Network Forecast with Support Vectors Machines

Paola Bermolen and Dario Rossi
TELECOM ParisTech, France – INFRES Department
firstname.lastname@enst.fr

***Abstract*— In this paper, we address the problem of forecasting a function of the link load, such as the peak load or percentiles of its distribution, during an arbitrary time interval. As forecasting technique, we select Support Vector Machines (SVM), which have shown very good performance in several areas including, more recently, networking. SVM are known for its generalization ability with respect to unseen data and their suitability to on-line learning.**

Using a hands-on approach, we evaluate the impact of several parameters on the SVM prediction accuracy. In order to gather robust results, we consider several real-world traffic traces, representative of very different network scenarios –such as ISP, Ethernet, WiFi LAN and enterprise networks– so as to be able to print out a fine-grained picture of the gain brought by SVM in the context of network load forecast. Our results show that SVR may provide accurate predictions and substantial gain over a naive estimation technique, and furthermore that the use of several machines in parallel can both ease SVM tuning and increase forecast performance as well.

I. Introduction

As IP network are truly becoming multi-service (e.g., triple play convergence) and Internet applications evolve (e.g., Skype and other P2P-VoIP software and more recently PPlive and other P2P-TV systems), the Internet traffic is becoming increasingly complex and dynamic. The ability of accurately forecasting such variability can be crucial for both ISPs and telcos, and furthermore for very different tasks: e.g., short-time scale prediction can be coupled to self-management techniques, while long-time scale prediction is a valuable tool for capacity planning.

In contrast with most of the work related to network load forecast, which are based on the analysis of time series properties [1], we prefer to focus on techniques that avoid to make any assumptions on the phenomenon under observation. In particular, different models have been proposed in the literature, but the majority of these approaches relies on specific assumptions about the network traffic (e.g., they are tailored to capture Long Range Dependence LRD [2] at short and long timescales, etc.). Also, the effectiveness of these models usually relies on the precise estimation of some traffic parameters (e.g. Hurst parameter of the arrival time series), whose estimation can be delicate and computationally intensive.

We address the problem of forecasting (an arbitrary function of) the link load by means of Support Vector Machines (SVM) [3], [4]. More precisely, we aim to predict the maximum (or a percentile) of the link load over a given temporal horizon.

This work was funded by the Celtic project TRANS.

SVM are quite new statistical learning techniques, that can be defined in the classification as well as regression context. These techniques have been widely used in different contexts after its origins in optical character recognition: for example, time series prediction [5] or more recently networking [6], [8], [9], only to name a few possible applications that are in some way related to our work. We consider here the regression approach (SVR) in which basically a prediction model is constructed from a training set and this model can be applied later to unseen data. A major property of SVR is its generalization capacity, i.e accurate prediction even for data that is not included in the training set. Also, these models allow continuous and adaptive on-line learning which is clearly suitable for networking purposes.

In our previous work [6], we preliminary explored the problem of link load forecast at short time scales, adopting an approach known as "embedding process" in the SVM lingo: basically, the link load is treated as a *series*, whose future value is forecast based on an arbitrary number of its past measurements. Despite a number of positive aspects of SVM were outlined in [6], forecast performance were not deemed satisfactory enough to justify SVM deployment. Results indeed showed that SVR based models are rather robust to parameter variation, which constitutes an undoubted positive aspect. Also, we found that the shorter the timescale, the harder the prediction. Moreover, when the number of past samples used as features is exiguous, adding a few more features dramatically improves the prediction accuracy. Clearly, as more features are added, the gain saturates (i.e., adding more features does not yield to a better accuracy), but it does not degrade neither (which further confirms SVM robustness). Yet, despite a good accordance with the actual data, the SVM embedding process approach only yielded a marginal gain over simple estimation techniques.

In this paper, taking a radically different approach from [6], we devise techniques that significantly improve the accuracy of SVM link load forecast. In more detail, as SVM input we consider a *summary* of statistical properties (e.g., mean, variance, quartiles, peak, etc.) of the link load, as opposite to the past measurement of the link load *series* itself. Following a hands-on approach, we quantify the impact of several factors (such as forecast timescale, samples aggregation strategy, input feature combination, forecast target, type of traffic, etc.) in the SVR forecast accuracy. Some preliminaries results can be found in [7]. Finally, we also show that the use of an "intelligent" combination of different machines, can bring further advantages in both the forecast accuracy as well as in

the tuning of SVM, which allows us to construct a very robust model. We evaluate our model over different real world traffic traces representing very different network scenarios (Ethernet and WiFi LAN, ISP and enterprise). Our results show that SVR may provide accurate predictions and very significant gain over both naive estimation techniques and also over the "embedding process" adopted in [6].

The remainder of this paper is organized as follow. In Sec. II we briefly describe the Support Vector Regression theory, and introduce our forecast framework. Sec. III describes then the experimental dataset that we use in Sec. IV to explore the impact of the different parameters in the SVR forecast performance. Results when several machines in parallel are considered are shown in Sec. V, while a comparison with the embedding process is presented Sec. VI. Finally, conclusions and future work are addressed in Sec. VII.

II. 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, R^d is the space of the input features x_i and y_i is the phenomenon under investigation. In ϵ-SV regression [10] 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 $f(x) = \langle w, x \rangle + b$, where $\langle \cdot, \cdot \rangle$ denote the dot product in R^d. Flatness in this case can be ensured by minimizing the norm $\|w\|^2$, which leads to the following convex optimization problem:

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

In the above formulation, slack variables ξ_i, ξ_i^* are included to cope with 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.

