N dataset samples, and evaluate the prediction accuracy over the remaining $N-S$ samples. The process is repeated S times for each value of S, changing the training and evaluation set every time, and RMSE results are reported in Fig. 4 as a function of the training size S (top x-axis) and ratio S/N (bottom x-axis). Three regions are clearly distinguishable: whenever the number of samples is exiguous, the SVM is *under-trained* and prediction error is large; afterwards, SVM rapidly learns and the error quickly drops until the SVM is *over-trained* and the error slightly increases. In our situation, in the learning zone $S/N \in [1/16, 1/4]$ the RMSE stays constant, and no further learning improvement is observed for $S > 625$. This is symptomatic of a situation in which the SVM is not effectively learning the structure of the time series: intuitively, what happens is that a few samples are enough (provided that they span the range of values achieved by the time series) to learn the series *average*. Moreover, as the contribution of the b term of (5) is prominent and very close to the average of the time series, the support vector contribution is not as significant as it should be in a proper learning situation: the prediction will then fluctuate around the average, much as in the case of a simple moving average – thus, we cannot unfortunately expect significant accuracy improvements over d-MA. This is also reflected in the number of support vectors, which is always very close to the training size $S_V \simeq S$. Its clear that the number of S_V decreases if the parameter ϵ increases and this can be done without much cost in the RMSE. This fact also affect the forecast complexity, as the number of computations that have to be performed for each forecast operation depends on the number of support vectors.

D. Embedded Parameter Impact

In this section, we explore the impact of the embedding parameters d and τ in the prediction accuracy of SVM versus d-MA. As a first remark, results are different, though quantitatively very close: in other words, SVM does not exhibit a significant improvement over d-MA. Nevertheless, let us investigate more closely the impact of the number d of previous samples, which is depicted in Fig. 5 for both nightly (right) and daily (left) periods at the $\tau = 1\,\text{s}$ timescale. Considering the nightly period, it can be seen that, the knowledge of a few elements is useful for the d-MA prediction, as long as the number of previous observation is small; indeed, when $d > 5$ the d-MA filter is averaging not useful information worsening its accuracy. At the same time, while the d-LSO limit the error for high values of d, it may actually *worsen* the accuracy at low values of d (and is furthermore sensitive to its parameter tuning). Conversely, the SVM is naturally robust even to unreasonable choices of d, which means that the SVM its unable to learn from the new features but also that this new features does not degrade its performance. In the daily period case the d-MA resent much of the periodical fluctuation: intuitively, the error is minimum whenever the forecast is exactly a multiple of the periodical lag (indeed, for $d = 5$ samples, two of them thus 2/5 are correlated, while

