Short-Term Traffic Flow Forecasting: An Experimental Comparison of Time-Series Analysis and Supervised Learning

Marco Lippi, Matteo Bertini, and Paolo Frasconi

***Abstract*—The literature on short-term traffic flow forecasting has undergone great development recently. Many works, describing a wide variety of different approaches, which very often share similar features and ideas, have been published. However, publications presenting new prediction algorithms usually employ different settings, data sets, and performance measurements, making it difficult to infer a clear picture of the advantages and limitations of each model. The aim of this paper is twofold. First, we review existing approaches to short-term traffic flow forecasting methods under the common view of probabilistic graphical models, presenting an extensive experimental comparison, which proposes a common baseline for their performance analysis and provides the infrastructure to operate on a publicly available data set. Second, we present two new support vector regression models, which are specifically devised to benefit from typical traffic flow seasonality and are shown to represent an interesting compromise between prediction accuracy and computational efficiency. The SARIMA model coupled with a Kalman filter is the most accurate model; however, the proposed seasonal support vector regressor turns out to be highly competitive when performing forecasts during the most congested periods.**

***Index Terms*—Intelligent transportation systems, support vector machines, traffic forecasting.**

I. Introduction

In RECENT years, traffic forecasting has become a crucial task in the area of intelligent transportation systems (ITSs), playing a fundamental role in the planning and development of traffic management and control systems. The goal is to predict traffic conditions in a transportation network based on its past behavior. Improving predictive accuracy within this context would be of extreme importance, not only to inform travelers about traffic conditions but also to design and realize infrastructures and mobility services and schedule interventions. For this reason, in recent years, there has been great effort in developing sensor instruments supporting traffic control systems [1]. These detector systems usually provide measurements about flow (or volume), speed, and lane occupancy within a transportation network. Prediction problems can therefore be differentiated according to the observed—and predicted—physical quantities. Flow prediction and travel time delay estimation are the two tasks that, throughout the years, have received most of the attention [2], [3].

Formally, given a sequence $\{X_1, \ldots, X_t\}$ of some observed physical quantity (such as speed or flow) at a given node of a transportation network, the forecasting problem consists in predicting $X_{t+\Delta}$ for some prediction horizon Δ. Throughout the years, a wide variety of methods for this kind of task has been developed in several different research areas, ranging from statistics to machine learning and from control systems to engineering. Some of these methods very often share similar ideas and face similar problems, but unfortunately, comparisons are made difficult by the use of a different terminology and of different experimental settings, due to the heterogeneous research areas in which many of these approaches originated.

In trying to aggregate these techniques into groups sharing common characteristics, a first set of methods can be individuated as the family of classical time-series approaches. The famous work of Box and Jenkins [69] played a fundamental role in the development of this research area, particularly by popularizing a specific class of these methods, which comprises combinations and generalizations of autoregressive moving average (ARMA) models. Starting from this work, a number of different models originated. In the application field of traffic forecasting, we can cite autoregressive integrated moving average (ARIMA) [4]–[6], which is an integrated version of ARMA, including stationarity in the model, and seasonal ARIMA (SARIMA) [7] capturing the periodicity that is typical of many time-series processes. Throughout the years, the ARIMA model has been used as the basis for a wide number of variants. The KARIMA model [8] uses a two-stage architecture, where a first level performs clustering between time-series samples, aggregating them using a self-organizing map (or Kohonen map) [9]. Upon each cluster in this map, an ARIMA model is finally built. Vector ARMA (VARMA) [10] and space-time ARIMA (STARIMA) [11] are two models that have been proposed for short-term traffic flow forecasting in multiple nodes within a transportation network, employing multivariate models. Whereas VARMA is a general extension of ARMA, which captures linear dependence relations

Manuscript received February 10, 2012; revised July 27, 2012 and November 25, 2012; accepted January 30, 2013. This work was supported in part by the Foundation for Research and Innovation, University of Florence, under Grant SSAMM-2009 and in part by the Italian Ministry of Education, University, and Research under PRIN project 2009LNP494. The Associate Editor for this paper was W.-H. Lin.

M. Lippi was with the Department of Systems and Informatics, University of Florence, Florence 50121, Italy. He is now with the Department of Computer Engineering and Mathematical Sciences, University of Siena, Siena 53100, Italy (e-mail: lippi@dii.unisi.it).

M. Bertini and P. Frasconi are with the Department of Information Engineering, University of Florence, Florence 50121, Italy (e-mail: mbertini@dsi.unifi.it; p-f@dsi.unifi.it).

Digital Object Identifier 10.1109/TITS.2013.2247040

between multiple time series using $N \times N$ parameter matrices to model dependence relations among the different time series, STARIMA is capable of working with fewer parameters, and therefore, it is more suitable for large-scale applications of traffic flow forecasting. Another class of time-series models widely used in many contexts, although particularly in financial forecasting, is given by volatility models, such as autoregressive conditional heteroskedasticity or generalized autoregressive conditional heteroskedasticity [12], for which Robert Engle was awarded the Nobel Prize for Economics in 2003. An application of GARCH to modeling volatility in urban transportation networks is given in [13].

From the point of view of machine learning algorithms, several methodologies have been applied to the task of traffic forecasting. The two most-used approaches are certainly artificial neural networks (ANNs) [14] and support vector regression (SVR) [15]. They are both supervised learning algorithms that can be trained to learn a function between some input features and the output, which is represented by the target to be predicted. Historically, ANNs are among the first machine learning methods that have been applied to traffic forecasting. For this reason, there exists a huge number of publications, which either propose different ANN-based architectures applied to this task or use ANNs as a baseline competitor against diverse classes of methods (e.g., see [16] or [17]). In [18], SVR with a radial basis function (RBF) kernel has been used to predict travel time in a transportation network, achieving good performance in comparison with trivial predictors such as the seasonal mean (SM) or a random walk (RW). In [15] and [19], an online version of SVR is employed to solve the same problem; however, both these works show some limitations in the experimental setup. Whereas in the first case, the proposed approach is only compared against ANNs (for which no model selection procedure is described), in the second case, only a few days are considered as test set examples, and there is no direct comparison against classical time-series approaches. The other machine learning approaches worth mentioning are Bayesian networks [20], echo-state networks [21], nonparametric regression [22], [23], and approaches based on the genetic algorithm, which has been often coupled with ANNs [24], [25]. In [26], a hybrid layered system made up of a combination of a Kohonen map with several neural networks is proposed. Even machine learning has recently tried to extend methodologies toward the multivariate case, using methodologies coming from statistical relational learning (SRL), which combines first-order logic representation and probabilistic graphical models. An example of application to collective traffic forecasting is given in [27].

In the context of control systems and automation engineering, Markov chains [28] and, particularly, Kalman filters [29], [30] are the methods that have shown the best experimental results.

All these methods have been used not only in traffic forecasting but also in several other application fields, such as economic series analysis [31], ecology and environment monitoring [32], [33], industrial processes control systems [34], and many others. However, there are only a few works in the literature in which a direct comparison between these different categories of approaches is given [35]. Therefore, the aim of this paper is twofold. First, we present an extensive review of the most popular short-term traffic flow forecasting methods, proposing a common view using the formalism of graphical models and showing a wide experimental comparison. Second, we introduce a new supervised learning algorithm based on SVR, which is capable of exploiting the seasonality of traffic data, as models such as SARIMA do, resulting in an interesting compromise between predictive accuracy and computational costs.

This paper is structured as follows. In Section II, we present a unifying view of the many developed approaches to time-series forecasting, using the formalism of graphical models. Then, in Section III, we introduce seasonal kernels for SVR. An extensive experimental comparison ranging over a wide variety of predictors is given in Section IV. Finally, the conclusion and future work are presented in Section V.

II. Unified View for Short-Term Traffic Forecasting Methods Using Graphical Models

The literature on short-term traffic forecasting covers a broad spectrum of research areas including statistics, control systems, and computer science. Unfortunately, too many works in this field lack comparison with similar methodologies originated in different research areas, and this is true both for theoretical and experimental comparisons.

From the statistical point of view, both frequentist and Bayesian approaches have been extensively studied. In this paper, we only take the frequentist perspective (for applications of Bayesian methods to traffic forecasting, see, e.g., [36]–[38]). In addition, we restrict our focus to univariate time series.

A. Representation

We give a unified view of the most commonly used approaches to short-term traffic forecasting using the formalism of graphical models, trying to highlight similarities and differences between all methods.

A graphical model is a probabilistic model, where nodes represent random variables, and edges connect nodes that are conditionally dependent. The two most used categories of graphical models are given by Bayesian networks (which employ directed acyclic graphs) and Markov networks (represented by undirected graphs).