The training problem (1) can be solved more easily in its *dual* formulation, which results in a quadratic optimization problem with a unique solution, thus avoiding to get stuck on a local minimum. The solution of (1) yields to the forecast function $f(x)$, which 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) conditions:

$$f(x) = \sum_{i=1}^{S} (\alpha_i - \alpha_i^*) \langle x_i, x \rangle + b = \sum_{i=1}^{S_V} (\alpha_i - \alpha_i^*) \langle x_i, x \rangle + b \tag{2}$$

However, not all x_i are needed to calculate $f(x)$ but only those $S_V < S$ training points x_i, whose $\alpha_i, \alpha_i^* \neq 0$ and which are referred to as *support vectors*. 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. Interestingly indeed, there exist very efficient algorithms for the solution of the SVM problem: for example, in this paper we use the LibSVM [11] tool, which implements a Sequential Minimal Optimization decomposition technique, whose computationally complexity is *linear* in the number of support vectors.

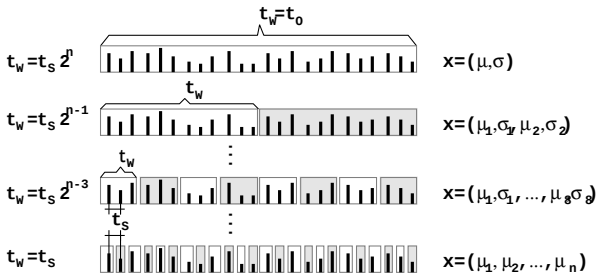

Fig. 1. Synoptic of the general framework used throughout this work

The dual formulation also provides the key to its non-linear extension, by means of the so called "kernel trick" [12], which basically boils down to restate both the optimization problem (1) and the forecast function (2) in terms of a kernel function $k(x, x')$. More formally, a function is called a *kernel* if it corresponds to a dot product in some feature space $\mathcal{F}$, which has a higher dimension than the original feature space. In this work we use the radial basis kernel $k(x, x') = e^{-\gamma \|x - x'\|^2}$, to which corresponds an infinite dimensional feature space, due to the good performance shown in both time series prediction and network contexts.

As a general remark, SVR performance is affected by many parameters, belonging to two different classes: a first set is related to the SVR regression method and kernel function, whereas a second set pertains to the input (i.e., features) and output (i.e., target) spaces. A "grid optimization" routine is used to systematically explore parameters of the first set, such as the smoothing factor C, the tolerance ϵ and the kernel parameter γ, and to select the tuple $(C^*, \gamma^*, \epsilon^*)$ that minimizes the prediction error: in [6], we shown that SVR is rather robust to the parameter selection, provided that the above parameters are selected in a "reasonable" range. Conversely, modification of the input features, the output target, etc., can significantly affect the forecast accuracy: therefore, in the following we will restrict our attention to parameters belonging to the latter class.

B. Framework

Assume, for the moment, that we want to predict the peak network load on a given temporal horizon, using past observations of the load mean μ and standard deviation σ as input features. Let us denote the temporal horizon of the forecast by t_F, and let us further assume that predictions for the time frame t_F are based on observations of a time frame t_O of equal length, i.e., $t_O = t_F$. Let us now introduce, with the help of Fig. 1, the notation that will be used throughout this

TABLE I
TRAFFIC TRACES PROPERTIES

	ISP	Campus	WiFi	Ent
Period	May'06	May'06	Sep'07	Dec'04
Duration	15h	15h	15h	1h
Rate [Mbps]	60.1	30.0	28.8	7.7
Flows	$3.4{\cdot}10^6$	$6.2{\cdot}10^6$	$3.3{\cdot}10^6$	$58{\cdot}10^3$
Pkts	$461{\cdot}10^6$	$329{\cdot}10^6$	$195{\cdot}10^6$	$9.7{\cdot}10^6$
Bytes	$0.4{\cdot}10^{12}$	$0.2{\cdot}10^{12}$	$0.2{\cdot}10^{12}$	$3.5{\cdot}10^9$
Pkts/Flow	134	53	59	167
Bytes/Pkts	864	615	997	361
IP src	$2.8{\cdot}10^3$	$11.4{\cdot}10^3$	$3.4{\cdot}10^3$	$3.2{\cdot}10^3$
IP dst	$493{\cdot}10^3$	$699{\cdot}10^3$	$431{\cdot}10^3$	$2.7{\cdot}10^3$

work. As sketched in the top of the figure, a single pair of features (μ, σ) can be gathered from the whole time frame t_O, which can then be used as SVR inputs to predict the maximum load in the subsequent time frame t_F. Alternatively, the samples constituting the time frame t_O can be *aggregated* into several windows of duration t_W, where a separate set of input features (μ_i, σ_i) can then be calculated for each window i. For instance, going down one step, the window length is halved $t_W = t_O/2$, which doubles the number of features, producing $(\mu_1, \sigma_1, \mu_2, \sigma_2)$. Potentially, this dichotomic splitting procedure could continue until the window size reaches a minimum, given by the traffic sampling time (i.e.,. $t_W = t_S$). However, as 1-sample windows do not allow us to evaluate σ, in the following we disregard the degenerated case $t_W = t_S$ and consider $t_W \geq 2t_S$. It is worth noting that the case with 1-sample windows when the input is defined as the mean, is equivalent to the "embedding process" analyzed in [6]. This means that the input is defined as a vector $x \in \mathbb{R}^d$, where the fact that $t_S = 1$ second, implies that $d = t_O$. In Sec. VI we will show that the approach presented in this work brings better results (10% in the worst case) than the "embedding process".

