Support vector regression for link load prediction

Paola Bermolen [a], Dario Rossi [a,*]

[a] TELECOM ParisTech, INFRES Department, 46 rue Barrault, Paris 75013, France

Keywords:
Network forecast
Traffic measurement
Supervised learning
Support vector machines

A B S T R A C T

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 explores the use of Support Vector Regression (SVR) for the purpose of link load forecast. SVR works well in many learning situations, because they generalize to unseen data, and are amenable to continuous and adaptive online learning – an extremely desirable property in network environments. Motivated by the encouraging results recently gathered by means of SVR on other networking applications, our aim is to enlighten whether SVR 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 SVR lingo, and adopt a hands-on approach to evaluate SVR performance. In more detail, we perform a sensitivity analysis of the parameters involved, assess the computational complexity for training and validation, dig into the correlation structure of the prediction errors and evaluate techniques to extend the forecasting horizon. Our finding is that accuracy results are close enough to be tempting, but not enough to be convincing. Yet, as SVR exhibit a number of advantages, such as good robustness and flexibility properties, furthermore at a price of a limited complexity, we then speculate on what directions can be undertaken to ameliorate its performance in this context.

1. Introduction

It is fairly well accepted that, as a result of network services and Internet applications evolution, network traffic is becoming increasingly complex. 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 telephony, television and gaming applications implies that we may be forced to re-think what we mean by ''data" traffic. Moreover, the widespread usage of application layer overlays directly translates into a much higher variability of the data traffic injected into the network.

In this paper, we question whether such variability can be efficiently forecasted, and if so, with what level of accuracy. The supervised prediction technique we selected is Support Vector Machines (SVM), a set of classification and regression techniques, introduced in the early nineties [1], that are grounded in the framework of statistical learning theory. Basically, Support Vector Regression (SVR) uses 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 online learning, an extremely desirable property in network environments. Initially bound to the optical character recognition context, the use of SVM rapidly spread to other fields, including time series prediction [2] and, more recently, networking [3–6]. Motivated by such encouraging results, we focus on link load forecast based only on past measurements, following an approach known as ''embedded process" [2]. This problem is of great interest in

* Corresponding author. Tel.: +33 145817563.
E-mail address: dario.rossi@telecom-paristech.fr (D. Rossi).

networking for both capacity planning and self-management application (e.g. bandwidth provisioning, admission control, trigger of backpressure mechanisms, etc.).

Though the SVM approach fits well to longer timescales 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: adopting a hands-on approach to the SVM regression, we evaluate the effectiveness of SVR for link load forecast by exploring a rather extensive parameter and design 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. We compare the performance with those achievable using Moving Average and Auto-Regressive models: our results show that, despite a good accordance with the actual data, the SVR gain achievable over simple prediction methods is not enough to justify its deployment for link load prediction at short time scales. Yet, we have to tribute SVR of a number of extremely positive aspects: for instance, SVR models are rather robust to parameter variation, and their computational complexity is far from being prohibitive, which makes them suitable for online prediction. Moreover, we experimentally verify that errors calculated over consecutive samples are independent and identically distributed, which allows the evaluation of confidence intervals. Finally, we also investigate methods to extend the forecast horizon using forecasted values as input for a new prediction: interestingly, this approach of recursive SVR may significantly extend the achievable forecast horizon, entailing only a very limited accuracy degradation.

The remainder of the paper is organized as follows. After discussing related work in Section 2, we briefly overview the Support Vector Regression theory in Section 3. In Section 4 we specify the methodology we follow in applying SVR models to link load forecast, as well as describing the other forecasting techniques that will be used for comparison purposes. A complete and extensive sensitivity analysis of SVR performance is reported in Section 5, whereas further details on the temporal evolution of the error, computational complexity considerations and result of recursive SVR are reported in Section 6. Finally, concluding remarks and future work are addressed in Section 7.

2. Related work

Most of the work related to network load forecast is based on the analysis of time series properties. In this context, a number of very different models [7–9] 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) [10] 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 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 [11,12], we prefer to focus on techniques that, avoid making any assumptions on the phenomenon under observation, allow for intrinsically more robust and flexible prediction. A simple local Gaussian predictor is provided in [11] 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 [12], where the authors compare formula-based versus history-based prediction schemes, showing that even simple moving-average models are able to yield satisfactory results (provided that one copes with major error sources).

The forecasting technique that we plan to evaluate in this paper falls in the class of Support Vector Machines (SVM) [13]: despite its relatively short existence, the literature of SVM is already full blown. At the same time, while the use of SVM for *classification* is relatively more popular in networking research, especially in the context of anomaly and intrusion detection [5,6], the use of SVM for *regression* is largely left unexplored. To the best of our knowledge, the only work that explores the use of SVR techniques in the networking field is [3,4]. TCP throughput prediction on a given path is the object of [3], where the forecast is based on a combination of path properties (such as queueing delays and available bandwidth) and on the performance of prior file transfers as well. Authors show that when the path properties are *precisely* known (e.g., when they are provided by an "oracle"), SVR is able to predict TCP throughput within 10% of the actual value in 90% of the cases – which represents nearly a 3-fold improvement in accuracy over prior history-based methods. Also, in more realistic scenarios and using less accurate measurements of path properties (e.g., gathered by means of active probes), the predictions can be made within 10% of the actual value nearly 50% of the time – which still represents a 60% improvement, with a furthermore much lower impact on end-to-end paths.

The authors of [4] focus instead on the prediction of the latency toward an unknown IP address, based on the latency knowledge toward other previously contacted IP addresses. Using as input features vectors of IP address bits (transformed into a 32 dimension input space, where each bit of the address corresponds to a different dimension), authors show that the estimation performance is within the 30% of the true value for approximately three-quarters of the latency prediction on a large Internet data set. More in detail, 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 [13], the problem of forecasting future values of a series based only on previous observation of the same phenomenon is known as an "embedding process" [2]. However, its application has usually targeted domains other than the networking context, and the series that SVR 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 SVR can prove to be a useful tool also for link load forecast.