We denote by $\{X_1, \ldots, X_t\}$ and $\{\eta_1, \ldots, \eta_t\}$ two sequences of random variables representing the observed time series and a random noise process, respectively. Different models originate from different assumptions on the class of probability distributions from which these variables are drawn.

The simplest modeling approach is perhaps the autoregressive structure. For a given integer p, in autoregressive models, it is assumed that X_t is conditionally independent of $\{X_1, \ldots, X_{t-p-1}\}$ and of past noise, when $\{X_{t-1}, \ldots, X_{t-p}\}$ are given. The independence structure is illustrated through a directed graphical model in Fig. 1, where black nodes represent the observed time series, gray nodes represent the unobserved noise process, and the white node represents the random variable that is to be predicted. Under this model, noise variables η_t are marginally independent (a white process) but

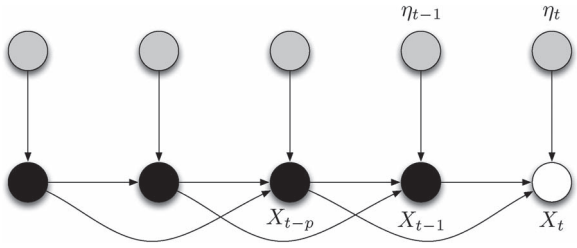

Fig. 1. Independence relations in an autoregressive model with $p = 2$. Gray nodes represent hidden variables, modeling the white noise process η_t. Black nodes represent the observed past time series. White node represents the target random variable.

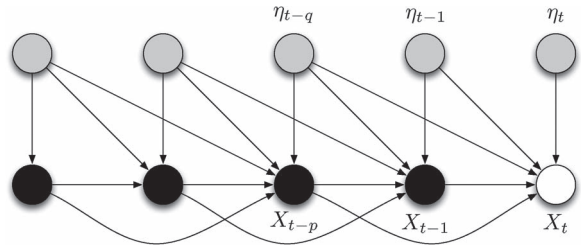

Fig. 2. Independence relations in ARMA models.

conditionally interdependent given the observed time series. Model specification is completed by defining a structure for the conditional probability $p(X_t|X_{t-1}, \ldots, X_{t-p}, \eta_t)$. The usual choice is

$$X_t \sim \mathcal{N}\left(f(X_{t-1}, \ldots, X_{t-p}), \sigma\right) \tag{1}$$

where σ is the noise variance and f is a regression function. Many of the most-used algorithms in machine learning and statistical methods in classical time-series analysis can be derived from this formulation. In the simplest case, f is a linear combination of past observations, i.e.,

$$f(X_{t-1}, \ldots, X_{t-p}; \psi) = \psi_0 + \sum_{i=1}^{p} \psi_i X_{t-i} \tag{2}$$

where $\boldsymbol{\psi}$ is a vector of adjustable parameters. This linear model is typically indicated as AR(p). More complicated models incorporate nonlinear dependence relations. One possibility is to represent f as a feedforward neural network (see, e.g., [39] for an application to traffic problems). In this case

$$f(X_{t-1}, \ldots, X_{t-p}; \psi, \mathbf{W}) = \psi_0 + \sum_{j=1}^{h} \psi_j \phi_j(X_{t-1}, \ldots, X_{t-p}; \mathbf{W}) \tag{3}$$

where h is the number of hidden units, $\mathbf{W}$ is a weight matrix, and each "feature" ϕ_j is a nonlinear function of past observations, which depends on further adjustable parameters $\mathbf{W}$, i.e.,

$$\phi_j(X_{t-1}, \ldots, X_{t-p}; \mathbf{W}) = \tanh\left(\sum_{i=1}^{p} w_{ij} X_{t-i}\right). \tag{4}$$

A second possibility is to obtain nonlinearity via kernel function K that measures the similarity between the current sequence of past p observations and another sequence of p observations, i.e.,

$$f(X_{t-1}, \ldots, X_{t-p}; \alpha) = \sum_{\tau} \alpha_\tau K\left(\{X_{t-1}, \ldots, X_{t-p}\}, \{X_{\tau-1}, \ldots, X_{\tau-p}\}\right) \tag{5}$$

where $\boldsymbol{\alpha}$ are adjustable parameters. The most popular choice for K is the RBF kernel (see, e.g., [18], [40], and [41] for applications to traffic problems), i.e.,

$$K(a, b) = \exp\left(\gamma \|a - b\|^2\right) \tag{6}$$

where γ is a parameter controlling the complexity of the function space from which f is chosen. One more possibility is to take a nonparametric approach, where the number of parameters is not fixed but is allowed to grow with data. Popular methods include nonparametric regression (see, e.g., [22] and [42]–[44]) and locally weighted regression (see, e.g., [45] and [46]). A comparison between parametric and nonparametric approaches is given in [47].

Moving average models capture the dependence relations between the random variable to be predicted and the unobserved noise process. More precisely, a moving average model of order q estimates the continuation of a time series by modeling conditional probability $p(X_t|\eta_{t-1}, \ldots, \eta_{t-q})$. Moving average models are less interpretable than autoregressors, as the error terms against which the coefficients are fitted are not observable. In the formalism of graphical models, white node X_t is connected only to gray nodes $\eta_{t-q}, \ldots, \eta_t$ so that the moving average model estimates $p(X_t|\eta_{t-q}, \ldots, \eta_t)$.

Within the context of time-series analysis, the ARMA and ARIMA models [4] are probably the most popular. An ARMA(p, q) model combines an autoregressive model of order p with a moving average process of order q, whereas an ARIMA(p, d, q) model also includes a differentiated component of order d, to handle nonstationarity, i.e.,

$$\psi(B)(1 - B)^d X_t = \omega(B)\eta_t \tag{7}$$

where B is the back-shift (or lag) operator ($B^k X_t = X_{t-k}$), and ψ and ω are the two polynomials for the AR and MA components, respectively. This class of methods has become widely used in many contexts (in addition to traffic forecasting), owing to the study presented by Box and Jenkins [69], which describes a detailed procedure to perform model selection and parameter fitting. It is straightforward to also represent ARMA and ARIMA using graphical models. Fig. 2, for example, shows the structure of ARMA(p, q), modeling conditional probability $p(X_t|X_{t-1}, \ldots, X_{t-p}, \eta_t, \ldots, \eta_{t-q})$.

A key feature early introduced in ARIMA models to improve predictive accuracy consists in taking into account the periodic nature of traffic data. This is the case with the SARIMA, which was first introduced by Box and Jenkins and has been successfully applied to short-term traffic forecasting in [7]. The SARIMA approach includes weekly dependence relations within the standard autoregressive model, by proving that the time series obtained as the difference between the observations in two subsequent weeks is weakly stationary. The core of the idea is that similar conditions typically hold at the same

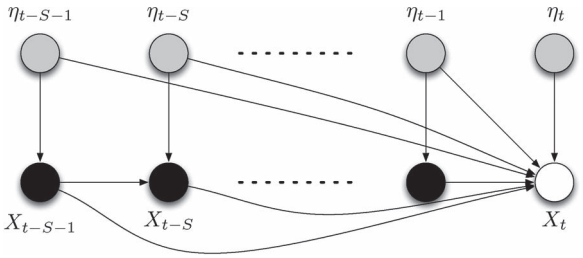

Fig. 3. Independence relations in a SARIMA $(1, 0, 1)(0, 1, 1)_S$ model.

hour of the day and within the same weekdays. The resulting SARIMA$(p, d, q) \times (P, D, Q)_S$ model adds to the standard ARIMA a seasonal autoregressive, a seasonal moving average, and a seasonal differential component, i.e.,

$$\psi(B)\Psi(B^S)(1-B)^d(1-B^S)^D X_t = \omega(B)\Omega(B^S)\eta_t \quad (8)$$

where Ψ and Ω are the two polynomials for the seasonal components. In this case, the graphical model has simply to be adapted to include the differentiated component, as it happens for the ARIMA case, and to represent the dependence with the past period. Fig. 3 shows the graphical model for the case of SARIMA$(1, 0, 1)(0, 1, 1)_S$, which is one of the most used versions of this method. The conditional probability, which is estimated in this case, is

$$p(X_t|X_{t-1}, X_{t-S}, X_{t-S-1}, \eta_t, \eta_{t-S}, \eta_{t-S-1}). \quad (9)$$

Classic SARIMA is a linear model; therefore, regression function f is computed as

$$\begin{aligned} f(&X_{t-1}, X_{t-S}, X_{t-S-1}, \eta_t, \eta_{t-1}, \eta_{t-S}, \eta_{t-S-1}; \psi, \omega) \\ &= \psi_0 + \psi_1 X_{t-1} + X_{t-S} + \psi_1 X_{t-S-1} + \eta_t \\ &\quad + \omega_1\eta_{t-1} + \omega_S\eta_{t-S} + \omega_1\omega_S\eta_{t-S-1}. \end{aligned} \quad (10)$$

The SARIMA model can be considered a state of the art within the category of classical time-series approaches.