For the sake of clarity, in the following we will refer to a forecast *horizon* t_F, based on an *observation* time frame t_O, which is possibly split into several time *windows* t_W, containing a number of samples collected using a constant *sampling time* t_S. Also, samples contained in the i-th window can be *consolidated* into a set of features, for example, (μ_i, σ_i) in this example: therefore, the SVM input will be the *union* of all feature sets, over all windows in which the observation time frame has been split. The impact of the above variables will be studied at length in the following, but, unless otherwise stated, we will refer to $t_F = t_O = t_W = 64$ seconds (i.e., the power of 2 closest to 1 minute interval), using a sampling interval of $t_S = 1$ second. In this case, no splitting is performed.

At time scale t_S, for each dataset let $\lambda(t)$ be the traffic load measured in the time interval $[t - t_S, t]$. By quantizing the time in multiples of t_S, we obtain a time series $\{\lambda_k\}_{k\in\mathbb{N}}$, where λ_k is the average traffic load measured in the interval $[(k-1)t_S, kt_S]$. For the SVR training, we use a sliding window of length $t_O = t_F$ over this time series to build all possible inputs/outputs pairs (x_i, y_i). A subset of this dataset is used for training, i.e., to solve the SVR problem and gather the forecast function $f(\cdot)$. The model accuracy is then evaluated over the complement of the training set, i.e., on unknown data. In the remainder of this paper, for each experiment 60% of the available data is used for SVM training and 40% for validation. Also, each experiment is repeated 30 times (using different training and validation sets), so that each experimental point corresponds to the average result over different SVR instances.

III. EXPERIMENTAL DATASET

Prior to investigate the SVM performance, let us briefly introduce the different real-world traces used throughout this work: we directly monitored an ISP access link and the campus egress router ports of our Ethernet and WiFi LANs, but we also make use of the enterprise network data made available by the LBNL/ICSI tracing project [13]. Details on these datasets are reported in Tab. I, such as traces duration and bitrate, amount of flows, packets and bytes observed, average packet and flow lengths, count of distinct IP source and destination hosts. All traces are 15 hours long and were collected between May 2006 and September 2007, with the exception of the enterprise traffic, which is 1 hour long[1] and was gathered during December 2003. The ISP dataset is very peculiar, as 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, whereas no PSTN link is offered: clearly, all flavors of p2p applications are present in this downstream dataset. Ethernet and WiFi traces are typical examples of campus LAN downstream traffic, measured at the campus egress router, representing the aggregated traffic of the hosts having Ethernet or Wireless access respectively. Ethernet Campus traffic consists of a mix of Web, intranet services and Internet applications (a firewall tries to block p2p traffic, although some as, e.g. Skype, still manage to go through) whereas WiFi access is mostly used for Web browsing, mail and instant messaging. Finally, Enterprise traffic is also particular, as intranet services constitute the most important part of the traffic – for a thorough analysis of the LBNL/ICSI traffic, we refer the reader to [14].

Intuitively, these different traffic characteristics will translate into different prediction accuracy. For explanatory purposes, let us show in Fig. 2 the peak p and 95-th percentile p_{95} computed over 1 second long time-windows for both ISP and Campus traces. We easily realize that the prediction will be more difficult for the Campus trace since there are a lot of uncorrelated "spikes" in both series, especially considering the peak load. Conversely, it can be seen also, that for the ISP trace, both series are very similar, which make us expect the ISP trace to be a relatively easier forecast scenario with respect to the Campus one.

IV. EXPERIMENTAL RESULTS

In this section we explore SVM performance for different inputs and outputs, for varying values of the temporal parameters and for different traces and traffic types.

[1]This is due to the measurement methodology in [14], where different switch ports are monitored every hour.

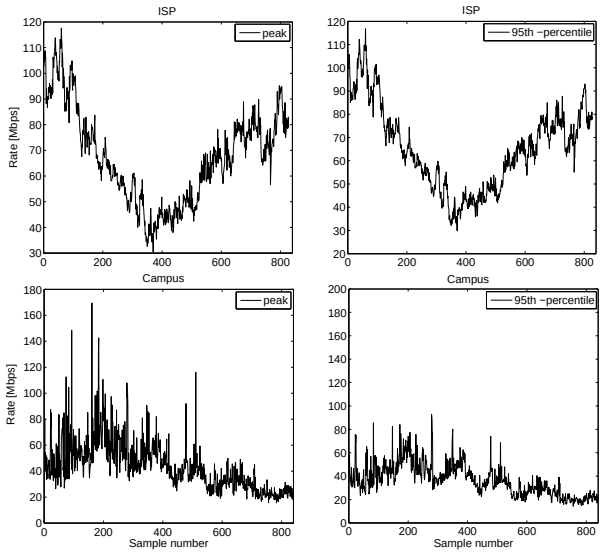

Fig. 2. Peak p and 95-th percentile p_{95} of the link load for ISP and Campus traces