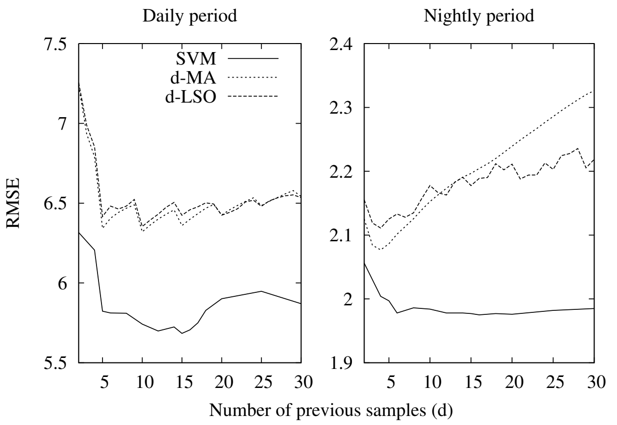

Fig. 5. Impact of the number d of previous samples on the forecast accuracy

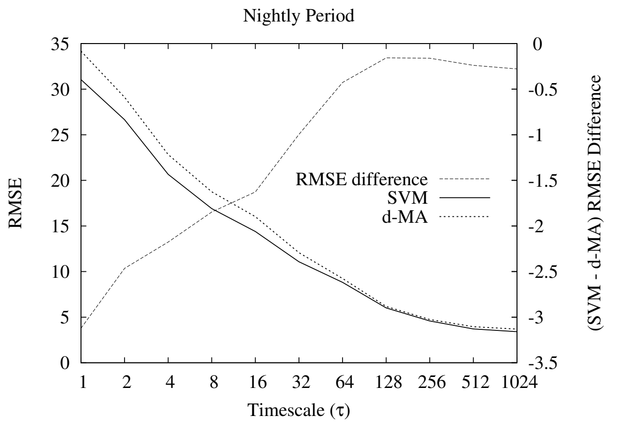

Fig. 6. Impact of the timescale τ on the forecast accuracy

for $d = 6$ only 2/6 are correlated, and so on) and grows in between two multiples. Interestingly, the LSO heuristic is not helpful in this case (as in this case the periodic fluctuation may be seen as a level shift and thus disregarded), while the robustness of SVM is preserved. Finally, setting $d = 5$ (thus, the best case for the d-MA filter but *not* for the SVM), we investigate whether SVM forecast bring any improvement at short time-scales. Fig. 6 depicts the prediction RMSE as a function of τ for the nightly period only: behavior of both predictors is similar, with short time scales constituting a stiffer scenario, as it can be expected in reason of the much higher traffic variability shown early Tab. I. The picture also report the RMSE difference of the two forecast techniques, from which it can be gathered that at shorter time-scales, SVM brings about a 10% improvement over d-MA.

V. Discussion and Conclusion

This paper is the first to explore the SVM regression mechanism for the purpose of link load forecast: using a hands-on approach, we tuned the SVM performance, comparing them with those of a simple moving average based technique. Our result shown that SVM based models are rather robust of SVM to parameter variation, which constitutes an undoubted positive aspect. At the same time, the gain over simple prediction methods is not enough to justify its deployment for link load prediction at short time scales. It is our belief that this work constitute a starting point for further investigation, whose directions are highlighted in the following. First, the gathered results raise a question about what can be done to ameliorate the performance of SVM at short time scales. Undoubtedly, the features considered by the embedded process regression alone, cannot provide the needed accuracy that SVM need in order to challenge more reliable methods such as [6], [7]. An open issue is whether and what manipulation of the time series (e.g., such as differentiae) can bring some enhancement, and also what are the additional features that could be worth feeding the SVM with (e.g., statistical properties of the time series, the number of active flows, the load breakdown by transport layer protocols, etc.). Another open issue is whether the use of other kernels (e.g., multilineal or other that can take into account the characteristics of the time series) could also ameliorate the SVM accuracy. Also, provided that SVM performance were good enough to justify its use, the knowledge of the forecast confidence interval would be extremely useful. As final remark, we point out that SVM may be more suitable on longer timescale: indeed, considering, e.g., time-of-day and day-of-week features it should be surprisingly easy to forecast periodic load fluctuations (e.g., lunch breaks and week-ends).

Acknowledgement

This work was funded by the Celtic project TIGER.

References

[1] B. E. Boser, et al. "A training algorithm for optimal margin classifiers". *ACM COLT'92,* pp. 144–152, Pittsburgh, PA, 1992
[2] K.-R. Muller, at al. "Predicting time series with support vector machines," In *Artificial Neural Networks* Springer, Berlin, 1999
[3] M. Mirza, et al. "A machine learning approach to TCP throughput prediction," In *ACM SIGMETRICS'07*, San Diego, CA, Jun. 2007,
[4] R. Beverly, et al. "SVM Learning of IP Address Structure for Latency Prediction," In *ACM MineNet'06*, Pisa, Italy, Sep. 2006
[5] P.J.Brockwell et al., "Introduction to time series and forecasting," Springer, Berlin, 1996
[6] J. Beran, "Statistics for Long-memory Processes," Chapman & Hall, London, 1994
[7] B. Krithikaivasan et al., "ARCH-based Traffic Forecasting and Dynamic Bandwidth Provisioning for Periodically Measured Nonstationary Traffic," IEEE/ACM Transaction on Networking, June 2007.
[8] W. E. Leland, et al. Similar Nature of Ethernet Traffic," *IEEE Transactions on Networking*, Vol. 2, No. 1, pp. 1–15, 1994.
[9] N.G.Duffield, et al. "Resource management with hoses: point to cloud services for virtual private network," *IEEE/ACM Transactions on Networking*, Vol.10, No. 5, Oct. 2002.
[10] Q. He et al. " On the predictability of large transfer TCP throughput", ACM SIGCOMM'05, Philadelphia, USA, Aug 2005
[11] A.J.Smola, et al. "A tutorial on support vector regression" In *Statistics and Computing*, Vol. 14, pp. 199–222, Kluwer Academic Pub., 2004.
[12] V.Vapnik, "The nature of statistical learning theory," Springer, NY, 1995.
[13] M. Aizerman et al. "Theoretical foundations of the potential function method in pattern recognition learning," *Automation and Remote Control,* Vol. 25, pp. 821-837, 1964
[14] S. Ruping at al. "Support Vector Machines and Learning about Time,", ICASSP'03, Hong Kong 2003.
[15] I. Mierwa et al. "YALE: rapid prototyping for complex data mining task," *ACM SIGKDD'06,* Philadelphia, PA, USA, 2006
[16] V. Cherkassky et al. "Practical Selection of SVM Parameters and Noise Estimation for SVM Regression," *Neural Networks*, Jan. 2004