B. Learning

In the frequentist setting, fitting a model to data involves optimizing a suitable objective function with respect to unknown parameters $\boldsymbol{\theta}$, i.e.,

$$\hat{\theta} = \arg\min_{\theta} \sum_t L\left(f(X_{t-1}, \ldots, X_{t-p}; \theta), X_t\right) + \lambda R(\theta). \quad (11)$$

Loss function L measures the discrepancy between the predicted and the observed physical quantity, whereas $R(\theta)$ is a regularization term that is used to avoid ill-conditioned problems and prevent overfitting.

A common choice for loss function L [see (2), (3), and (5)] is the square loss $L(a, b) = (a - b)^2$. In this case, (11) essentially implements the maximum-likelihood principle (assuming Gaussian data). Many commonly used regularization terms may be interpreted as log priors on the parameters. This corresponds to the maximum *a posteriori* approach to parameter fitting. For example, the L2 regularizer $\lambda R(\boldsymbol{\theta}) = \lambda\|\boldsymbol{\theta}\|_2$ may be interpreted as a Gaussian prior with zero mean and variance $1/2\lambda$.

In the case of SVR, the objective function is defined by using the ϵ-insensitive loss [48], i.e.,

$$L(a, b) = \begin{cases} 0, & \text{if } |a - b| \leq \epsilon \\ |a - b| - \epsilon, & \text{otherwise} \end{cases} \quad (12)$$

where ϵ is a given error tolerance. The name originates from the fact that support vectors are generated only when the error is above ϵ. Compared with the squared loss, this has the advantage of producing a sparse solution where many $\alpha_\tau = 0$ in (5). The associated optimization problem is convex [48].

It must be underlined that in ARMA and (S)ARIMA models with moving average components, observations are regressed on unobserved (hidden) variables. This is an important characteristic, which distinguishes these models from machine learning approaches such as neural networks, local weighted regression, or SVRs. Nevertheless, it is worth mentioning that Bayesian inference is also capable of capturing dependence relations with unobserved random variables.

Recently, SARIMA has been also used in combination with Kalman filter [49], which is employed to estimate the unobserved random variables, that is, the components of the moving average process. In this case, it is convenient to rewrite the SARIMA model in state space. Following [49], with reference to (10), we can define state vector $\mathbf{a}_t$ as:

$$\mathbf{a}_t = [\psi_0 \quad \psi_1 \quad \omega_1 \quad \omega_S] \quad (13)$$

and observation vector $\mathbf{z}_t$ as

$$\mathbf{z}_t = [1 \quad Y_{t-1} \quad -\eta_{t-1} \quad -\eta_{t-S}] \quad (14)$$

where Y_t is the differenced time series ($Y_t = X_t - X_{t-S}$). The Kalman filter is a recursive estimator of a discrete-time controlled process, which, for the problem considered, can be described by the following linear stochastic difference equation:

$$\mathbf{a}_t = \mathbf{a}_{t-1} + \nu_{t-1} \quad (15)$$

using a measurement

$$Y_t = \mathbf{z}_t\mathbf{a}_t^T + \omega_1\omega_S\eta_{t-S-1} + \eta_t \quad (16)$$

where ν_{t-1} is the transition error, having covariance matrix $\mathbf{Q}$.

The update procedure of Kalman filter computes both the state, employing (15), and the *a priori* estimated error covariance, which is computed following a random-walk update rule. After updating state $\mathbf{a}_t$, an *a posteriori* computation is performed, both for the state and the error covariance, which will be used in the subsequent *a priori* computation. The Kalman filter is known to be an optimal estimator for linear system models with additive independent white noise. In [49], the Kalman filter has been shown to be the best adaptive estimator when compared against adaptive filters such as the recursive least squares and the least mean squares filter, although only weekly seasonality is considered within the experimental evaluation. It is worth saying that, while learning algorithms

such as SVR or neural networks assume independent and identically distributed (i.i.d.) examples, ARMA processes or Kalman filters are capable of modeling interdependence relations in the time series.

C. Algorithmic Issues

Although presenting several analogies, all these approaches belonging to heterogeneous research areas differ in many aspects. In the following, we present some of these differences, highlighting advantages and disadvantages of some of the approaches previously described, and discussing several algorithmic issues.

1) On-Sample versus Out-of-Sample Model Selection: One of the main differences between classic time-series analysis and machine learning algorithms exists in the adopted procedure for model selection during the training phase. In many cases, time-series-based algorithms work *in-sample*, which means that they validate the models directly on the data used during parameter fitting [50]; on the other hand, the typical machine learning practice relies on *out-of-sample* model selection. In-sample procedures suggest using some tools, such as the autoregressive correlation function or the partial autoregressive correlation function plots, to select a reasonable set of parameter configurations [e.g., ARMA(1,0,0) or ARMA(0,1,1)], then fitting the model using all the available data, and finally picking the best model using some criterion, such as the Akaike information criterion [51] or the Bayesian information criterion [52], [53]. Nowadays, however, employing an *out-of-sample* procedure has become commonplace, by simply splitting the available data in two sets: the samples to be used for model fitting and the samples to be used for model evaluation. Several studies have analyzed the advantages of this kind of criterion for parameter selection [54], [55], despite a greater computational cost during the training phase. By employing an out-of-sample procedure, data are split into a *training set* and a *validation set*. Given a set of possible parameter configurations, different models are fitted on the training set and evaluated on the validation set to choose the best parameter configuration. Finally, performance is estimated on a third *test set* containing new cases, unseen during the training phase. These differences are mostly historical, as recent work has been done to better analyze the statistical properties of *out-of-sample* model selection [56] and to propose cross-validation as the preferred way of measuring the predictive performance of a statistical model.

2) On-Line versus Off-Line Algorithms: An algorithm is *off-line* if it is designed to have access to all the samples (even multiple times) and to fit the parameter in one shot. An *on-line* algorithm is designed to have access to every sample only once and to update the fitted parameters incrementally. Moreover, potentially any *on-line* algorithm can forecast, have the true value as feedback, and update the parameters. Examples of this are online SVR in [15] and [19] and variations of standard Kalman filters as in [29].

3) Forecasting Phase: The forecasting phase differences are strictly related to the correlated/independent example hypothesis. For example, in the case of plain autoregressive models, neural networks, or SVRs, if the examples are i.i.d., the forecasting machine operates as a mapping function. Given an input, it will calculate the output, and the output will be the same for any equal input window. This is not true, for example, when a Kalman filter is employed to estimate the state of the model. In the latter case, the forecast will depend both on the input and on the state of the model.

4) Multiple Information Sources: Machine learning algorithms can easily integrate, within a single model, information coming from different sources. In the case of traffic forecasting at a certain location, knowledge of the flow conditions in the surrounding nodes of the transportation network and the trend in some correlated time series can supply crucial information to improve the predictor. To integrate the benefits of these two different directions of research, several hybrid methods have been recently introduced, combining aspects and ideas from both perspectives. The KARIMA model [8], for example, first learns a Kohonen map [9] as an initial classifier and then builds an ARIMA model for each class individuated by the self-organizing map. In the direction of SRL [57], VARMA and STARIMA [11] and Markov logic network [27] models try to incorporate, in a single model, information coming from interrelated time series to perform joint forecasts at multiple nodes in the transportation network. Another recent approach in this sense, in the context of Bayesian approaches, is given by the multiregression dynamic model proposed in [58].

5) Training Set Dimension and Distribution Drift: Typically, machine learning algorithms take advantage of large amounts of data [59], their prediction errors usually being reduced as the dimension of the training set grows. Nevertheless, it is worth saying that, in time-series forecasting, the use of larger training sets does not necessarily imply the achievement of higher levels of accuracy. If the training set contains samples too far in the past, for example, the phenomenon of covariate shift can occur [60], and the distribution of the training data can differ too much from the distribution of the test data, producing a discrepancy that can overall degrade the performance of the predictor. As reported in Section IV, the performance of ARIMA and SARIMA is typically less dependent on the availability of large training sets.

6) Local Minima and Overfitting: Different from SVR, which solves a convex optimization problem, the learning algorithm for neural networks (the backpropagation algorithm [61] is usually employed) suffers the problem of local minima. Typically, a validation set is employed to measure the prediction error during training to stop the learning phase. This validation set, in fact, measures the generalization error of the neural network, since it is not used for gradient computation, which is employed to update the weights. In this way, the phenomenon of overfitting can be avoided [62], preventing the neural network from learning a set of weights, which has adapted too much to the training data but has weak predictive accuracy on unseen examples.