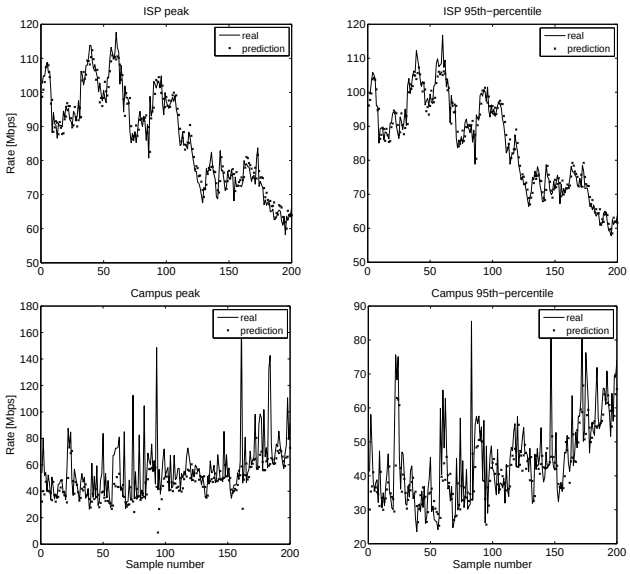

Fig. 3. Comparison of real and predicted values for ISP and Campus traces

To quantify the forecast accuracy, we consider the root mean square error, defined as RMSE $= \sqrt{\sum_i^n (y_i - \hat{y}_i)^2/n}$ which is suited to assess the quality of an estimator in terms of both its *variation* and *unbiasedness*. For reference purposes we consider the *naive* estimation, i.e., it assumes that the value in the next horizon t_F will remain equal to the value achieved in the last observation time frame t_O. Thus, we may also express the relative RMSE gain of SVR forecast with respect to the naive prediction as $G = (RMSE_{naive} - RMSE_{SVM})/RMSE_{naive}$.

A. Features and target impact

Let us start by considering different statistical properties of the link load variable as target (i.e., *output* of the model), and by feeding SVR with different combinations of their previous observations (i.e., *inputs* of the model). As statistical properties, we consider the mean μ, standard deviation σ, peak p and 95-th percentile p_{95}. Also, as far as output is concerned, for the sake of space, in the following we limitedly consider the problems of peak p and 95-th percentile p_{95} prediction. Due to space constraints, we show results only for the ISP and Campus dataset since they reflect the variety of performance results that can be obtained; we recall that we expect Campus to be a stiffer scenario, especially as far as peak load estimation is concerned.

As explained before, we preliminary fix SVR parameters (C,ϵ,γ) for each trace by performing a grid optimization for each output, using the pair (μ, p) as input (which is a reasonable choice given SVR robustness [6]). We recall that $t_O = t_F = t_W = 64$ seconds, whereas $t_S = 1$ second. For this parameter setting we show in Fig. 3 real values and SVR predictions for a random validation set for both ISP and Campus traces, to get a first (visual) idea of the prediction accuracy. As expected, the predictions of both peak and 95th-percentile are more accurate for the ISP trace: SVR is unable to forecast the important "spikes" present in the Campus traces, despite it is able to "follow" the curve. Also intuitively, the presence of these spikes makes peak prediction harder than 95-th percentile one. These differences are reflected in the RMSE: for the ISP trace RMSE is 3.65 for the peak and 3.39 for the 95th-percentile, whereas for the Campus trace these values are 21.46 and 9.94 respectively.

In what follow we will explore the use of different inputs in addition to (μ, p). More precisely, we will consider as inputs all possible combinations of input features μ, σ, p and p_{95}. Tab. II reports the RMSE obtained by the naive prediction and the corresponding SVR gain for the ISP and Campus dataset and for these different combinations of input features. Specifically, we select:

- the three best combinations of 2-features inputs (i.e., over all *pair* of statistical properties)
- the average of all 2-features inputs
- the average of inputs with 3 or more features
- the average of *all* possible combinations irrespectively of their length

First, we observe that the performance of both SVR and naive forecast significantly changes when different *traces* are considered: the RMSE change by about a multiplicative factor between 2.5 and 4. Second, considering different *outputs* as p and p_{95}, the RMSE variation can be either important (roughly, a factor of 2 in the Campus case) or irrelevant (as in the ISP dataset): as we early noticed this can be explained by the presence of significant load spikes in the Campus trace, which are harder to predict and thus yield to a larger forecast error. Third, the *input* features combination does influence the results, although with a smaller magnitude. Interestingly, it

TABLE II

COMPARISON OF NAIVE AND SVR FORECAST OF p AND p_{95}, FOR DIFFERENT INPUT FEATURES, WHERE ○=(μ),●=(μ, p),⋆=(μ, p_{95})

Input		ISP		Campus	
		p_{95}	p	p_{95}	p
Naive	**RMSE**	**3.44**	**3.91**	**8.28**	**16.55**
SVM Gain	1st	5.1% ○	8.2% ⋆	15.5% ●	21.2% ●
	2nd	5.0% ●	8.0% ●	15.1% ⋆	20.9% ⋆
	3rd	4.6% ⋆	7.9% ○	14.4% ○	20.8% ○
	$= 2$ feat	4.7%	7.7%	14.6%	20.1%
	≥ 3 feat	3.8%	6.3%	13.0%	15.9%
	All	4.3%	7.2%	13.9%	18.3%

can be seen that is better to use a small number of features to describe the statistical properties of the trace rather than a large one. Moreover, the three best input combinations yield to very similar results, although no clear winner can be identified, as the best input varies across traces and outputs: therefore, in what follow, we will limit ourselves to consider (μ),(μ, p),(μ, p_{95}) as inputs for the SVM prediction. Finally notice that, rather surprisingly, the (μ, σ) combination yields to *worse* results with respect to the simplest input choice (μ), and that this holds furthermore for both traces and outputs. Also, the combination (σ, p) provides the *worst* results of all combinations, which suggest that the mean μ should always be considered as input feature, irrespectively of the statistical properties considered as target of the forecast.

