This is the peer reviewd version of the followng article:

Collective traffic forecasting / Lippi, Marco; Bertini, Matteo; Frasconi, Paolo. - 6322:2(2010), pp. 259-273. (Intervento presentato al convegno European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, ECML PKDD 2010 tenutosi a Barcelona, esp nel 2010) [10.1007/978-3-642-15883-4_17].

SPRINGER-VERLAG BERLIN
Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

05/08/2024 22:32

(Article begins on next page)

Collective traffic forecasting

Marco Lippi, Matteo Bertini, Paolo Frasconi

Dipartimento Sistemi e Informatica, Università degli Studi di Firenze
lippi@dsi.unifi.it,bertinim@dsi.unifi.it,p-f@dsi.unifi.it

Abstract. Traffic forecasting has recently become a crucial task in the area of intelligent transportation systems, and in particular in the development of traffic management and control. We focus on the simultaneous prediction of the congestion state at multiple lead times and at multiple nodes of a transport network, given historical and recent information. This is a highly relational task along the spatial and the temporal dimensions and we advocate the application of statistical relational learning techniques. We formulate the task in the supervised learning from interpretations setting and use Markov logic networks with grounding-specific weights to perform collective classification. Experimental results on data obtained from the California Freeway Performance Measurement System (PeMS) show the advantages of the proposed solution, with respect to propositional classifiers. In particular, we obtained significant performance improvement at larger time leads.

1 Introduction

Intelligent Transportation Systems (ITSs) are widespread in many densely urbanized areas, as they give the opportunity to better analyze and manage the growing amount of traffic flows, due to increased motorization, urbanization, population growth, and changes in population density.

One of the main targets of an ITS is to reduce congestion times, as they seriously affect the efficiency of a transportation infrastructure, usually measured as a multi-objective function taking into account several aspects of a traffic control system, like travel time, air pollution, and fuel consumption. As for travel time, for example, it is often important to minimize both the mean value and its variability [13], which represents an added cost for a traveler making a given journey.

This management effort is supported by the growing amount of data gathered by ITSs, coming from a variety of different sources. Loop detectors are the most commonly used vehicle detectors for freeway traffic monitoring, which can typically register the number of vehicles passed in a certain time interval (flow), and the percentage of time the sensor is occupied per interval (occupancy). In recent years, there has been also a spread of employment of wireless sensors, like GPS and floating car data (FCD) [11], which will eventually reveal in real-time the position of almost every vehicle, by collecting information from mobile phones in vehicles that are being driven. These different kinds of data are heterogeneous,

and therefore would need a pre-processing phase in order to be integrated and used as support to the decision processes. A large sensor network corresponds to a large number of potentially noisy or faulty components. In particular, in the case of traffic detectors, several different fault typologies might affect the system: communication problems on the line, intermittent faults resulting in insufficient or incomplete data transmitted by the sensors, broken controllers, bad wiring, etc.

In Urban Traffic Control (UTC) systems, such as the Split Cycle Offset Optimization Technique (SCOOT) system [15] and the Sydney Coordinated Adaptive Traffic (SCAT) system [16], short-term forecasting modules are used to adapt system variables and maintain optimal performances. Systems without a forecasting module can only operate in a reactive manner, after some event has occurred. Classic short-term forecasting approaches usually focus on 10-15 minutes ahead predictions [19, 24, 20]. Effective proactive transportation management (e.g. car navigation systems), arguably needs forecasts extending on longer horizons in order to be effective.

Most of the predictors employed in these traffic control systems are based on a time series forecasting technology. Time series forecasting is a vast area of statistics, with a wide range of application domains [3]. Given the history of past events sampled at certain time intervals, the goal is to predict the continuation of the series. Formally, given a time series $\mathcal{X} = \{x_1, \ldots, x_t\}$ describing the dynamic behavior of some observed physical quantity x_j, the task is to predict x_{t+1}. In the traffic management domain, common physical quantities of interest are (i) the traffic *flow* of cars passing at a given location in a fixed time interval, (ii) the *average speed* observed at a certain location, (iii) the *average time* needed to travel between two locations.