III. Seasonal Kernels

To exploit the seasonality of the traffic flow time series, following the SARIMA approach, we introduce a seasonal

kernel for SVR. The underlying idea is that, for example, to predict a time-series continuation at 3 P.M. using an SVR, past examples in the training set that were observed at 8 A.M. or 10 A.M. should be less informative than those registered between 2 P.M. and 4 P.M., because similar behaviors in the transportation network are more likely to happen within the same part of the day. We show here that this similarity criterion can be easily plugged into a kernel function, which can be used to weigh the contributions of the training samples, according to the observation timestamps. This seasonal kernel can be seen as an interesting compromise between the predictive accuracy of SARIMA and the computational efficiency of SVR.

A. RBF Seasonal Kernel

When using kernel machines to build a predictor, the choice of the proper kernel function is a crucial element to enhancing the overall accuracy. A common choice is to employ the RBF kernel, i.e.,

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right) \tag{17}$$

or a polynomial kernel, i.e.,

$$K(x_i, x_j) = (x_i \cdot x_j + 1)^d \tag{18}$$

which are both able to capture nonlinear dependence relations between features. However, it is often the case that an ad hoc kernel designed for a specific application achieves better performance, being able to measure similarity between objects in the domain in a more appropriate way.

In the case of time-series forecasting, the desired kernel should be able to capture the periodic nature of data, so that two examples are more similar if they describe events (i.e., time series) that happened in similar temporal situations, e.g., within the same part of the day. Following the idea of seasonality as in the SARIMA model, we substituted the RBF kernel with a newly designed similarity function $K_s((x_i, t_i), (x_j, t_j))$, where x_i and x_j are two objects representing the time series at absolute timestamps t_i and t_j, respectively. Assuming S as the seasonal period of the time series, we designed this new function as

$$K_s\left((x_i, t_i), (x_j, t_j)\right) = \exp\left(-\gamma \|x_i - x_j\|^2\right) \times \exp\left(-\gamma_S d_p(t_i, t_j)^2\right) \tag{19}$$

where $d_p(t_i, t_j)$, which is defined as

$$d_p(t_i, t_j) = \min\left\{d_s(t_i, t_j), S - d_s(t_i, t_j)\right\} \text{ with} \tag{20}$$

$$d_s(t_i, t_j) = |(t_i - t_j) \bmod S| \tag{21}$$

represents the temporal distance from samples x_i and x_j.

B. Linear Seasonal Kernel

A similar result can be obtained by combining the seasonal kernel with a linear kernel, in place of an RBF. The advantage of using a linear kernel is the possibility of exploiting fast training algorithms, such as stochastic gradient descent [63], which can solve the SVM optimization problem in primal space, by updating the model parameters after computing the loss function for each training example. In this framework, we used the following objective function:

$$f(x) = \sum_{p=1}^{S} \left\{ \left(\sum_{i=1}^{n} \sigma\left(p, \pi(i)\right) \cdot L\left(\langle x_i, w_p \rangle, y_i\right) \right) + \frac{\lambda}{2} \|w_p\|^2 \right\} \tag{22}$$

where S is the period, L is the ϵ-insensitive loss, y_i is the prediction target for example x_i, w is a matrix $w \in \mathbb{R}^{S \times m}$ (m being the number of features), λ is a regularization coefficient, and $\sigma(p, \pi(i))$ is a scaling factor incorporating the seasonality of the model. Basically, each row w_p in matrix w represents a different set of parameters for each interval p within period S. For each training example i, the loss function is rescaled according to the similarity $\sigma(p, \pi(i))$, where $\pi(i)$ is the position of example i within period S. Function $\sigma(\cdot)$ simulates Gaussian smoothing according to the following equation:

$$\sigma\left(p, \pi(i)\right) = \exp\left(-\kappa \|p - \pi(i)\|^2\right) \tag{23}$$

where κ is a tunable parameter. The key idea is that closer periods within a day typically share more similar behaviors than more distant periods, and their contribution in computing predictive function f should therefore be more significant. For example, a time-series window sampled at 4 P.M. should be more similar to another window sampled at 3 P.M. rather than a third window sampled at 7 A.M.

IV. Experiments

A. Data

The data used for our experiments are taken from the California Freeway Performance Measurement System (PeMS) [64], which is a very large database of measurements collected by over 30 000 sensors and detectors placed around nine districts in California. The whole system covers 164 freeways, with a total number of 6328 mainline vehicle detector stations and 3470 ramp detectors. Most of the sensors are single detectors, that is, one loop per lane per detector station, and typically measure raw vehicle count (traffic flow) and occupancy (amount of time the loop detector is active). Only in some cases can double-loop detectors directly measure instantaneous speed, which is otherwise artificially reconstructed. We collected nine months (January–September 2009) of flow data from a set of 16 stations[1] randomly selected from the whole California District 7 (LA/Ventura). The time series was smoothed by aggregating flow data into nonoverlapping 15-min intervals (15 min is an interval typically used in the literature as a reasonable tradeoff between signal and noise [39], [47], [49]). This results in 96 samples per day per station.

For each station, the first four months (January–April 2009) are used as a training period, the following two months (May–June 2009) are used as a validation period for model

[1]A station is an aggregation of detectors from different lanes in the same location.

TABLE I
SET OF 16 STATIONS USED IN OUR EXPERIMENTS: STATIONS USED FOR MODEL SELECTION (SHOWN IN BOLD)

Station	Highway	Lanes
715916	5 S	4
716312	91 E	6
716551	134 W	4
716841	605 N	4
716933	5 N	5
717087	10 W	4
717123	10 E	4
717152	10 E	4
717190	10 E	4
717269	60 W	5
718144	101 S	4
737344	10 E	4
763626	110 N	4
764151	5 S	4
767678	605 N	4
768682	5 N	5

selection and parameter tuning, and the remaining three months (July–September) are used as a test period. Only 5 of these 16 stations were used to perform model selection (see Table I).

We only included workdays in our data set. This is a quite common scenario in the literature (e.g., see [10], [11], [21], [23], and [26]). In addition, since the distribution of traffic flow during holidays is largely different with respect to workdays, by including only workdays, we had the opportunity of comparing daily and weekly seasonality. (Otherwise, daily seasonality would have been less significant when comparing a workday with a holiday.)

B. Experimental Setup

Our task consisted in predicting traffic flow 15 min ahead of a given timestamp. The set of tested competitors consists of the following algorithms:

1) RW, which is a simple baseline that predicts traffic in the future as equal to current conditions ($X_t = X_{t+1}$);
2) SM, which predicts for a given time of the day the average in the training set;
3) ARIMA model with Kalman filter ($\mathrm{ARIMA_{Kal}}$);
4) SARIMA model with maximum-likelihood fitting ($\mathrm{SARIMA_{ML}}$);
5) SARIMA model with Kalman filter ($\mathrm{SARIMA_{Kal}}$);
6) ANNs;
7) SVR with RBF kernel ($\mathrm{SVR_{RBF}}$);
8) SVR with RBF kernel multiplied by a seasonal kernel ($\mathrm{SVR^S_{RBF}}$);
9) SVR with linear seasonal kernel ($\mathrm{SVR^S_{lin}}$).

As for the two SARIMA approaches, they differ both in the forecasting phase and in the parameter-fitting algorithm. Concerning forecasts, $\mathrm{SARIMA_{Kal}}$ uses a Kalman filter as in [49], whereas $\mathrm{SARIMA_{ML}}$ solves a simple recursive equation as in [7]. As for the learning phase, $\mathrm{SARIMA_{ML}}$ minimizes the negative of the likelihood, whereas in $\mathrm{SARIMA_{Kal}}$, we followed the two approaches described in [65] and [66], to fit the Kalman state-space model. The first algorithm is used to compute the covariance matrix associated with the initial value of the state vector, whereas the second algorithm is used within the forecasting phase. Different from [49], covariance matrix

TABLE II
PARAMETERS OBTAINED BY MODEL SELECTION FOR THE MACHINE LEARNING ALGORITHMS. C, ϵ, AND γ ARE SVR PARAMETERS; γ^S IS THE SEASONAL KERNEL PARAMETER; N_H IS THE NUMBER OF HIDDEN UNITS FOR ANNS; ρ IS THE LEARNING RATE; T IS THE NUMBER OF ITERATIONS; AND w IS THE FEATURE WINDOW SIZE FOR ALL ALGORITHMS

Algorithm	Parameters
SVR_{RBF}	$C = 5$, $\epsilon = 0.01$, $\gamma = 5$, $w = 3$
SVR^S_{RBF}	$C = 5$, $\epsilon = 0.01$, $\gamma = 1$, $\gamma^S = 192$, $w = 3$
SVR^S_{lin}	$\lambda = 0.001$, $\epsilon = 0.01$, $w = 3$, $T = 150000$
ANN	$N_H = 10$, $\rho = 0.01$, $T = 100$, $w = 3$