3. Support vector machines overview

Support Vector Machines (SVM) [1] are a set of classification and regression techniques, that are a non-linear extension of the Generalized Portrait algorithm developed in Russia in the sixties. As previously outlined, SVM can be used for either classification or regression, to which we restrict our attention for the remainder of this work. In the following, we briefly overview the theory behind the use of SVM for function estimation, introducing at the same time the most relevant notions and parameters, with special attention to those parameters whose impact we investigate later on. In a sense, this overview is thus instrumental to the understanding of the performance evaluation section, but for a more thorough coverage of SVM we refer the reader to the excellent surveys [13–15].

3.1. Support vector regression

Suppose that we are given a *training* set $\{(x_1, y_1), \ldots, (x_S, y_S)\} \subset R^d \times R$, where R^d is the space of the input features x_i, and y_i is the phenomenon under investigation. In ϵ-SVR [16] the goal is to find a function $f(x)$ whose deviation from each target y_i is at most ϵ for all training data, and at the same time, is as "flat" as possible. For the sake of clarity, we first consider the linear case i.e. $f : \mathbb{R}^d \to \mathbb{R}$, such that

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

where $\langle\cdot,\cdot\rangle$ denotes 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:

$$\min \tfrac{1}{2}\|w\|^2 + C\sum_{i=1}^{S}(\xi_i + \xi_i^*), \quad \text{s.t.} \begin{cases} y_i - \langle w, x_i\rangle - b \leqslant \epsilon + \xi_i, \\ -y_i + \langle w, x_i\rangle + b \leqslant \epsilon + \xi_i^*, \\ \xi_i, \xi_i^* \geqslant 0. \end{cases} \tag{2}$$

In the above formulation, slack variables ξ_i and ξ_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 target greater than ϵ. Notice that this tradeoff makes SVM rather different from traditional error minimization problems, and very robust to outliers. The above formulation is equivalent to the use of the ϵ-insensitive loss function in the theory of error risk minimization with regularization [16]. Loss function (3) represents the fact that there is no loss (or cost) for deviations smaller than ϵ and that larger deviations will be linearly penalized:

$$L(\xi) = \begin{cases} 0 & \text{if } |\xi| \leqslant \varepsilon, \\ |\xi| - \varepsilon & \text{otherwise.} \end{cases} \tag{3}$$

Thus, a first peculiarity of SVR is its root in the risk minimization theory [16] – 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 SVR problem can be seen as an *extension* of more traditional regression techniques: for instance, when $L(\xi) = |\xi|^2$ is used as loss function, then we fall into the case of a minimum square error regression problem.

3.2. 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 yields a quadratic optimization problem with a unique solution, avoiding the problem of getting stuck in a local minimum. The solution of the dual problem yields the function $f(x)$, which can be written as a linear combination of the training data, the Lagrange multipliers α_i, α_i^*, and the constant term b, whose computation stems from the Karush–Kuhn–Tucker (KKT) conditions:

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

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

The Lagrange multipliers verify the constraints:

$$\sum_{i=1}^{S}(\alpha_i - \alpha_i^*) = 0 \quad \text{and} \quad 0 \leqslant \alpha_i, \alpha_i^* \leqslant C. \tag{6}$$

The KKT conditions also imply 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*. An important consequence of the above fact is that the problem complexity is *independent* of the dimension of the input space, but rather depends only in the number of support vectors.

3.3. Kernel trick

The dual formulation also provides the key to the non linear extension of SVR. This second, very important, peculiarity of SVM is known under the name of "kernel trick" [17]. 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 solution of the dual problem only requires the knowledge of the dot product, it is sufficient to know how to compute the *kernel* function $k(x, x') = \langle \phi(x), \phi(x')\rangle$ rather than knowing the mapping function $\phi(x)$ explicitly. This is a desirable feature, as actually applying the mapping into the higher-dimensional space may be computationally infeasible. More formally, a function $k(x, x')$ is called a *kernel* if it corresponds to a dot product in some feature space $\mathcal{F}$ and if it fulfills the Mercer's condition [18]. By restating the optimization problem in terms of the kernel, it follows that $f(x)$ can be written as

$$f(x) = \sum_{i=1}^{S_V} (\alpha_i - \alpha_i^*) k(x_i, x) + b. \quad (7)$$

In this work, we investigate the use a of a radial basis kernel, to which an infinite dimensional mapping space corresponds. This choice is motivated by the good performance shown in both the time series prediction [19] and more general network [3] contexts:

$$k(x, x') = e^{-\gamma \|x - x'\|^2}. \quad (8)$$

4. Forecast techniques

This section details how we apply the SVR framework to the load forecast problem, as well as introducing other techniques, such as Moving Average (MA) and Auto-Regressive (AR) models, whose performance will be compared with SVR's.

We stress that we deliberately avoid comparison with other unsupervised predictors such as the Local Gaussian Predictor (LGP) [11], as we experimentally verified that in the short timescales considered in this paper it systematically overestimates the incoming traffic rate. While in some applications 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 contexts it is not (e.g., when admission control is performed, over-estimating the incoming load unjustifiably increases the flow reject ratio): thus we prefer to avoid introducing any a priori bias.

We also point out that, in principle, an option to accommodate short timescales variability due to Internet traffic burstiness, could be to exploit the *a priori* knowledge of the scaling relations between rate and variance at different timescales as in [11]. However, as previously stated, we prefer to avoid any assumptions on the phenomenon under observation, and we will thus limit our comparison to MA and AR models.

4.1. Support vector regression

In the context of time series prediction by means of SVR, there is no a priori restriction on the type and number of input features. A known approach, which we adopt in this paper and explain in the following, is the so called "embedding process" [2]. Let be $\lambda(t)$ the traffic load measured in the time interval $[t - \tau, t]$. By quantizing the time in multiples of τ, we obtain a time series $\{\lambda_k\}_{k \in \mathbb{N}}$, where λ_k is the average traffic load measured in the interval $[(k-1)\tau, k\tau]$. The SVR embedding process then uses an arbitrary number d of past measurement of the above series in order to predict its future value. Thus, when given a d-dimensional input x, a trained SVR function returns as output $\hat{y} = f(x)$ a forecast of the target y, which in our case are:

$$x = (\lambda_{k-(d-1)}, \ldots, \lambda_{k-1}, \lambda_k) \quad \text{and} \quad y = \lambda_{k+1}. \quad (9)$$

In order to construct all the possible x input tuples, we use a sliding window of length d over the time series to construct all possible input/output pairs, obtaining the set $\{(x_i, y_i)\}_{i=1 \ldots L-d}$ where L is the time series length.