Historically, many statistical methods have been developed to address the problem of traffic forecasting: these include methods based on auto-regression and moving average, such as ARMA, ARIMA, SARIMA and other variants, or non-parametric regression. See [19] and references therein for an overview of these statistical methodologies. Also from a machine learning perspective, the problem of traffic forecasting has been addressed using a wide number of different algorithms, like support vector regression (SVR) [24], Bayesian networks [20] or time-delay neural networks (TDNNs) [1]. Most of these methods address the problem as single-point forecasting, intended as the ability to predict future values of a certain physical quantity at a certain location, given only past measurements of the same quantity at the same location. Yet, given a graph representing a transportation network, predicting the traffic conditions at multiple nodes and at multiple temporal steps ahead is an inherently relational task, both in the spatial and in the temporal dimension: for example, at time t, the predictions for two measurement sites s_1 and s_2, which are spatially close in the network, can be strongly interrelated, as well as predictions at t and $t+1$ for the same site s. Inter-dependencies between different time series are usually referred to as Granger's causality [8], a concept initially introduced in the domain of economy and marketing: time series A is said to Granger-cause time

series B if A can be used to enhance the forecasts on B. Few methods until now have taken into account the relational structure of the data: multiple Kalman filters [23], the STARIMA model (space-time ARIMA) [10] and structural time series models [7] are the first attempts in this direction.

The use of a statistical relational learning (SRL) framework for this kind of task might be crucial in order to improve predictive accuracy. First of all, SRL allows to represent the domain in terms of logical predicates and rules, and therefore to easily include background knowledge in the model, and to describe relations and dependencies, such as the topological characteristics of a transportation network. Within this setting, the capability of SRL models to integrate multiple sources and levels of information might become a key feature for future transportation control systems. Moreover, the SRL framework allows to perform collective classification or regression, by jointly predicting traffic conditions in the whole network in a single inference process: in this way, a single model can represent a wide set of locations, while propositional methods should typically train a different predictor for each node in the graph.

Dealing with large data sets within SRL is a problem which still has to receive adequate attention, but it is one of the key challenges of the whole research area [5]. Traffic forecasting is a very interesting benchmark from this point of view: for example, just considering highways in California, over 30,000 detectors continuously generate flow and occupancy data, producing a huge amount of information. Testing the scalability of inference algorithms on such a large model is a crucial point for SRL methodologies.

Moreover, many of the classic time series approaches like ARIMA, SARIMA and most of their variants, are basically linear models. Non-linearity, on the other hand, is a crucial issue in many application domains in order to build a competitive predictor: for this reason, some attempts to extend statistical approaches towards non-linear models have been proposed, as in the KARIMA or VARMA models [22, 4].

Among the many SRL methodologies that have been proposed in recent years, we employ Markov logic [6], extended with grounding-specific weights (GS-MLNs) [12]. The first-order logic formalism allows to incorporate background knowledge of the domain in a straightforward way. The use of probabilities within such a model allow us to handle noise to take into account statistical interdependencies. The grounding-specific weights extension enables the use of vectors of continuous features and non-linear classifiers (like neural networks) within the model.

2 Grounding-Specific Markov Logic Networks

Markov logic [6] integrates first-order logic with probabilistic graphical models, providing a formalism which allows us to describe a domain in terms of logic predicates and probabilistic formulae. While a first-order knowledge base can be seen as a set of *hard* constraints over possible worlds (or Herbrand interpretations), where a world violating even a single formula has zero probability, in

Markov logic such a world would be *less probable*, but not impossible. Formally, a Markov logic network (MLN) is defined by a set of first-order logic formulae $\mathcal{F} = \{F_1, \ldots, F_n\}$ and a set of constants $\mathcal{C} = \{C_1, \ldots, C_k\}$. A Markov random field is then created by introducing a binary node for each possible ground atom and an edge between two nodes if the corresponding atoms appear together in a ground formula. Uncertainty is handled by attaching a real-valued weight w_j to each formula F_j: the higher the weight, the lower the probability of a world violating that formula, others things being equal. In the discriminative setting, MLNs essentially define a template for arbitrary (non linear-chain) conditional random fields that would be hard to specify and maintain if hand-coded. The language of first-order logic, in fact, allows to describe relations and inter-dependencies between the different domain objects in a straightforward way. In this paper, we are interested in the supervised learning setting. In Markov logic, the usual distinction between the input and output portions of the data is reflected in the distinction between evidence and query atoms. In this setting, an MLN defines a conditional probability distribution of query atoms Y given evidence atoms X, expressed as a log-linear model in a feature space described by all possible groundings of each formula:

$$P(Y = y|X = x) = \frac{\exp\left(\sum_{F_i \in F_Y} w_i n_i(x, y)\right)}{Z_x} \tag{1}$$

where F_Y is the set of clauses involving query atoms and $n_i(x, y)$ is the number of groundings of formula F_i satisfied in world (x, y). Note that the feature space jointly involves X and Y as in other approached to structured output learning. MAP inference in this setting allows us to collectively predict the truth value of all query ground atoms: $f(x) = y^* = \arg\max_y P(Y = y|X = x)$. Solving the MAP inference problem is known to be intractable but even if we could solve it exactly, the prediction function f is still linear in the feature space induced by the logic formulae. Hence, a crucial ingredient for obtaining an expressive model (which often means an accurate model) is the ability of tailoring the feature space to the problem at hand. For some problems, this space needs to be high-dimensional. For example, it is well known that linear chain conditional random fields (which we can see as a special case of discriminative MLNs), often work better in practice when using high-dimensional feature spaces. However, the logic language behind MLNs only offers a limited ability for controlling the size of the feature space. We will explain this using the following example. Suppose we have a certain query predicate of interest, $\texttt{Query}(\texttt{t}, \texttt{s})$ (where, e.g., the variable $\texttt{t}$ and $\texttt{s}$ represent time and space) that we know to be predictable from a certain set of attributes, one for each $(\texttt{t}, \texttt{s})$ pair, represented by the evidence predicate $\texttt{Attributes}(\texttt{t}, \texttt{s}, \texttt{a}_1, \texttt{a}_2, \ldots, \texttt{a}_\texttt{n})$. Also, suppose that performance for this hypothetical problem crucially depends, for each t and s, on our ability of defining a nonlinear mapping between the attributes and the query. To fix our ideas, imagine that an SVM with RBF kernel taking $\texttt{a}_1, \texttt{a}_2, \ldots, \texttt{a}_\texttt{n}$ as inputs (treating each (s, t) pair as an independent example) already produces a good classifier, while a linear classifier fails. Finally, suppose we have some available

background knowledge, which might help us to write formulae introducing statistical interdependencies between different query ground atoms (at different $\mathtt{t}$ and $\mathtt{s}$), thus giving us a potential advantage in using a non-iid classifier for this problem. An MLN would be a good candidate for solving such a problem, but emulating the already good feature space induced by the RBF kernel may be tricky. One possibility for producing a very high dimensional feature space is to define a feature for each possible configuration of the attributes. This can be achieved by writing several ground formulae with different associated weights. For this purpose, in the Alchemy system[1], one might write an expression like

$$\mathtt{Attributes}(\mathtt{t}, \mathtt{s}, +\mathtt{a}_1, +\mathtt{a}_2, \ldots, +\mathtt{a}_\mathtt{n}) \Rightarrow \mathtt{Query}(\mathtt{t}, \mathtt{s})$$

where the $\mathtt{+}$ symbol preceding some of the variables expands the expression into separate formulae resulting from the possible combination of constants from those variables. Different weights are attached to each formula in the resulting expansion. Yet, this solution presents two main limitations: first, the number of parameters of the MLN grows exponentially with the number of variables in the formula; second, if some of the attributes ai are continuous, they need to be discretized in order to be used within the model.

GS-MLNs [12] allow us to use weights that depend on the specific grounding of a formula, even if the number of possible groundings can in principle grow exponentially or can be unbound in the case of real-valued constants. Under this model, we can write formulae of the kind:

$$\mathtt{Attributes}(\mathtt{t}, \mathtt{s}, \$\mathtt{v}) \Rightarrow \mathtt{Query}(\mathtt{t}, \mathtt{s})$$

where $\mathtt{v}$ has the type of an n-dimensional real vector, and the $\$$ symbol indicates that the weight of the formula is a parameterized function of the specific constant substituted for the variable $\mathtt{v}$. In our approach, the function is realized by a discriminative classifier, such as a neural network with adjustable parameters θ. The idea of integrating non-linear classifiers like neural networks within conditional random fields has been also recently proposed in conditional neural fields [14].

In MLN with grounding-specific weights, the conditional probability of query atoms given evidence can therefore be rewritten as follows:

$$P(Y = y|X = x) = \frac{\exp\left(\sum_{F_i \in F_Y} \sum_j w_i(c_{ij}, \theta_i) n_{ij}(x, y)\right)}{Z_x} \tag{2}$$

where $w_i(c_{ij}, \theta_i)$ is a function of some constants depending on the specific grounding, indicated by c_{ij}, and of a set of parameters θ_i.