Q is estimated at each step, rather than estimated once and kept unchanged (see [65] and [66] for details). A modified version of the `auto.arima` procedure in R [67] was employed in our experiments. As presented in [68], the algorithm AS 154 proposed in [65] has a storage requirement of $O(r^4/8)$, where r is, in this case, $r = S + 1$, S being the seasonality of the model. A model with a weekly seasonality has a period of $S = 96 \times 5 = 480$ (we consider a five-working-day week), which results in a huge storage requirement. The modified implementation computes the estimated covariance matrix P_t iteratively as the solution of $P_t = P_{t-1} + Q$, using a fraction of the required memory. Another possibility that can be tackled to restrict memory requirements is to split the data in five different partitions, that is, one for each weekday. This setting allows training a different model for each weekday with a daily seasonality, therefore reducing spatial complexity. This partitioning trick produces results comparable with the weekly periodic approach, and therefore, due to the computational requirements, it has been employed for $\mathrm{SARIMA_{Kal}}$ model fitting. The same procedure can be followed for all the other methods, which were therefore trained using either 1) a single data set containing all five working days (weekly seasonality); 2) five different data sets, that is, one for each weekday, producing five different models with daily seasonality; or 3) a single data set with all working days but trained considering a daily seasonality. After running model selection, solution 1 was chosen for $\mathrm{SVR_{RBF}}$, solution 2 was chosen for $\mathrm{SVR^S_{lin}}$ and ANNs, and solution 3 was chosen for $\mathrm{SVR^S_{RBF}}$.

The available features are: 1) the past flow time series, from which window w of the most recent samples is usually extracted; 2) the SM at the current timestamp; and 3) the SM at 15 min ahead. For all the tested versions of SVR and ANN algorithms, some parameters for feature selection were tuned during the validation phase. In particular, windows w of 3, 6, or 12 samples were tested, corresponding to 45, 90, or 180 min before the current time of the day. Moreover, all the algorithms were also tried with or without the SM at current and future timestamps in feature space. In all the models, such features improved prediction accuracy and were therefore included in the final models.

For $\mathrm{SVR_{RBF}}$, we performed model selection on C [$C = 1/2\lambda$ referring to (11)] and γ (γ being the tuning parameter for the smoothness of the decision boundary in the RBF kernel), whereas for $\mathrm{SVR^S_{RBF}}$, we also had to choose γ^S. $\mathrm{SVR^S_{lin}}$ needs model selection on λ [see (22)], whereas for the ANN, the number of hidden neurons and the learning rate were tuned in the validation phase as well. Table II shows the choice

TABLE III
MAPE$_{100}$ ERROR WITH COMMON OR SPECIFIC PARAMETER CONFIGURATIONS

Station	SVR$_{RBF}$		SVR$^{S}_{RBF}$	
	specific	common	specific	common
716933	5.601	5.669	5.542	5.546
717087	5.255	5.293	5.154	5.173
717123	5.012	5.038	4.966	5.013
717152	5.486	5.486	5.385	5.412
717190	5.300	5.316	5.161	5.167

of parameters obtained by model selection for these machine learning algorithms.

The model selected for SARIMA is $(1, 0, 1)(0, 1, 1)_S$, as proposed in [7], and the best setting is that with five different models (one for each weekday) for the Kalman state-space model and that with a single weekly period for SARIMA$_{\mathrm{ML}}$. The model selected for ARIMA is the model proposed by the `auto.arima` procedure in R [67] with a single model for all weekdays.

All tests were run on a 3-GHz processor with 4-MB cache and 16 GB of RAM.

C. Results and Discussion

We ran a first experiment with the aim of comparing plain RBF kernels with seasonal kernels and, at the same time, testing the robustness of kernel parameters over multiple stations in our data set. In all our experiments, we measured MAPE$_{100}$ as in [49], where MAPE is the mean absolute percentage error, which is measured as follows:

$$\mathrm{MAPE} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{X_t - \hat{X}_t}{X_t}\right| \tag{24}$$

where $\hat{X}_t$ is the predicted flow. In MAPE$_{100}$, the sum is computed only over the terms where $X_t > 100$ vehicles/hour, therefore considering in the error measurement only those periods where traffic flow is significant. MAPE$_{xx}$ is typically used in traffic forecasting to avoid the problem that few examples with low traffic flow might greatly affect error measurement. This would happen, for example, at nighttime when few vehicles per hour are detected.

Table III shows that SVR with a seasonal kernel achieves higher accuracy than plain SVR with an RBF kernel. The choice of a common parameter configuration has a slight effect on the performance of the resulting classifier. It is worth mentioning that using a single common parameter configuration, the time-consuming model selection phase can be skipped when training on new stations, dramatically reducing the computational requirements. When comparing the seasonal kernel against the RBF kernel on a larger test set (see Table V), the advantage is statistically significant according to a Wilcoxon paired test with a p value of < 0.01.

To measure the impact of the training set dimension on the different algorithms, we built a learning curve by measuring the performance of each classifier as a function of the number of training months. Fig. 4 shows the learning curve of each algorithm, whereas Table IV gives all the details on these

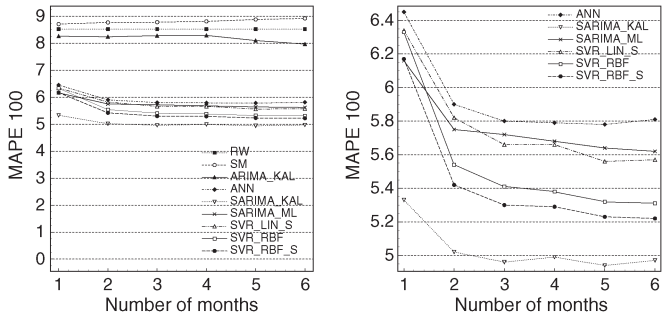

Fig. 4. Learning curves of the compared algorithms, reporting prediction error (MAPE$_{100}$) as a function of the number of months used in the training set. (Left) All algorithms. (Right) Zoom on best performing algorithms.

experiments. SARIMA$_{\mathrm{Kal}}$ is the best model in terms of accuracy; however, its performance does not improve when including in the training set more than two months; SVR$^{S}_{\mathrm{RBF}}$ is the second best model and can take advantage of a larger training set. Nevertheless, after three months of training, none of the algorithms showed a statistically significant improvement, and therefore, in all the subsequent experiments, we adopted a training set of three months, which is a good compromise between predictive accuracy and training time. The fact that adding more examples in the training set does not yield an improvement in prediction accuracy is possibly due to the covariate shift [60] already cited in Section II. This is confirmed by the behavior of the SM predictor, which quickly degrades its performance both employing larger training sets (as shown in Fig. 4) and in the case of wider horizon lags. When performing forecasts at further prediction horizons (i.e., predicting at K months ahead, keeping the training set fixed), in fact, the worsening in performance of the SM predictor is extremely marked. For example, on station 716933, MAPE$_{100}$ grows from 8.7, when forecasting at one month ahead, to 11.1, when the lag grows to five months. A deeper analysis of this covariate shift will be the subject of further investigations in future works, as understanding this phenomenon might play a crucial role in adapting the predictors to the dynamics of the transportation network. Table V reports the results of each algorithm, detailed per station. It should be noticed that SARIMA$_{\mathrm{ML}}$ performs worse than both SARIMA$_{\mathrm{Kal}}$ and SVR with an RBF kernel. In addition, ANNs are consistently less accurate than plain SVR.

We also analyzed the performance of each algorithm during periods that typically present high levels of congestion. We chose the morning peak between 7 A.M. and 8 A.M. and measured the prediction error for each test station, which is reported in Table VI. The very high values of MAPE of the RW and SM predictors indicate difficulty of the task. It is interesting to notice that in this case, SARIMA$_{\mathrm{Kal}}$ is not consistently the best predictor for each station, as for some locations, SVRs achieve better performance. These results suggest that the development of an ITS might take advantage of the combination of different predictors into a hybrid system.