A subset of this set will be used as training set, i.e., to construct the SVR forecast function $f(x)$; then, the model accuracy will be evaluated over the *complement* of the training set, i.e., on unknown data. More precisely, the SVR training set is constructed by randomly selecting a few out of all the possible x input tuples: in other words, SVR training can be thought as realized by means of a jumping window. The impact of the training set selection will be thoroughly examined later.

In general terms, τ can be thought as the observation timescale, the dimension d as the minimum number of state variables required to describe the system and their product $d\tau$ corresponds to the average system memory length. The traditional "embedding process" assumes that the observed time series is the projection of a deterministic dynamic operating in a high-dimensional state space: in this case, the parameters d and τ can be obtained by running (rather computationally intensive) geometric heuristics on input data. However, we point out that the above assumptions do not apply directly to our context, as the network load dynamic is clearly not well represented by a deterministic process. Moreover, our aim is rather to build a robust engine for the online estimation of traffic load on *arbitrary* timescales: thus, we prefer to avoid constraints 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: as a side effect of this choice, we will extend the sensitivity analysis of SVR to a wider range of parameters.

4.2. Moving-average models

Given a time series $\{\lambda_k\}_{k \in \mathbb{N}}$, the one-step d-order Moving Average (d-MA) predictor can be defined as

$$\hat{\lambda}_{k+1} = \frac{1}{d} \sum_{i=k-d+1}^{k} \lambda_i. \quad (10)$$

As a general remark, if d is too small the predictor cannot smooth out the noise in the underlying measurements, whereas a too large value of d makes it slow to adapt to non-stationarity properties of the data. The predictor (10) is the simplest among the unsupervised forecast methods; yet, in a slightly different context, the authors of [12] showed that, despite its simplicity, d-MA is able to provide accurate results provided that it copes with the two major error sources: namely, Level-Shift and Outliers (LSO). We implement the LSO heuristics as in [12], and denote with d-LSO the corresponding predictor. Basically, outliers are just ignored, whereas the detection of level-shift triggers a filter restart. In more detail, considering a set of measurements $\{\lambda_1, \ldots, \lambda_d\}$, a sample λ_k is said to be an *outlier* whenever it differs from the median of the set by more than a relative difference ψ. Moreover, an increasing (decreasing) *level-shift* is detected in correspondence of λ_k whenever the following conditions jointly holds:

- the measurements $\{\lambda_1, \ldots, \lambda_{k-1}\}$ are all higher (lower) than $\{\lambda_k, \ldots, \lambda_d\}$;

- the median of the first portion $\{\lambda_1, \ldots, \lambda_{k-1}\}$ is higher (lower) than the median of the second portion of the set by more than a relative difference χ;
- $k+2 \leqslant n$, to avoid classifying an outlier as a level shift.

When a level shift is detected, all measures prior to λ_k are ignored and the predictor is restarted from λ_k. In the following, we select $(\chi, \psi) = (0.3, 0.4)$ as in [12], which corresponds to a good parameter choice in our data set as well.

4.3. Auto-regressive models

The last class of forecast models that we will compare SVR performance with is the Auto-Regressive (d-AR) one. d-AR models are similar to d-MA models: the main difference is that they take into account not only previous observations, but previous *predictions* as well:

$$\lambda_k = \sum_{i=1}^{d} \varphi_i \lambda_{k-i} + \epsilon_k. \tag{11}$$

d-AR predictors takes the general form of (11), where φ_i are parameters of the model and ϵ_k is a noise factor. In the embedding process context, a linear prediction function leads to a *class* of autoregressive models, and fitting procedures can be used to extract the most appropriate regression function within the class. Indeed, it has been shown [19] that the use of a *linear kernel* yields to an d-AR model equivalent to the one estimated through other means such as, e.g., the Yule–Walker equations [7]. At the same time, the advantage of using SVR to provide d-AR models lies in its simpler fitting procedure and robustness in the presence of outliers: thus, in the following we will use a SVR with linear kernel to evaluate the performance of autoregressive models.

5. Sensitivity analysis of SVR performance

Support vector embedded process regression is affected by many parameters, pertaining to two different areas. A first set, related to the SVR itself, includes the training size S and the smoothing factor C of (2), the tolerance ϵ of the loss function (3), and the parameter γ of the kernel function (8). The second set is instead related to the embedding process parameters, i.e., the timescale τ and dimension d of (9). In the following, we provide a very thorough and careful tuning of SVR, with the twofold intent of (i) evaluating the extent of SVR accuracy for link load prediction, as well as (ii) assessing the sensitivity of SVR forecast performance to the above parameter variation. Results reported in this paper are gathered through JMySVM [20] an open source SVR implementation distributed along with the Rapidminer [21] software tool.

5.1. Input data

Prior to inspecting the impact of the above parameters on SVR performance, we need to provide details on the input data, that were collected at the POP of a major Italian ISP. This dataset is very interesting, 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, namely the downlink one. We then extracted several 10,000 second long (about 2h45) subsets of the trace: here, we report results referring to two different subsets, namely a daily-busy period and nightly-idle one. In the daily subset, average link load is 121.3 Mbps, whereas in the nightly subset (N), average load was 66.2 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.

Further details relative to the mean and standard deviation of the load at different timescales are reported in Table 1 for even values of i: it can be seen that, generally, the lower the timescale, the higher the load variance. Also, considering the coefficient of variation CoV $= \sigma/\mu$, we notice that load variation is more important during the night, where CoV is roughly double than the daily subset.

For all different timescales, we construct a $N = 10{,}000$ long dataset with the partitioning criterion illustrated in Fig. 1a: the subset for the $(i-1)$th timescale corresponds to the central portion of the ith one.