In Sec. V we will show that it is actually unnecessary to inspect which combination of input yield the best results: the underlying idea is to use several machines in parallel, each of which is trained with different inputs for the same output (so that it is either possible to e.g., *combine* the forecast power of different machines, or also *automatically select* the best input combination).

B. Aggregation and timescales impact

In this section, we explore the impact of *timing* related parameters, such as the forecast horizon t_F and the window length t_W. It is worth to note that we have fixed $t_S = 1$ second, since this choice implies more flexibility in the selection of the the forecast horizon and will allow us to explore a wider range of values for the aggregation time window t_W parameter. We focus again on the prediction of p and p_{95} load, considering only ISP and Campus datasets, and report the SVM results averaged over 30 repetitions for each of the three best 2-features input combinations described so far.

We first consider the forecast horizon $t_F = t_O$, assuming that the window t_W is consolidated into a single set of features (i.e. $t_W = t_O = t_F$). Neglecting the degenerated case $t_F = 1\,\text{s}$, we explore values of $t_F = 2^i\,\text{s}$ for $i \in [1, 6]$, where for instance $t_F = 16$ means that we observe an interval of 16 seconds (or 16 samples, since $t_S = 1\,\text{s}$) and predict the output value over the next 16 seconds interval. Results for naive and SVR prediction of p and p_{95} are reported in

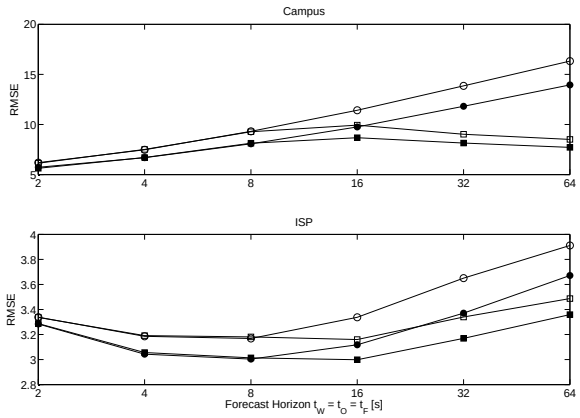

Fig. 4. Impact of the forecast horizon t_F on forecast accuracy

Fig. 4 for both the ISP and Campus traces [2]. Interestingly, the forecast accuracy is diversely affected by the forecast horizon for the different traces: this reflects the fact that network usage is much different across the datasets, and so are the temporal traffic dynamics. For example, considering the ISP trace, both p and p_{95} can be predicted with higher accuracy over a medium temporal horizon (i.e., $8 \leq t_F \leq 16$). Conversely, in the Campus dataset the p RMSE error monotonously increases with t_F, whereas the p_{95} RMSE exhibits an opposite symmetrical behavior with respect to the ISP case (i.e., medium values of t_F yield to worse performance).The increasing error for peak is in accordance with the important spikes shown in Fig. 3 since it could be easier to predict values in the near future being not possible to extend the forecast horizon. This fact have less impact when the percentile is considered as output since the spikes are not so important.

While it is hard to draw general conclusions from these specific behaviors, nevertheless one can gather that some time horizon are definitively easier to predict. Pushing this intuition a bit further, we are interested in answering whether, aiming at predicting link load over *arbitrary* time horizons, it could be beneficial to aggregate data into windows corresponding to timescales where forecast is known to be more accurate. For instance, consider the Campus case: in order to predict next minute's p_{95} load, would it be better to use i) a single set of features from the last minute window, or ii) several sets of features gathered from separate smaller windows? To answer this question, fixing $t_F = t_O = 64$, we consider the impact of the aggregation window t_W by splitting the observation time frame t_O into several windows of duration $t_W = 2^i\,\text{s}$ with $i \in [1, 6]$ as described in Fig. 1. For instance, $t_W = 32\,\text{s}$ means that the observation period t_O is split into two intervals of 32 samples, each of which gets consolidated into a different set of features: SVR is then fed with the *union* of these sets.

Results are reported in Fig. 5 in terms of the RMSE as a

[2]Clearly, the results for small time scales are equal for peak and percentile, since in such time scales, they usually coincide.

function of the window duration t_W. Moreover, we point out that, being the naive estimation RMSE constant for a fixed $t_O = t_F = 64$, the SVR gain is thus directly proportional to the error. From comparison of Fig. 4 and Fig. 5, we have a partial confirmation of our intuition: for instance, in both ISP p and p_{95} cases the minimum error at $t_O = t_F = 64$ is achieved when $t_W = 16\,\text{s}$, which is precisely the value that minimized the RMSE in Fig. 5. In other words, there are cases where for an *arbitrary* forecast horizon t_F, it is preferable to split the observation time frame t_O into different windows $t_W < t_F$, whose duration t_W^* should be selected so as to minimize the forecast error.