Even if prediction accuracy is certainly a crucial feature of a forecasting model, computational time should be also taken into account when building a large-scale traffic monitoring system, considering both training time and prediction time. In Table VII, we summarize these computational costs for all the

TABLE IV
LEARNING CURVE FOR ALL THE COMPARED ALGORITHMS, USING ONE TO SIX MONTHS OF TRAINING SET AND ALWAYS EMPLOYING THE SAME THREE MONTHS FOR TEST. RESULTS ARE AVERAGED ON 16 STATIONS. MAPE_{100} IS REPORTED

RW	SM	SARIMA_{ML}	SARIMA_{Kal}	ARIMA_{Kal}	SVR_{RBF}	SVR^S_{RBF}	SVR^S_{lin}	ANN
$8.52_{(0.59)}$	$8.71_{(0.93)}$	$6.16_{(0.71)}$	$\mathbf{5.33}_{(0.68)}$	$8.26_{(0.68)}$	$6.33_{(0.75)}$	$6.17_{(0.75)}$	$6.34_{(0.69)}$	$6.45_{(0.59)}$
$8.52_{(0.58)}$	$8.77_{(0.86)}$	$5.75_{(0.59)}$	$\mathbf{5.02}_{(0.45)}$	$8.24_{(0.80)}$	$5.54_{(0.47)}$	$5.42_{(0.49)}$	$5.82_{(0.50)}$	$5.90_{(0.48)}$
$8.52_{(0.58)}$	$8.78_{(0.86)}$	$5.72_{(0.59)}$	$\mathbf{4.96}_{(0.46)}$	$8.28_{(0.73)}$	$5.41_{(0.47)}$	$5.30_{(0.47)}$	$5.66_{(0.49)}$	$5.80_{(0.45)}$
$8.52_{(0.58)}$	$8.81_{(0.86)}$	$5.68_{(0.60)}$	$\mathbf{4.99}_{(0.47)}$	$8.29_{(0.72)}$	$5.38_{(0.46)}$	$5.29_{(0.47)}$	$5.66_{(0.50)}$	$5.79_{(0.43)}$
$8.52_{(0.58)}$	$8.88_{(0.87)}$	$5.64_{(0.58)}$	$\mathbf{4.94}_{(0.47)}$	$8.10_{(0.74)}$	$5.32_{(0.48)}$	$5.23_{(0.47)}$	$5.56_{(0.47)}$	$5.78_{(0.39)}$
$8.52_{(0.58)}$	$8.92_{(0.87)}$	$5.62_{(0.59)}$	$\mathbf{4.97}_{(0.47)}$	$7.97_{(0.48)}$	$5.31_{(0.49)}$	$5.22_{(0.48)}$	$5.57_{(0.50)}$	$5.81_{(0.35)}$

TABLE V
DETAILED RESULTS FOR ALL 16 TEST STATIONS, USING THREE MONTHS FOR TRAINING SET AND THREE MONTHS FOR TEST. MAPE_{100} IS REPORTED

Station	RW	SM	SARIMA_{ML}	SARIMA_{Kal}	ARIMA_{Kal}	SVR_{RBF}	SVR^S_{RBF}	SVR^S_{lin}	ANN
715916	9.02	9.24	6.75	**5.71**	9.49	6.01	5.99	6.27	6.31
716312	8.22	8.55	5.66	**4.89**	8.15	5.21	5.19	5.80	5.99
716551	9.47	9.58	6.22	**5.35**	7.97	5.80	5.76	6.29	6.20
716841	9.04	8.31	5.58	**4.99**	9.35	5.46	5.36	5.74	5.85
716933	7.66	8.91	5.67	**5.02**	7.59	5.39	5.35	5.62	5.69
717087	8.33	7.83	5.38	**4.80**	8.27	5.21	5.20	5.44	5.51
717123	7.75	7.87	4.98	**4.50**	7.07	4.85	4.77	5.07	5.25
717152	8.51	8.66	5.80	**5.16**	8.18	5.62	5.54	5.87	6.00
717190	8.30	9.30	5.60	**5.08**	8.13	5.46	5.38	5.75	5.75
717269	8.97	11.02	6.67	**5.47**	9.09	6.01	5.81	6.24	6.24
718144	9.31	9.75	6.05	**5.37**	8.00	5.75	5.59	6.09	6.15
737344	8.81	9.16	6.20	**5.14**	8.88	5.86	5.75	5.86	6.35
763626	7.94	8.44	4.75	**4.21**	7.24	4.76	4.55	4.91	5.28
764151	8.34	8.04	5.69	**4.96**	7.93	5.28	5.11	5.46	5.79
767678	9.14	8.42	5.99	**4.94**	9.41	5.59	5.40	5.66	5.82
768682	7.58	7.46	4.56	**3.79**	7.73	4.22	4.12	4.45	4.59

algorithms, considering the setting with three months for training and three months for test. For prediction time, results are averaged on the number of forecasts (i.e., the number of time instants for which each model is asked a prediction), whereas for training time, the average is computed on the number of trained models. We report both prediction time, which has to be taken into account for real-time forecasts, and training time, which can become a bottleneck if the goal is to build a large-scale ITS. (For example, the PeMS data set contains over 6000 detector stations, and therefore, training a different model even for a subset of them can become hardly affordable.)

$\text{SARIMA}_{\text{Kal}}$ is the algorithm requiring the largest training time, more than twice that for $\text{SVR}^S_{\text{RBF}}$, owing to the expensive fitting of the Kalman state-space model and the computation of the covariance matrix. $\text{SARIMA}_{\text{ML}}$ is in fact much faster, employing a more naive optimization algorithm. Moreover, it is interesting to notice that $\text{SVR}^S_{\text{lin}}$, despite achieving a slightly lower prediction accuracy, is much faster than $\text{SARIMA}_{\text{Kal}}$ and SVR_{RBF} models, taking advantage of the stochastic gradient optimization algorithm, and might therefore represent an interesting tradeoff solution between computational expenses and forecasting precision.

V. CONCLUSION AND FUTURE WORK

We have presented an extensive experimental review of many statistical and machine learning approaches to short-term traffic flow forecasting. Following the approach in SARIMA, we have also proposed two new SVR models, employing a seasonal kernel to measure similarity between time-series examples. The presented results confirm that seasonality is a key feature in achieving high accuracy; however, the most accurate models often require high computational resources both during the training phase and at prediction time. For this reason, the presented seasonal kernel approach might be a reasonable compromise between forecasting accuracy and computational complexity issues. In particular, while SARIMA employed in combination with the Kalman filter ends up being the best model on average, the proposed approach is particularly competitive when considering predictions during highly congested periods. The SARIMA version that does not include a Kalman filter and the ANNs perform consistently worse than SVR with an RBF kernel, which, in turn, is less accurate than the seasonal-kernel variant.

As a future direction of research, we conjecture that a significant improvement in time-series forecasting may come from relational learning, where interrelations between different time series are taken into account. In a road network, for example, the traffic conditions in neighboring nodes are inherently interdependent, and exploiting this interdependence should be crucial in improving the prediction accuracy of related time series. Moreover, different from classical statistical approaches, relational learning algorithms can easily deal with multiple sources of information, which might become a crucial feature in the forthcoming years, where huge data sets might become available from wireless sensors, Global Positioning System, and floating-car data.

Another very interesting direction of research, which has been indicated by the experimental results presented in this paper, consists of investigating the covariate shift in traffic

TABLE VI
RESULTS DURING THE MORNING PEAK (7 A.M.–8 A.M.), DETAILED PER STATION. MAPE_{100} IS REPORTED

Station	RW	SM	SARIMA_{ML}	SARIMA_{Kal}	ARIMA_{Kal}	SVR_{RBF}	SVR^{S}_{RBF}	SVR^{S}_{lin}	ANN
715916	14.20	10.52	6.87	5.40	13.89	4.98	**4.89**	5.84	5.60
716312	17.62	12.34	8.27	**7.07**	14.21	7.96	7.82	9.39	9.39
716551	22.98	10.20	7.32	**6.22**	16.01	6.50	6.27	7.04	8.14
716841	19.65	9.12	6.98	**5.62**	18.47	6.49	6.78	6.27	7.14
716933	16.82	10.00	6.47	5.84	13.74	**4.87**	5.34	6.08	5.48
717087	17.78	10.29	5.78	**4.62**	14.20	4.82	5.05	6.30	5.61
717123	15.02	7.15	5.88	**5.11**	7.90	5.74	5.65	5.74	5.81
717152	16.60	7.30	6.39	**5.19**	12.06	5.20	5.55	5.34	5.82
717190	16.78	7.58	6.01	**5.09**	12.31	5.56	5.75	5.85	5.33
717269	20.33	18.58	8.63	7.18	15.07	6.81	**5.87**	8.16	7.60
718144	21.75	12.34	7.28	6.01	13.85	**5.53**	5.86	6.77	7.40
737344	13.48	6.66	6.14	**5.10**	9.82	5.62	5.96	5.50	6.66
763626	20.73	11.73	6.31	5.27	12.30	**4.81**	4.85	5.95	6.64
764151	17.51	7.04	5.87	4.78	12.35	4.83	**4.62**	4.84	5.10
767678	20.02	10.17	7.45	**6.05**	18.87	6.69	7.27	6.73	7.66
768682	18.33	7.03	5.04	4.22	16.14	**4.12**	4.24	4.44	4.21

TABLE VII
TRAINING AND PREDICTION TIME FOR ALL THE ALGORITHMS, AVERAGED ON THE 16 NODES CONSIDERED IN THE TEST SET

Model	Training time (s)	Prediction time (s)	Error (MAPE_{100})
ARIMA_{Kal}	6.8	0.4	8.3
SARIMA_{Kal}	608.9	8.9	5.0
SARIMA_{ML}	11.3	0.5	5.7
ANN	70.6	0.7	5.8
SVR^{S}_{lin}	18.7	0.1	5.7
SVR_{RBF}	219.0	1.6	5.4
SVR^{S}_{RBF}	276.2	10.7	5.3
SM	–	0.2	8.8
RW	–	0.1	8.5

time-series data. Our results, in fact, show that the accuracy of the SM predictor starts degrading when the temporal distance between training and test set grows too much, and for the other predictors, no further improvement is observed when using larger training sets including past months, which are too distant from prediction time. It would certainly be very interesting and challenging to better analyze this phenomenon, which might affect periodical retraining of the models.