The daily and nightly datasets present another interesting difference. Fig. 1b depicts the autocorrelation function of the load at timescale $\tau = 1$ s: the peak of the busy trace exhibits 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 $\tau = 1$ s (i.e., λ_k and λ_{k-5}) and more generally any two samples that are $5n$ seconds apart.

5.2. SVR parameters

To tune the SVR performance, we start by performing a *grid optimization* process, which boils down to the selection of a tuple $(C^*, \epsilon^*, \gamma^*)$ of SVR parameters. The best tuple is chosen as the one that minimizes the Root Mean Square Error (RSME) of the prediction, a metric which asses the quality of the estimator in terms of its variation and unbiasedness. RMSE has the same units as the quantity being estimated (specifically, Mbps in our case), and is defined as

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i}^{n}(y_i - \hat{y}_i)^2}. \tag{12}$$

We point out that while RMSE is more suited than the Relative Error (RE) to assess the quality of estimation, at the same time its interpretation is somewhat harder than that of the RE index. As such, in the following we will mainly use RMSE to drive the parameter tuning and selection, but will use both RMSE and RE metrics to quantify the forecast accuracy.

Table 1
Input trace: link load at different timescales.

	i	0	2	4	6	8	10
	t_i (ms)	1	4	16	64	256	1024
Day-time	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
Night-time	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

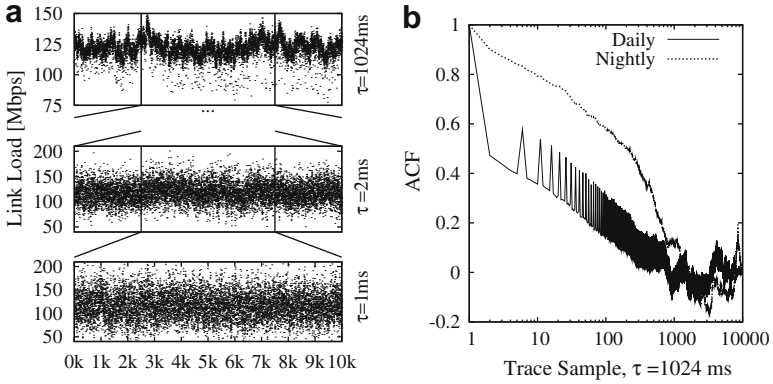

Fig. 1. (a) Subset choice at different timescales for the daily dataset and (b) autocorrelation function of the daily and nightly trace datasets.

Fixing for the moment $d = 5$ and $\tau = 1$ s, we construct a dataset with all possible inputs/outputs by means of a sliding window as described in Section 4. We selected 20% of the dataset at random to train the SVM, and tested the prediction accuracy over the remaining 80%.

We choose 10 values for each of the SVR parameters, for a total of 1000 tuples (C, ϵ, γ). 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 (7): if we consider that, $|\alpha_i - \alpha_i^*| \leqslant C$ and $|k(x_i, x_j)| \leqslant 1$, we have that $|f(x)| \leqslant S_V C$, which yields $C \leqslant |f(x)|/S_V$. 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 all training data are support vectors $S_V = S$, or only a single data point is a support vector $S_V = 1$. Finally, [22] suggest that ϵ 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.

This grid optimization process for the daily trace yields to a minimum RMSE = 5.9 when $(C^*, \epsilon^*, \gamma^*) = (30, 5, 0.05)$, to which a relative error RE = 3.5% corresponds. Fig. 2 shows the whole (C, ϵ, γ) parameter set explored, conditioning over each of the three parameters. For the sake of clarity, let us consider the leftmost plot of Fig. 2, whose x-axis represents the C parameter values. Each point in the plot represents a single experiment, and for any value of the C parameter on the x-axis, 100 points are plotted that correspond to the 100 combinations of the other two parameters ϵ and γ. The plot also contains 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 and 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. 2, 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 (iii) that the impact of ϵ is less significant with respect to C and γ. A similar operation on the nightly trace yielded a different best combination of parameters, namely (25, 0.1, 0.001). 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.

5.2.1. Training size impact

Next, we explore the impact of the training set size S on SVR performance: we build a training set with S randomly chosen samples and evaluate the prediction accuracy over the remaining samples. The process is repeated 10 times for each value of S, changing the training and validation set every time. RMSE results are reported in the boxplot Fig. 3 as a function of the training size S (top x-axis) and ratio S/N over the total trace size (bottom x-axis). The boxes report the lower quartile, median, and upper quartile values; the lines extending from each end of the boxes represents the extent of the rest of the data (i.e., the maximum and minimum values), and outliers are not filtered out.

First, it can be observed that there exists an noticeable variation in the RMSE performance for a given training set size, emphasizing the importance of the training set

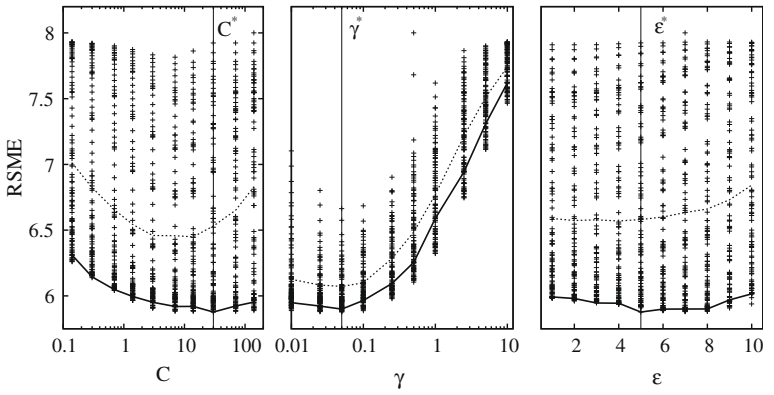

Fig. 2. Grid optimization process for the selection of $(C^*, \epsilon^*, \gamma^*)$.