At the same time, interpretation of Campus results is more complex, and undermine the generality of the above observation. Considering p_{95} estimation, it can be seen that Fig. 4 would suggest the use of small ($t_W \leq 4$) or even large ($t_W \geq 32$) windows – but not intermediate values of t_W, as the RMSE is concave in t_W. Fig. 5 states that smaller errors can be achieved by using large $t_W \geq 32$ windows (and consequently, a smaller number of features), which does not contradict our intuition, despite t_W value do not precisely correspond to the optimum of Fig. 4. Prediction results of p are of harder interpretation, as the minimum error can be achieved when the observation time frame is *not* split (i.e., $t_W = 64$), despite the prediction error monotonously increases with $t_W = t_F$ as shown in Fig. 4. Thus, it seems as though the specific network scenario may play a very significant role in determining SVR forecast performance, and that moreover this happen in a non trivial way. A possible, partial, explanation lies in the fact that shorter windows t_W translate into a higher number of features, which as early noted tends to "confuse" the SVR prediction: therefore, there may be cases where this increased number of features simply offsets the potential benefits brought by the splitting procedure.

C. Traces and traffic breakdown impact

In this section, we analyze the different traces of Tab. I in the attempt to quantitatively bound the extent of SVR benefits. We limit the analysis to the peak load estimation at $t_F = t_O = 64\,\text{s}$ and report in Tab. III the naive RMSE (R), as well as the maximum gain (G%) brought by SVR (over all possible window values t_W). First of all, the table reports the coefficient of variation (CoV), defined as the ratio of the standard deviation over the mean load and measured at two different timescales, as a statistical compact index of the link load variation. More specifically, CoV(∞) is evaluated over the whole trace, and represents the long-term load variability, whereas CoV(64 s) is the mean CoV evaluated over $64\,\text{s}$ long windows and is representative of the traffic "burstiness" at short timescales. Notice that there is only a weak correlation between the CoVs and the forecast accuracy: for instance, RMSE for Campus is the highest despite its CoV is lower than both WiFi or Enterprise ones – which is reasonable since relevant forecast errors, corresponding to Campus load spikes, are quadratically penalized by the RMSE metric.

In order to test whether forecast accuracy not only depends on the traffic trace but on the traffic class as well, we partition each trace into different traffic aggregates and apply the SVR load estimation to each aggregate separately. Specifically, we first extract all "server" traffic from the trace, considering only those packets that involve well-known transport layer ports, and then extract a second aggregate constituted by Web traffic only (i.e., port 80). Aggregates have a very different importance depending on the trace we are considering: e.g., ISP traffic volume is dominated by p2p traffic, while server traffic plays a major roles in the Enterprise trace and HTTP constitutes the dominant fraction of the WiFi dataset. Tab. III details the traffic breakdown across traces reporting the relative volume, expressed in terms of bytes (B%) and flows (F%), of Server/Total and Web/Server traffic. We stress that this difference in the relative volume of traffic aggregates affects the RMSE value, which indeed also depends on the actual *scale* of the data. For example in the case of Web traffic for the ISP trace, the fact that RMSE is very small is also tied to the fact that Web browsing only represents a small fraction of the total traffic. For this reason, and to allow a more direct comparison across traffic aggregates, the table also reports the gain in percentage with respect to the naive prediction method.

From Tab. III, is easy to gather that the gain over the naive estimation ranges between 10% and 30% , with an average

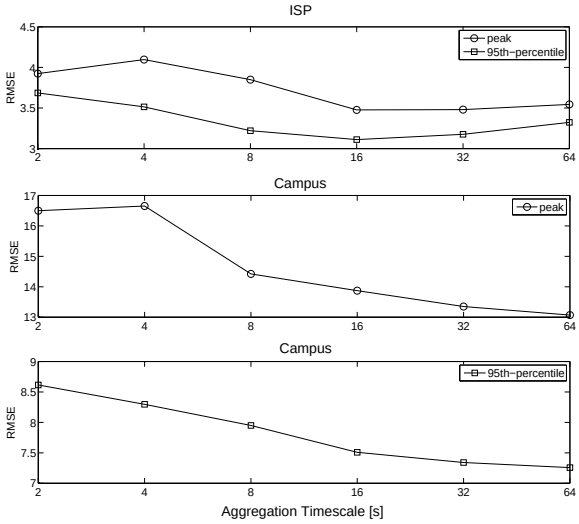

Fig. 5. Impact of the window timescale t_W on forecast accuracy

TABLE III
NAIVE RMSE ERROR (R) AND SVR GAIN (G%) FOR DIFFERENT TRACES (I,C,W,E) AND FOR DIFFERENT TRAFFIC BREAKDOWN IN TERMS OF SERVICE TYPE (ALL, SRV, WEB), BYTES (B%) AND FLOWS (F%)

	CoV		All		Srv/All		Services		Web/Srv		Web	
	∞	64 s	G%	R	B%	F%	G%	R	B%	F%	G%	R
I	0.29	0.05	12	3.7	1	2	22	1.1	8	2	15	0.5
C	0.40	0.17	21	13.4	44	21	19	14.2	66	38	18	12.5
W	0.88	0.54	11	10.6	92	89	11	10.6	91	80	10	10.6
E	0.64	0.56	25	10.2	62	37	27	10.0	27	33	10	4.9

TABLE IV

BEST AND WORST CHOICE AND PERCENTAGE OF MOST DISTANT AND CLOSEST TO THE MEAN PREDICTION

	Best	Worst	Most distant	Closest
ISP	(3) - 100%	(2) - 100%	(2) - 39%	(4) - 37%
Campus	(3) - 76%	(2) - 100%	(1) - 38%	(4) - 34%
WiFi	(3) - 69%	(1) - 100%	(1) - 60%	(2) - 35%