REFERENCES

[1] L. A. Klein, M. K. Mills, and D. R. P. Gibson, *Traffic Detector Handbook [Electronic Resource]*, 3rd ed. Washington, DC, USA: U.S. Dept. of Transportation, 2006.

[2] J. Wohlers, A. Mertins, and N. Fliege, "Time-delay estimation for compound point-processes using hidden Markov models," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 1996, vol. 5, pp. 2817–2820.

[3] R. Li, G. Rose, and M. Sarvi, "Evaluation of speed-based travel time estimation models," *J. Transp. Eng.*, vol. 132, no. 7, pp. 540–547, Jul. 2006.

[4] G. Box, G. M. Jenkins, and G. Reinsel, *Time Series Analysis: Forecasting & Control*, 3rd ed. Englewood Cliffs, NJ, USA: Prentice-Hall, Feb. 1994.

[5] C. Moorthy and B. Ratcliffe, "Short term traffic forecasting using time series methods," *Transp. Plan. Technol.*, vol. 12, no. 1, pp. 45–56, Jul. 1988.

[6] S. Lee and D. B. Fambro, "Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting," *J. Transp. Res. Board*, vol. 1678, pp. 179–188, Jan. 1999.

[7] B. M. Williams and L. A. Hoel, "Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results," *J. Transp. Eng.*, vol. 129, no. 6, pp. 664–672, Nov. 2003.

[8] M. Van Der Voort, M. Dougherty, and S. Watson, "Combining Kohonen maps with arima time series models to forecast traffic flow," *Transp. Res. C, Emerg. Technol.*, vol. 4, no. 12, pp. 307–318, Oct. 1996.

[9] T. Kohonen, "The self-organizing map," *Neurocomputing*, vol. 21, no. 1–3, pp. 1–6, Nov. 1998.

[10] Y. Kamarianakis and P. Prastacos, "Forecasting traffic flow conditions in an urban network: Comparison of multivariate and univariate approaches," *Transp. Res. Rec.*, vol. 1857, no. 2, pp. 74–84, Jan. 2003.

[11] Y. Kamarianakis and P. Prastacos, "Space–time modeling of traffic flow," in *Proc. ERSA*, Aug. 2002, pp. 1–21.

[12] R. Engle, "GARCH 101: The Use of ARCH/GARCH Models in applied econometrics," *J. Econ. Perspect.*, vol. 15, no. 4, pp. 157–168, Fall 2001.

[13] Y. Kamarianakis, A. Kanas, and P. Prastacos, "Modeling traffic volatility dynamics in an urban network," *Transp. Res. Rec.*, vol. 1923, no. 1, pp. 18–27, Dec. 2005.

[14] M. Jun and M. Ying, "Research of traffic flow forecasting based on neural network," in *Proc. Workshop Intell. Inf. Technol. Appl.*, 2008, vol. 2, pp. 104–108.

[15] H. Su, L. Zhang, and S. Yu, "Short-term traffic flow prediction based on incremental support vector regression," in *Proc. Int. Conf. Natural Comput.*, 2007, vol. 1, pp. 640–645.

[16] G. Zhang, B. E. Patuwo, and M. Y. Hu, "Forecasting with artificial neural networks: The state of the art," *Int. J. Forecast.*, vol. 14, no. 1, pp. 35–62, Mar. 1998.

[17] M. S. Dougherty and M. R. Cobbett, "Short-term inter-urban traffic forecasts using neural networks," *Int. J. Forecast.*, vol. 13, no. 1, pp. 21–31, Mar. 1997.

[18] C.-H. Wu, J.-M. Ho, and D.-T. Lee, "Travel-time prediction with support vector regression," *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 4, pp. 276–281, Dec. 2004.

[19] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, and L. D. Han, "Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions," *Exp. Syst. Appl.*, vol. 36, pt. 2, no. 3, pp. 6164–6173, Apr. 2009.

[20] S. Sun, C. Zhang, and G. Yu, "A Bayesian network approach to traffic flow forecasting," *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 1, pp. 124–132, Mar. 2006.

[21] Y. An, Q. Song, and X. Zhao, "Short-term traffic flow forecasting via echo state neural networks," in *Proc. 7th ICNC*, Y. Ding, H. Wang, N. Xiong, K. Hao, and L. Wang, Eds., Shanghai, China, Jul. 26–28, 2011, pp. 844–847.

[22] G. A. Davis and N. L. Nihan, "Nonparametric regression and short-term freeway traffic forecasting," *J. Transp. Eng.*, vol. 117, no. 2, pp. 178–188, Mar./Apr. 1991.

[23] H. Sun, H. X. Liu, H. Xiao, and B. Ran, "Short Term Traffic Forecasting Using the Local Linear Regression Model," Univ. of California, Irvine, Irvine, CA, USA, UCI-ITS-WP-02-2, 2002.

[24] P. Lingras and P. Mountford, "Time delay neural networks designed using genetic algorithms for short term inter-city traffic forecasting," in *Engineering of Intelligent Systems*, L. Monostori, J. Váncza, and M. Ali, Eds. Berlin, Germany: Springer-Verlag, 2001, ser. Lecture Notes in Computer Science, pp. 290–299.

[25] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Optimized and meta-optimized neural networks for short-term traffic flow prediction: A genetic approach," *Transp. Res. C, Emerg. Technol.*, vol. 13, no. 3, pp. 211–234, Jun. 2005.