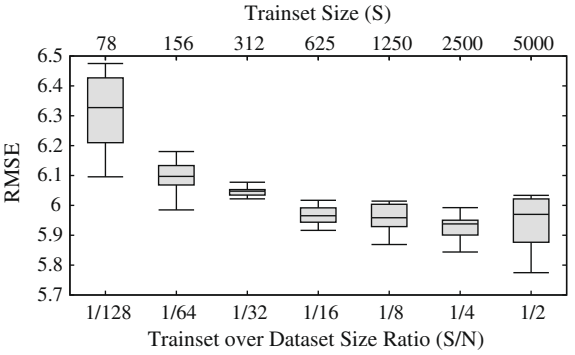

Fig. 3. Impact of training size on prediction accuracy.

in the model construction. Then, it can be observed that whenever the number of samples is small ($S/N < 1/16$), the SVM is *under-trained* and prediction error is large. For instance, relative error, not shown on the picture, grows beyond 10when $S/N < 1/128$.

Afterwards, SVM rapidly learns and both the RMSE and RE errors quickly drop (specifically, RMSE = 5.9 and RE = 3.5% when $S/N = 1/4$), until the SVM is *over-trained* ($S/N \geqslant 1/2$) and the error slightly increases (RE = 4.2%). In our situation, in the zone $S/N \in [1/8, 1/4]$ the RMSE stays at acceptable values, with a minimum in $S/N = 1/4$, which validates our choice of $S/N = 20\%$.

5.3. 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 and AR models. As earlier explained in Section 4 AR models are evaluated through a SVR with a linear kernel. In this case, the parameters that must be set are only C and ϵ: by performing a new grid optimization process, we obtain that the best choice is keeping the values C and ϵ already obtained for the radial basis kernel. Fig. 4 reports the RMSE results as a function of the number d of previous samples, for both nightly (right) and daily (left) periods at the $\tau = 1$ s timescale. Each point in the plot correspond to the average result over 10 repetitions of the experiment. As a first remark, results are different, though quantitatively very close: in other words, SVR does not appear to offer a significant improvement, especially over d-AR models.

Nevertheless, let us investigate more closely the daily period, where the d-MA is heavily affected from 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 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 SVR models is preserved. It has to be noted that both SVR and d-AR models are similarly affected by variations of d, but that the use of a radial kernel yields to slightly more accurate results.

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 starts averaging useless information, actually worsening its accuracy. At the same time, while the d-LSO limits 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, SVR outperforms d-MA for all values of d, and is also naturally robust even to unreasonable choices of d – i.e., increasing the number of features neither ameliorate nor degrade the resulting accuracy. When considering d-AR models, we find out that the performance is very close to that SVR: this can be expected as in this case, the very small value of the $\gamma = 0.001$ means that linear kernel is a good approximation of the radial one. Still, the slightly better performance achieved by the radial basis kernel confirms it to be a good choice, other than for throughput [3] and latency [4] prediction, even for the purpose of link load forecast.

Finally, setting $d = 5$ (thus, the best case for the d-MA filter but *not* for the SVM), we investigate whether SVM forecasts bring any improvement at short time-scales. RMSE of the prediction is depicted in Fig. 5 as a function of τ, where we report only the nightly period to avoid cluttering the pictures. Behavior of both predictors is similar,

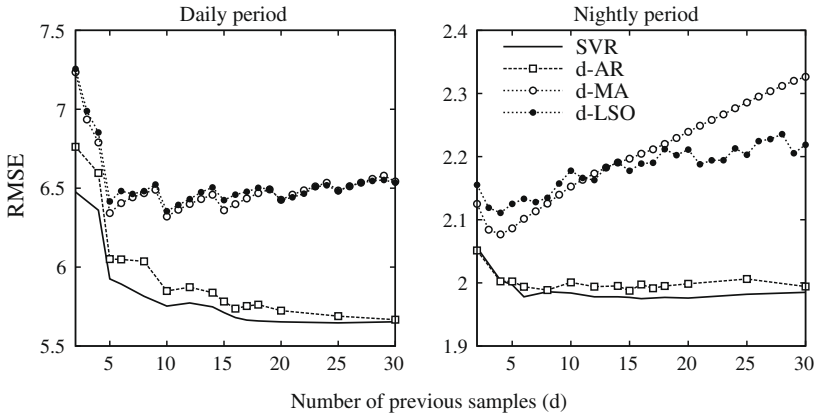

Fig. 4. Impact of the number d of previous samples on the forecast accuracy.

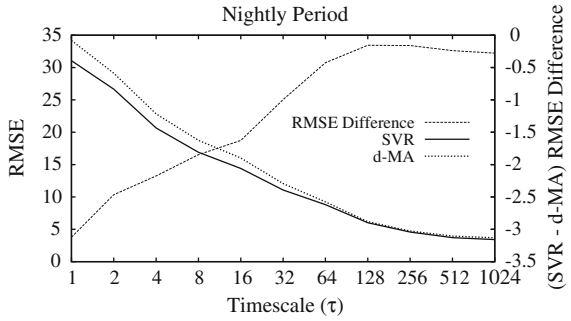

Fig. 5. Impact of the timescale τ on the forecast accuracy.

with short time scales constituting a stiffer scenario, as can be expected as a result of the much higher traffic variability shown earlier in Table 1. The picture also shows the RMSE difference of the two forecast techniques, from which it can be gathered that at very short time-scales (1 ms), SVM brings about a 10% improvement over d-MA.

6. A closer look at forecast performance

6.1. Temporal evolution

After having performed a sensitivity analysis of SVR, let us dig deeper into SVR performance, starting by an investigation of the temporal evolution of SVR prediction. Let us start with an example where, using the SVR parameters obtained through the grid optimization, we fix $\tau = 1$ and $d = 10$ and devote the first 2000 sample of the daily dataset for training and the last 2000 consecutive samples for testing.

The left portion of Fig. 6 reports the temporal evolution of real data and SVR forecast (top) as well as the relative error RE (bottom), from which it can be gathered that SVR prediction closely follows the real link load evolution, though the latter exhibits several load "drops", whose duration is on the order of a seconds, which SVR is unable to predict. We stress, however, that from a provider perspective it may not be critical to be able to precisely predict such short load drops, since no specific action needs to be undertaken as a reaction. This is clearly in stark opposition with respect to load spikes, which we argue would be more critical to be able to anticipate, in order to promptly trigger possible counter-measures.