TABLE V

GAIN OF PARALLEL SVR OVER THE BEST AND THE WORST INPUT

	ISP		Campus		WiFi	
Strategy	Best	Worst	Best	Worst	Best	Worst
I	-0.5%	5.4%	0.1%	2.1%	0.1%	6.8%
II	-0.1%	5.7%	0.4%	2.4%	0.1%	6.8%
III	0.0%	5.8%	0.1%	2.1%	0.2%	6.9%

of 17% – which is clearly a very significant gain. Moreover, results confirm that the specific dataset considered strongly affects the SVR performance, whereas the forecast accuracy does not exhibit a significant variation across the different traffic aggregates[3]. An important remark is that the optimal t_W^* value (i.e., the one reported in Fig. 4 to which the most accurate prediction when $t_W = t_F = t_O$ corresponds) only varies across *traces*, but remains the same over all *aggregates* for a given trace. More precisely, the values of t_W^* that yield to the highest gain are $t_W = 64\,\text{s}$ for Campus, $t_W = 16\,\text{s}$ for WiFi or ISP and $t_W = 2\,\text{s}$ for Enterprise respectively.

Unfortunately, as early pointed out, there is no generally applicable guideline to properly select t_W. A possible strategy could be to use the *largest* possible window $t_W = 64$, which correspond to the *most concise* summary. A natural question is then to quantify the accuracy loss whether $t_W = 64$ is used instead of t_W^*. Interestingly, the loss in SVR accuracy under this sub-optimal heuristic choice is actually very limited. Specifically, results are on average only 4% above the optimum, with a maximum of 11% for Enterprise traffic: furthermore, if we neglect the latter (too short) dataset, the mean accuracy loss drops to a very limited 2%. Therefore, we can conclude that, although properly setting the SVR parameters may be a cumbersome task, at the same time SVR models are robust enough to have very good performance even under non optimal settings.

V. PARALLEL SVR

In this section we study the performance obtained by using several SVMs in parallel, i.e. for the same output several machines are trained and the final prediction is a *combination* of the results obtained by the single machines. The idea is that the use of several parallel machines can increase the forecast power. In this case we limit ourselves to consider the peak p as output. In particular, a different SVM is trained for each of the following inputs: $(1) = (\mu)$, $(2) = (\mu, \sigma)$, $(3) = (\mu, p)$ and $(4) = (\mu, p_{95})$ (which are chosen according to Tab. II). Different predictions are obtained form these machines and combined according to the following strategies:

I. the final prediction is the average of all predictions except the one that obtained the worst RMSE over the validation set
II. idem, but for each point in the validation set, we neglect the prediction that is *furthest* from the mean prediction of all machines
III. idem, but the prediction is defined as the one that is *closest* to the mean prediction

For the first case (I) the final prediction is defined depending on the performance (RMSE) of the different machines over all the validation set: the same combination is used for all points in this set. Conversely, in the last two cases the best combination is redefined for each point, i.e. the decision can be taken online with only punctual knowledge of the system.

It is our aim here to compare the performance of these three strategies with respect to the best and worst input combination, which possibly changes between datasets and outputs. Thus, for a given validation set, we determine the best and worst machine (in terms of RMSE). For each point in the validation set, we also identified which was the machine that predicted the value nearest and furthest to the mean of all predictions. This procedure is repeated 200 times and results are shown in Tab. IV for ISP, Campus and WiFi datasets. The first two columns indicate the machine (or features combination) that obtained the best and worst RMSE in the majority of the repetitions (this majority is shown as a percentage in the table). The last two columns indicate the machine whose prediction was the closest and the most distant for the majority of points in the validation set and repetitions (again this majority is shown as a percentage).

Some remarks are in order. Regarding the first strategy, we find that the best input changes over the different repetitions, in accordance with the results already shown in Tab. II. However the worst machine does not change, motivating the first considered strategy (I). More in details, since the worst machine is different across the traces, but not changes over a particular trace, a startup phase can be used to evaluate the worst machine which is then not considered for future predictions. Moreover we find out that the best (worst) input not necessarily coincides with the closest (most distant) predictor. This last observation justifies the consideration of the last two strategies (II and III).

We report in Tab. V the gain of using parallel SVMs over the best and worst results obtained by a single SVM. Performance are very similar for all the strategies, with strategy III obtaining slightly better results. Interestingly, *any* strategy employing parallel SVM is also better than a single machine using all the input (μ, σ, p, p_{95}). Moreover, with respect to the best 2-feature input, the results of parallel SVR are generally better but sometimes worse (i.e. strategies I and II in the ISP case), with a difference always smaller than 0.5%. This means that although no real gain is obtained, at the same time no noticeable performance loss happens either.

[3]The most important variations appears for the Enterprise traffic, but the results may be negatively biased by the excessively short trace duration.

TABLE VI
GAIN OF PARALLEL SVR OVER THE "EMBEDDING PROCEDURE"

ISP		Campus		WiFi	
RE	RMSE	RE	RMSE	RE	RMSE
11.6%	10.3%	25.7%	14.8%	12.2%	11.8%

In other words, the use of several machines in parallel can be an excellent strategy when the best input is not clear. A possible objection is that this approach may raise the computational cost of training several machines at the same time: to this extent, we stress that a common PC can support *several thousand* of such SVR machines in real-time [6].