[26] J. Zhu and T. Zhang, "A layered neural network competitive algorithm for short-term traffic forecasting," in *Proc. CORD Conf.*, Dec. 2009, pp. 1–4.
[27] M. Lippi, M. Bertini, and P. Frasconi, "Collective traffic forecasting," in *Proc. ECML/PKDD*, 2010, pp. 259–273.
[28] J. G. Yu, C. Zhang, L. Zhuang, and J. Song, "Short-term traffic flow forecasting based on Markov chain model," in *Proc. IEEE Intell. Veh. Symp.*, 2003, pp. 208–212.
[29] I. Okutani and Y. J. Stephanedes, "Dynamic prediction of traffic volume through Kalman filtering theory," *Transp. Res. B, Methodol.*, vol. 18, no. 1, pp. 1–11, Feb. 1984.
[30] Y. Xie, Y. Zhang, and Z. Ye, "Short-term traffic volume forecasting using Kalman filter with discrete wavelet decomposition," *Comput.-Aided Civil Infrastruct. Eng.*, vol. 22, no. 5, pp. 326–334, Jul. 2007.
[31] K. Leong, *Book review: Periodicity and Stochastic Trends in Economic Time Series*. Oxford, U.K.: Oxford Univ. Press, 1998.
[32] Y. Li, E. P. Campbell, D. Haswell, R. J. Sneeuwjagt, and W. N. Venables, "Statistical forecasting of soil dryness index in the southwest of Western Australia," *Forest Ecol. Manage.*, vol. 183, no. 1–3, pp. 147–157, Sep. 2003.
[33] E. Cadenas and W. Rivera, "Wind speed forecasting in the south coast of Oaxaca, México," *Renew. Energy*, vol. 32, no. 12, pp. 2116–2128, Oct. 2007.
[34] F.-M. Tseng and G.-H. Tzeng, "A fuzzy seasonal arima model for forecasting," *Fuzzy Sets Syst.*, vol. 126, no. 3, pp. 367–376, Mar. 2002.
[35] L. Xuemei, D. Lixing, D. Yuyuan, and L. Lanlan, "Hybrid support vector machine and arima model in building cooling prediction," in *Proc. Int. Symp. Comput. Commun. Control Autom.*, May 2010, vol. 1, pp. 533–536.
[36] J. Whittaker, S. Garside, and K. Lindveld, "Tracking and predicting a network traffic process," *Int. J. Forecast.*, vol. 13, no. 1, pp. 51–61, Mar. 1997.
[37] B. Ghosh, B. Basu, and M. O'Mahony, "Bayesian time-series model for short-term traffic flow forecasting," *J. Transp. Eng.*, vol. 133, no. 3, pp. 180–189, Mar. 2007.
[38] E. Horvitz, J. Apacible, R. Sarin, and L. Liao, "Prediction, expectation, and surprise: Methods, designs, and study of a deployed traffic forecasting service," *CoRR*, vol. abs/1207.1352, Jul. 2012.
[39] B. L. Smith and M. J. Demetsky, "Traffic flow forecasting: Comparison of modeling approaches," *J. Transp. Eng.*, vol. 123, no. 4, pp. 261–266, Jul. 1997.
[40] S. Sun and Q. Chen, "Kernel regression with a Mahalanobis metric for short-term traffic flow forecasting," in *Intelligent Data Engineering and Automated Learning (IDEAL)*, C. Fyfe, D. Kim, S.-Y. Lee, and H. Yin, Eds. Berlin, Germany: Springer-Verlag, 2008, ser. Lecture Notes in Computer Science, pp. 9–16.
[41] F. Wang, G. Tan, and Y. Fang, "Multiscale wavelet support vector regression for traffic flow prediction," in *Proc. 3rd Int. Conf. IITA*, 2009, pp. 319–322.
[42] J. Fan and Q. Yao, *Nonlinear Time Series: Nonparametric and Parametric Methods*. New York, NY, USA: Springer-Verlag, 2005, ser. Series in Statistics.
[43] S. Clark, "Traffic prediction using multivariate nonparametric regression," *J. Transp. Eng.*, vol. 129, no. 2, pp. 161–168, 2003.
[44] T. Zhang, L. Hu, Z. Liu, and Y. Zhang, "Nonparametric regression for the short-term traffic flow forecasting," in *Proc. Int. Conf. MACE*, Jun. 2010, pp. 2850–2853.
[45] H. Sun, H. X. Liu, H. Xiao, and B. Ran, "Use of local linear regression model for short-term traffic forecasting," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1838, no. 1, pp. 143–150, Jan. 2003.
[46] D. Nikovski, N. Nishiuma, Y. Goto, and H. Kumazawa, "Univariate short-term prediction of road travel times," in *Proc. IEEE Intell. Transp. Syst. Conf.*, Vienna, Austria, 2005, pp. 1074–1079.
[47] B. L. Smith, B. M. Williams, and R. K. Oswald, "Comparison of parametric and nonparametric models for traffic flow forecasting," *Transp. Res. C, Emerg. Technol.*, vol. 10, no. 4, pp. 303–321, Aug. 2002.
[48] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Advances in Neural Information Processing Systems 9, NIPS*, M. Mozer, M. I. Jordan, and T. Petsche, Eds. Denver, CO, USA: MIT Press, Dec. 2–5, 1996, pp. 155–161.
[49] S. Shekhar and B. M. Williams, "Adaptive seasonal time series models for forecasting short-term traffic flow," *Transp. Res. Rec.*, no. 2024, pp. 116–125, Jan. 2007.
[50] C. Chatfield, *The Analysis of Time Series: An Introduction*, 6th ed. London, U.K.: Chapman & Hall, Jul. 2003.
[51] H. Akaike, "A new look at the statistical model identification," *IEEE Trans. Autom. Control*, vol. AC-19, no. 6, pp. 716–723, Dec. 1974.
[52] H. Akaike, "On entropy maximization principle," in *Applications of Statistics*. Amsterdam, The Netherlands: North Holland, 1977, pp. 27–41.
[53] G. Schwarz, "Estimating the dimension of a model," *Ann. Stat.*, vol. 6, no. 2, pp. 461–464, Mar. 1978.
[54] P. N. Pant and W. H. Starbuck, "Innocents in the forest: Forecasting and research methods," *J. Manage.*, vol. 16, no. 2, pp. 433–460, Jun. 1990.
[55] L. J. Tashman, "Out-of-sample tests of forecasting accuracy: An analysis and review," *Int. J. Forecast.*, vol. 16, no. 4, pp. 437–450, Oct.–Dec. 2000.
[56] S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Stat. Surv.*, vol. 4, pp. 40–79, 2010.
[57] L. Getoor and B. Taskar, *Introduction to Statistical Relational Learning (Adaptive Computation and Machine Learning)*. Cambridge, MA, USA: MIT Press, 2007.
[58] C. M. Queen and C. J. Albers, "Intervention and causality: Forecasting traffic flows using a dynamic Bayesian network," *J. Amer. Stat. Assoc.*, vol. 104, no. 486, pp. 669–681, Jun. 2009.
[59] V. N. Vapnik, *Statistical Learning Theory*. Hoboken, NJ, USA: Wiley, 1998.
[60] H. Shimodaira, T. Akae, M. Nakai, and S. Sagayama, "Jacobian adaptation of HMM with initial model selection for noisy speech recognition," in *Proc. ICSLP*, Oct. 2000, pp. 1003–1006.
[61] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," in *Neurocomputing: Foundations of Research*, J. A. Anderson and E. Rosenfeld, Eds. Cambridge, MA, USA: MIT Press, 1988, pp. 696–699.
[62] C. M. Bishop, *Neural Networks for Pattern Recognition*. New York, NY, USA: Oxford Univ. Press, Jan. 1996.
[63] L. Bottou and Y. LeCun, "Large scale online learning," in *Advances in Neural Information Processing Systems 16*, S. Thrun, L. Saul, and B. Schölkopf, Eds. Cambridge, MA, USA: MIT Press, 2004.
[64] P. Varaiya, "Freeway performance measurement system: Final report," Univ. of California Berkeley, Berkeley, CA, USA, Tech. Rep. UCB-ITS-PWP-2001-1, 2001.
[65] G. Gardner, A. C. Harvey, and G. D. A. Phillips, "Algorithm as 154: An algorithm for exact maximum likelihood estimation of autoregressive-moving average models by means of Kalman filtering," *J. Roy. Stat. Soc. C, Appl. Stat.*, vol. 29, no. 3, pp. 311–322, 1980.
[66] R. H. Jones, "Maximum likelihood fitting of ARMA models to time series with missing observations," *Technometrics*, vol. 22, no. 3, pp. 389–395, Aug. 1980.
[67] "R: A language and environment for statistical computing," *Vienna Austria R Foundation for Statistical Computing*, vol. 1, no. 10, p. 2673, Sep. 2009.
[68] G. Melard, "Algorithm as 197: A fast algorithm for the exact likelihood of autoregressive-moving average models," *J. Roy. Stat. Soc. C, Appl. Stat.*, vol. 33, no. 1, pp. 104–114, 1984.
[69] G. E. P. Box and G. M. Jenkins, "Time series analysis," *Forecasting and Control*, 1970.

Marco Lippi received the Bachelor's degree in computer engineering, the Doctoral degree in computer engineering, and the Ph.D. degree in computer and automation engineering from the University of Florence, Florence, Italy, in 2004, 2006, and 2010, respectively.

He is currently a Postdoctoral Fellow with the Department of Computer Engineering and Mathematical Sciences, University of Siena, Siena, Italy. His work focuses on machine learning and artificial intelligence, with specific interests in statistical relational learning and applications to the fields of bioinformatics, time-series forecasting, and computer vision.

Dr. Lippi received the "E. Caianiello" Award for the best Italian Ph.D. thesis in the field of neural networks in 2012.

Matteo Bertini received the M. Sc. degree in informatics engineering and the Ph.D. degree from the University of Florence, Florence, Italy, in 2006 and 2010, respectively.

He is currently a Software Developer with Develer S.r.l., Florence.

Paolo Frasconi received the M. Sc. degree in electronic engineering and the Ph.D. degree in computer science from the University of Florence, Florence, Italy, in 1990 and 1994, respectively.

He is a Professor of computer science with the Department of Information Engineering, University of Florence, Florence, Italy. His research interests include machine learning, with particular emphasis on algorithms for structured and relational data, learning with logical representations, and applications to bioinformatics.

Mr. Frasconi is an Associate Editor of the Artificial Intelligence Journal and an Action Editor of the Machine Learning Journal. He co-chaired the International Conference on Prestigious Applications of Artificial Intelligence (PAIS 2012), the Association for the Advancement of Artificial Intelligence Special Track on Artificial Intelligence and Bioinformatics (AAAI 2010), the 20th International Conference on Inductive Logic Programming (ILP 2010), and the Fifth International Workshop on Mining and Learning with Graphs (MLG 2007).