The impact of load drops on SVR performance is further highlighted in the right portion of Fig. 6, which depicts a scatter plot of the real and forecasted load values (top) and the probability distribution function of the SVR relative error (bottom): from comparison of these pictures is clear that the bias toward over-estimation is entailed by the early noticed load drops, rather than by a systematic bias introduced by the model.

This asymmetry due to the load drops, slightly bias the mean relative error toward negative values $\mu = -0.0065$. At the same time, we point out that performing a randomness test for the sequence of relative errors measured over the test set, we find that errors are independent and identically distributed. In particular, results of a Runs Test state that we could not reject the hypothesis of randomness for this sequence (with a p-value of 0.3161): this allows us to calculate a 95% confidence interval for the error $I = [-0.0092, -0.0039]$.

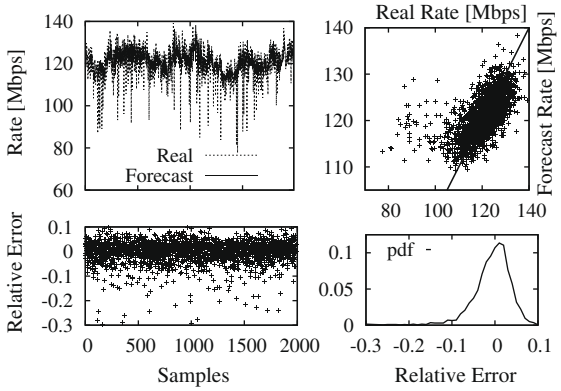

Fig. 6. Time evolution of SVR prediction.

This fact has very important consequences in the case of online predictions, as it could be used in two rather different contexts. For instance, when the forecast accuracy starts to degrade (as many predictions slightly fall outside the confidence interval), this may imply that a new training phase should be triggered. Conversely, if errors differ drastically from the previous ones, this might imply that something unusual is occurring with the traffic load – which might find useful applications in the field of anomaly detection.

6.2. Computational complexity

The SVR forecast performance can also be described in terms of their *cost*. First of all, it is important to observe that SVR involves two rather different tasks, namely model *training* and the actual *forecast* operation: the former translates into the solution of an optimization problem, whereas the latter only involves a limited number of simple operations. The above decoupling of SVR computational complexity is a very desirable property. Indeed, training and forecast tasks have to be performed at intrinsically different timescales: model training is an offline operation, that has to be done episodically (or at most periodically) and in the background, while forecasting need to be performed constantly and in an online fashion.

Concerning the model training, we need to stress that there exist very efficient algorithms for the solution of the SVR problem: for example, in this paper we use an implementation of a sequential minimal optimization decomposition technique, whose computationally complexity is *linear* in the number of support vectors.

The computation of the forecast itself only involves a number of simple operations as can easily be gathered by observing the SVR regression function (7). Moreover, an interesting point is that it is actually possible to *upper bound* a priori the number of support vectors returned as a solution to the problem. This can be done by adopting a slightly different formulation of the SVR problem, called ν-SVR [13]: as opposite to the ε-SVR case, though, we would no longer be able to tune the ε-tube, i.e., the margin ϵ below which errors are tolerated. Thus, rather than tuning the tolerable error, one could choose to fix the maximum cost that can be afforded – which could be extremely useful to limit the amount of CPU resources devoted to the online forecast operation.

Still, in the case of ε-SVR, the online forecast cost can be evaluated a posteriori, and as we will show, it clearly does not constitute a performance bottleneck. The primary complexity indicator for the SVR forecast operation is clearly the number of support vectors. Fig. 7 reports, for the daily dataset and radial basis kernel, the number of support vectors generated by the model as a function of the embedded parameter d. Box-plots report the median, first and third quartiles, minimum and maximum number of support vectors obtained over 100 different training per value of d; on the left y-axis, we also report the percentage of support vector over the training set size (which is again set to 20% of the dataset).

For small values of d, more support vectors are needed while for values greater than 5, the mean percentage remains almost constant around 17%, though we can still observe a variability of the results, which is due to the random selection of the training set. For the AR models (evaluated via the use of SVR with linear kernel) results are similar, with an increase of about 5% on the number of support vectors – which would thus imply a slightly higher computational complexity and resource requirement in both the offline training and the online forecast operations.

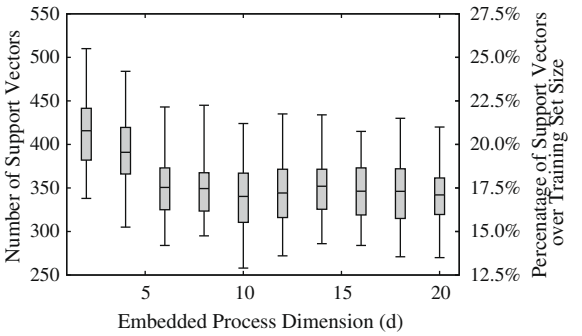

Fig. 7. Number of support vectors for different values of d.

Finally, we give a very rough but nevertheless useful feeling of the actual CPU requirement, reporting the typical execution time performance of our experimental campaign. Experiments, which were run on a Linux PC featuring a 2.0 GHz Intel Core 2 Duo processor equipped with 2 GB of RAM, show that:

- the offline training time is about 0.83 seconds per thousand support vectors – which means that the mean training time of a single SVR model of Fig. 7, averaged over all d values, is less than a third of second;
- the online forecast rate is above 9000 forecasts per second when $d = 10$ – thus much faster than real-time, which further testifies to the viability of online SVR forecast.

6.3. Extending the forecast horizon

Finally, an interesting question that we want to assess is whether it is possible to extend the forecasting horizon by *cascading* a series of SVR predictors – in other words whether recursively using forecasted results as an input for a new forecast is viable and promising in the case of link load forecast. We stress that the approach is feasible and intriguing, as results on computational complexity show that, when $\tau = 1$ s, it would be possible to actually cascade more than 9000 SVR predictors in a real-time online forecast: thus, we want to assess how fast the prediction accuracy degrades.