VI. COMPARISON WITH "EMBEDDING PROCEDURE"

Finally, we compare the performance of parallel SVM and the "embedding process" described in [6]. For this purpose we considered again the dataset ISP, Campus and WiFi, and the peak as the output, since it is the most difficult to predict. Given the results of the previous section, for parallel SVM we limit ourselves to the strategy III. Roughly, with this comparison we aim at knowing which approach is better to predict the maximum load in the next minute: the "embedding process" which uses as input a vector containing all the observed samples ($x \in \mathbb{R}^d$ with $d = 64$), or the approach presented in this paper which uses an intelligent combination of statistical summary as input.

Results are shown in Tab. VI, averaged over 100 repetitions, where in each case 60% of the data is used as training and the remainder as validation set. Considering both the relative error (RE) or the RMSE, results indicate that for all traces an important gain can be obtained with the new approach presented in this paper: 15% on average and 10% in the worst case. It can be observed that a substantial RE gain is obtained for the Campus trace, which we already knew to be a stiffer scenario. This is an encouraging result, which shows that the drawbacks of the "embedding procedure" can be overcome, confirming that SVM is a very interesting technique for the purpose of link load prediction.

VII. DISCUSSION AND CONCLUSION

In this paper, we apply Support Vector Regression to the purpose of link load prediction: investigating the impact of several parameters on the forecast accuracy and considering several real-work traces we gather results that are representative of rather different network environments.

Our main result is that an "intelligent" combination of several parallel SVR using different statistical summary as input can provide accurate prediction. Indeed, we observe a significant gain not only over naive estimation technique, but also with respect to time-series based SVM predictions [6] as well. These findings make necessary to compare SVR with other forecast techniques, such as e.g., Nadaraya-Watson [15], which we leave for further work.

This work further provides some useful insights to tune SVR predictors: for instance, as a rule of thumb, SVR accuracy improves when "compact" statistical summaries are used as inputs. Then, despite the best input feature set possibly depends on the forecast output, we have shown that the selection can be avoided by using parallel SVMs. Concerning time-related parameters, our experiments show that their optimal tuning may not be easy task. Yet, we point out that the best setting only depends on the network measurement point but is insensitive to the traffic breakdown. Second, the use of large input time-scales (corresponding to compact input summary) yield to a near-optimal forecast accuracy . Third, we argue that the use of parallel SVMs could bring benefits also concerning the tuning of time-related parameters.

Overall, we can conclude that SVR is an interesting technique for its flexibility, cost-effectiveness, accuracy and robustness. SVR is flexible since it applies to the forecast of different targets, while cost-effectiveness stems from the fact that both the offline training and the online forecast operations have a linear cost in the number of support vector. Moreover, given that a number of software tools are readily available, SVR techniques can be deployed right away. Finally, SVR has the potential to bring accurate results under a number of different scenarios: while parameter tuning may not be a trivial task, we stress that SVR performance are rather robust even under non optimal settings and that the use of parallel SVM helps in relieving this problem.

REFERENCES

[1] J. Beran, "Statistics for Long-memory Processes," Chapman & Hall, London, 1994

[2] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the Self-Similar Nature of Ethernet Traffic", *IEEE Transactions on Networking*, Vol. 2, No. 1, pp. 1–15, 1994.

[3] B. E. Boser, I. M. Guyon, and V. N. Vapnik. "A training algorithm for optimal margin classifiers". *ACM COLT'92*, Pittsburgh, PA, 1992

[4] Alex J.Smola and B. Scholkopf, "A tutorial on support vector regression" In *Statistics and Computing*, Kluwer Academic, Publishers. 2004.

[5] A. Smola, G. Ratsch, B. Scholkopf, J. Kohlmorgen, V. Vapnik, "Predicting time series with support vector machines", In *Artificial Neural Networks*, Springer, Berlin, 1999

[6] P. Bermolen and D.Rossi, "Support Vector Regression for Link Load Prediction", to appear in *Elsevier Computer Networks.*

[7] P. Bermolen and D.Rossi, "Network Forecast with Support Vector Machines", extended abstract in *Workshop on IP QoS and Traffic Control*, Lisbon, Portugal, Dec.2007.

[8] M. Mirza, J. Sommers, P. Bardford, X. Zhu "A machine learning approach to TCP throughput prediction," In *ACM SIGMETRICS'07*, San Diego, CA, Jun. 2007,

[9] R. Beverly, K. Sollins and A. Berger, "SVM Learning of IP Address Structure for Latency Prediction", In *ACM SIGCOMM Workshop on Mining Network Data (MineNet'06)*, Pisa, Italy, Sep. 2006

[10] V.Vapnik, "The nature of statistical learning theory", Springer, NY, 1995.

[11] R.-E. Fan, P.-H. Chen, and C.-J. Lin. "Working set selection using the second order information for training SVM", *Journal of Machine Learning Research 6*, pp. 1889–1918, 2005

[12] M. Aizerman, E. Braverman and L. Rozonoer, "Theoretical foundations of the potential function method in pattern recognition learning", *Automation and Remote Control*, Vol. 25, pp. 821-837, 1964

[13] V. Paxson, R. Pang, M. Allman, M. Bennett, J. Lee and B. Tierney, "LBNL/ICSI Enterprise Tracing Project (collection)", available at `http://imdc.datcat.org/`

[14] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, B. Tierney "A First Look at Modern Enterprise Traffic", In *ACM Internet Measurement Conference (IMC'05)*, Oct. 2005

[15] Nadaraya, E . A. " Nonparametric estimation of probability densities and regression curves", *Mathematics and its Applications* Vol. 20, 1989.