More formally, at time $t = d\tau$ we forecast the load at the next time step $t = (d+1)\tau$, feeding the SVR function $f(\cdot)$ with the last d observation of the series, getting as result the prediction $\hat{y}_{d+1} = f(\hat{x}_d)$. Always at time $t = d\tau$, we then forecast the load at time step $t = (d+2)\tau$ using the last d-1 observation of the series and the *forecasted* value $\hat{y}_{d+1}$ in the input $\hat{x}_{d+1}$, thus:

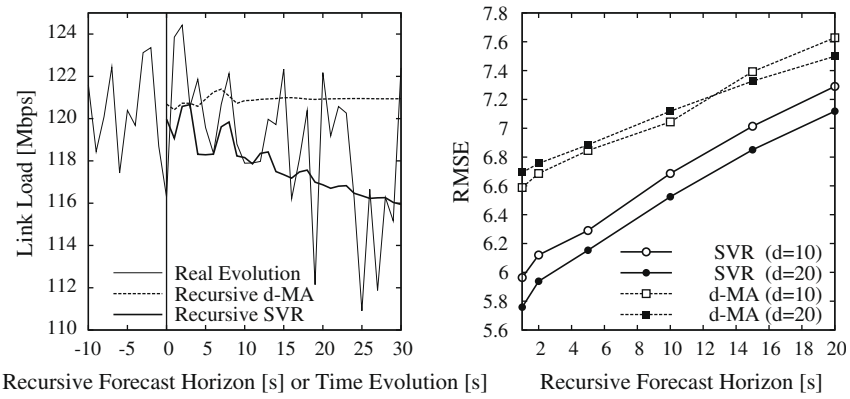

Fig. 8. Example of temporal evolution of recursive forecast (left) and RMSE error for different values of the forecast horizon (right).

$$x_d = \hat{x}_d = (\lambda_1, \ldots, \lambda_d), \qquad \hat{y}_{d+1} = f(\hat{x}_d),$$
$$\hat{x}_{d+1} = (\lambda_2, \ldots, \lambda_d, \hat{y}_{d+1}), \qquad \hat{y}_{d+2} = f(\hat{x}_{d+1}),$$
$$\hat{x}_{d+2} = (\lambda_3, \ldots, \lambda_d, \hat{y}_{d+1}, \hat{y}_{d+2}), \quad \hat{y}_{d+3} = f(\hat{x}_{d+2}).$$

This procedure can be iterated arbitrarily, and we define as *forecast horizon* H the number of SVR forecasts that are cascaded: notice that when $H \geqslant d$, this means that we are using only forecasted values as input features. In what follows, to avoid the influence of the training set, we train and validate the model over the same dataset: by doing so, we will be able to isolate the impact of the forecast horizon.

For illustration purpose, the left plot of Fig. 8 gives an example of recursive forecast for SVR and d-MA models when $d = 10$ and for an horizon up to $H = 30$ s. At time $t = 0$, we apply the cascading process and plot the $\hat{y}_h$ prediction for a given horizon h: notice that the x-axis represent the forecast horizon for both SVR and d-MA, whereas it represents the time (in the future) as far as the real series is concerned. In the picture, the $d = 10$ real load samples immediately preceding time $t = 0$ are also shown: these points determine the initial load forecast for time $t = 1$, after which the recursive process starts (and so SVR and d-MA models start to be fed with their own forecasts).

From this simple example, it can be seen that SVR is able to more closely follow the real data with respect to d-MA, at least for small values of H. Conversely, d-MA predictor quickly converges to a fixed point, though the precise value to which an ∞-cascaded d-MA filter converges depends on the actual *order* of the series.[1] Aiming at assessing the average RMSE and RE of the recursive forecast, and in order to gather results that are valid to a more general extent, we perform an exhaustive set of experiments, repeating the process 10 times for different training sets. The right portion of Fig. 8 depicts the RMSE error as a function of the forecast horizon for both SVR and d-MA and two values of $d \in \{10, 20\}$. RMSE is evaluated as the error between the cascaded value $\hat{y}_{d+H} = f(\hat{x}_{d+H-1})$ forecasted at time $t = d\tau$ and the real value λ_{d+H} at time $t = d\tau + H$. Results show that in all cases the RMSE error linearly increases with the forecast horizon H, but that cascaded SVR performance exhibit a lower error. Moreover, it can be gathered that $d = 20$ yields better results: this is not surprising, since in this case we are using more "real values" as input features.

[1] For example, it is easy to verify that the d-MA cascading process over $x = (1,2,3)$ converges to $2 + 1/3$, whereas it converges to $2 - 1/3$ in case of $x = (3,2,1)$.

Table 2
Relative error RE between H-horizon and "one-step" SVR forecast.

H	1	2	5	10	15	20
$d = 10$	0%	0.7%	1.1%	1.9%	2.5%	2.8%
$d = 20$	0%	0.8%	1.3%	1.9%	2.4%	2.8%

Finally, we compare the H-horizon cascaded SVR forecast (i.e., the recursive SVR forecast for the series value at time $t + H$, performed at time t), with the "one-step" forecast (i.e., a normal SVR forecast). This comparison helps us in assessing the extent of the degradation in the prediction quality, which can be solely imputed to the cascading process itself. These results, expressed in terms of RE, are reported in Table 2 for different values of H: rather surprisingly, results state that the prediction of a cascaded SVR with a rather large horizon of $H = 20$ s is only about 3% worse than a "one-step" prediction, and that, furthermore, this holds irrespectively of d. Interestingly, this suggests that, if good accuracy is obtained for $H = 1$, the recursive process would not much affect the quality of the prediction.

7. Discussion and conclusion

This paper explores the use of Support Vector Regression for the purpose of link load forecast: using a hands-on approach, we tune the SVR performance and compare it with those achievable by using Moving Average (MA) and Auto-Regressive (AR) models. Our results show that, despite a good accordance with the actual data, the SVR gain achievable over simple prediction methods such as MA or AR is not sufficient to justify its deployment for link load prediction at short time scales. Yet, we have to pay a tribute to SVR for a number of extremely positive aspects: for instance, SVR models are (i) rather robust to parameter

variation, (ii) their computational complexity is far from being prohibitive, and (iii) the cascading of SVR models may significantly extend the achievable forecast horizon, entailing only a very limited accuracy degradation.

It is our belief that this work constitutes a starting point for further investigation, whose directions are highlighted in the following. First, in order to gather more robust results, different traces representative of rather different network scenarios should be used to validate the extent of the above analysis. Then, the question remains about what can be done to improve the performance of SVR at short time scales: preliminary results seem to suggest that a manipulation of the time series (e.g., differentiation, statistical properties, etc.) may bring a significant benefit in terms of the forecast accuracy – in which case comparison over more sophisticated techniques for time series forecast would be needed. Another open issue is whether the use of other kernels (e.g., multi-linear or other that can take into account the characteristics of the time series) possibly improves the SVR accuracy, which could thus avoid the burden of costly time-series manipulation. Finally, other interesting directions for this research could possibly involve the evaluation of different forecast targets with respect to the average link load (such as the peak load, or the 95th percentile, etc.) as well as the analysis of longer timescales. Indeed, concerning the latter point, it could be interesting to investigate whether feeding SVR with features such as time-of-day and day-of-week would help in forecasting periodic load fluctuations (such as lunch breaks and week-ends).

Acknowledgement

This work was funded by the Celtic project TIGER.

References

[1] B.E. Boser, I.M. Guyon, V.N. Vapnik, A training algorithm for optimal margin classifiers, ACM COLT'92, Pittsburgh, PA, 1992, pp. 144–152.
[2] K.-R. Muller, A. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen, V. Vapnik, Predicting time series with support vector machines, Artificial Neural Networks, Lecture Notes in Computer Science, vol. 1327, Springer, Berlin, 1999.
[3] M. Mirza, J. Sommers, P. Bardford, X. Zhu, A machine learning approach to TCP throughput prediction, Proc. of ACM SIGMETRICS'07, San Diego, USA, 2007.
[4] R. Beverly, K. Sollins, A. Berger, SVM Learning of IP Address Structure for Latency Prediction, Proc. of ACM MineNet'06, Pisa, Italy, 2006.
[5] L. Khan, M. Awad, B. Thuraisingham, A new intrusion detection system using support vector machines and hierarchical clustering, The VLDB Journal 16 (2007).
[6] A.H. Sung, S. Mukkamala, Identifying important features for intrusion detection using SVM and neural networks, Proc. of IEEE SAINT'03, Orlando, FL, USA, 2003.
[7] P.J. Brockwell, R.A. Davis, Introduction to Time Series and Forecasting, Springer, Berlin, 1996.
[8] J. Beran, Statistics for Long-memory Processes, Chapman & Hall, London, 1994.
[9] B. Krithikaivasan, Y. Zeng, K. Deka, D. Medhi, ARCH-based traffic forecasting and dynamic bandwidth provisioning for periodically measured nonstationary traffic, IEEE/ACM Transaction on Networking (2007).
[10] W.E. Leland, M.S. Taqqu, W. Willinger, D.V. Wilson, On the self-similar nature of ethernet traffic, IEEE Transactions on Networking 2 (1994) 1.
[11] N.G. Duffield, A. Greenberg, P. Mishra, K.K. Ramakrishman, J.E. van der Merwe, Resource management with hoses: point to cloud services for virtual private network, IEEE/ACM Transactions on Networking 10 (2002) 5.
[12] Q. He, C. Dovrolis, M. Ammar, On the predictability of large transfer TCP throughput, Proc. of ACM SIGCOMM'05, Philadelphia, USA, 2005.
[13] A.J. Smola, B. Scholkopf, A tutorial on support vector regression, in: Statistics and Computing, vol. 14, Kluwer Academic Pub., 2004, pp. 199–222.
[14] N. Cristianini, J. Shawe-Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, New York, NY, 1999.
[15] C.J.C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition Journal of Data Mining and Knowledge Discovery, vol. 2, Springer, Netherlands, 1998. p. 2.
[16] V. Vapnik, The Nature of Statistical Learning Theory, Springer, NY, 1995.
[17] M. Aizerman, E. Braverman, L. Rozonoer, Theoretical foundations of the potential function method in pattern recognition learning, Automation and Remote Control 25 (1964) 821–837.
[18] J. Mercer, Functions of positive and negative type and their connection with the theory of integral equations, Philosophical Transaction of the Royal Society, London A 209 (1909) 415–446.
[19] S. Ruping, M. Katharina, Support vector machines and learning about time, Proc. of IEEE ICASSP'03, Hong Kong, 2003.
[20] T. Joachims, Making large-scale support vector machine learning practical, Advances in Kernel Methods: Support Vector Machines, MIT Press, Cambridge, MA, 1998.
[21] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, T. Euler, YALE: rapid prototyping for complex data mining task, Proc. of ACM SIGKDD'06, PA, USA, 2006.
[22] V. Cherkassky, Y. Ma, Practical selection of SVM parameters and noise estimation for SVM regression, Neural Networks 17 (2004) 1.

Paola Bermolen was born in 1976 in Montevideo. In 2004, she received a degree in Mathematics from the Universidad de la Republica, Montevideo, Uruguay, where she joined the Mathematical Department (IMERL) in 1998 as teaching assistant. Since October 2006 she is a Ph.D. student at TELECOM ParisTech in Paris, France. She has co-authored several papers presented in international conferences and journals, and she has been involved in different national and international projects. Her research interests are related to statistical characterization and analysis of network traffic, network modeling, and performance analysis in heterogeneous networks.

Dario Rossi was born in Torino, Italy, on a sunny sunday of March 1977. He received his M.Sc. degree in Electronic Engineering in 2001 and his Ph.D. in Electronic and Telecommunication Engineering in 2005, both from Politecnico di Torino. Between September 2003 and August 2004 he held a visiting researcher position in the Computer Science division at University of California, Berkeley. Since October 2006, he is an Associate Professor at the INFRES department at Ecole Nationale Superieure de Telecommunications (ENST), in Paris, France. He has coauthored over 30 papers in leading conferences and journals and currently holds three patents. He participated in the program committees of several conferences including IEEE ICC, IEEE IPCCC and IEEE ISCC. He is responsible for several European research projects, such as FP7 NAPA-WINE, Celtic TIGER and Celtic TRANS. His research interests are in the fields of peer-2-peer networks, Internet traffic measurement, sensor and vehicular networks.