Any inference algorithm for standard MLNs can be applied with no changes. During the parameter learning phase, on the other hand, MLN and neural network weights need to be adjusted jointly. The resulting algorithm can implement

[1] http://alchemy.cs.washington.edu

gradient ascent, exploiting the chain rule:

$$\frac{\partial P(y|x)}{\partial \theta_k} = \frac{\partial P(y|x)}{\partial w_i} \frac{\partial w_i}{\partial \theta_k}$$

where the first term is computed by MLN inference and the second term is computed by backpropagation. As in standard MLNs, the computation of the first term requires to compute the expected counts $E_w[n_i(x, y)]$:

$$\frac{\partial P(y|x)}{\partial w_i} = n_i(x, y) - \sum_{y'} P(y'|x) n_i(x, y^*) = n_i(x, y) - E_w[n_i(x, y)]$$

which are usually approximated with the counts in the MAP state y^*:

$$\frac{\partial P(y|x)}{\partial w_i} \simeq n_i(x, y) - n_i(x, y^*)$$

From the above equation, we see that if all the groundings of formula F_j are correctly assigned their truth values in the MAP state y^*, then that formula gives a zero contribution to the gradient, because $n_j(x, y) = n_j(x, y^*)$. For grounding-specific formulae, each grounding corresponds to a different example for the neural network: therefore, there will be no backpropagation term for a given example if the truth value of the corresponding atom has been correctly assigned by the collective inference.

When learning from many independent interpretations, it is possible to split the data set into minibatches and apply stochastic gradient descent [2]. Basically this means that gradients of the likelihood are only computed for small batches of interpretations and weights (both for the MLN and for the neural networks) are updated immediately, before working with the subsequent interpretations. Stochastic gradient descent can be more generally applied to minibatches consisting of the connected components of the Markov random field generated by the MLN. This trick is inspired by a common practice when training neural networks and can very significantly speedup training time.

3 Data preparation and experimental setting

3.1 The data set

We performed our experiments on the California Freeway Performance Measurement System (PeMS) data set [21], which is a wide collection of measurements obtained by over 30,000 sensors and detectors placed around nine districts in California. The system covers 164 Freeways, including a total number of 6,328 mainline Vehicle Detector Stations and 3,470 Ramp Detectors.

The loop detectors used within the PeMS are frequently deployed as single detectors, one loop per lane per detector station. The raw single loop signal is noisy and can be used directly to obtain only the raw count (traffic flow) and the occupancy (lapse of time the loop detector is active) but cannot measure the

Fig. 1. The case study used in the experiments: 7 measurement stations placed on three different Highways in the area of East Los Angeles.

speed of the vehicles. The PeMS infrastructure collects filtered and aggregated flow and occupancy from single loop detectors, and provides an estimate of the speed [9] and other derived quantities. In some locations, a double loop detector is used to directly measure the instantaneous speed of the vehicles. All traffic detectors report measurements every 30 seconds.

In our experiments, the goal is to predict whether the average speed at a certain time in the future falls under a certain threshold. This is the measurement employed by Google^{TM} Maps[2] for the coloring scheme encoding the different levels of traffic congestions: the yellow code, for example, means that the average speed is below 50 mph, which is the threshold adopted in all our experiments.

Table 1. Summary of stations used in experiments. VDS stays for Vehicle Detector Station and identifies each station in the PeMS data set.

Station	VDS	Highway	# Lanes
A	716091	I10-W	4
B	717055	I10-W	4
C	717119	I10-W	4
D	717154	I10-W	5
E	717169	I10-W	4
F	717951	I605-S	4
G	718018	I710-S	3

In our case study, we focused on seven locations in the area of East Los Angeles (see Figure 1), five of which are placed on the I10 Highway (direction West), one on the I5 (direction South) and one on the I710 (direction South) (see Table 1). We aggregated the available raw data into 15-minutes samples, averaging the measurements taken on the different lanes. In all our experiments we used the previous three hours of measurements as the input portion of the data. For all considered locations we predict traffic congestions at the next four lead times (i.e., 15, 30, 45 and 60 minutes ahead). Thus each interpretation spans

[2] http://maps.google.com

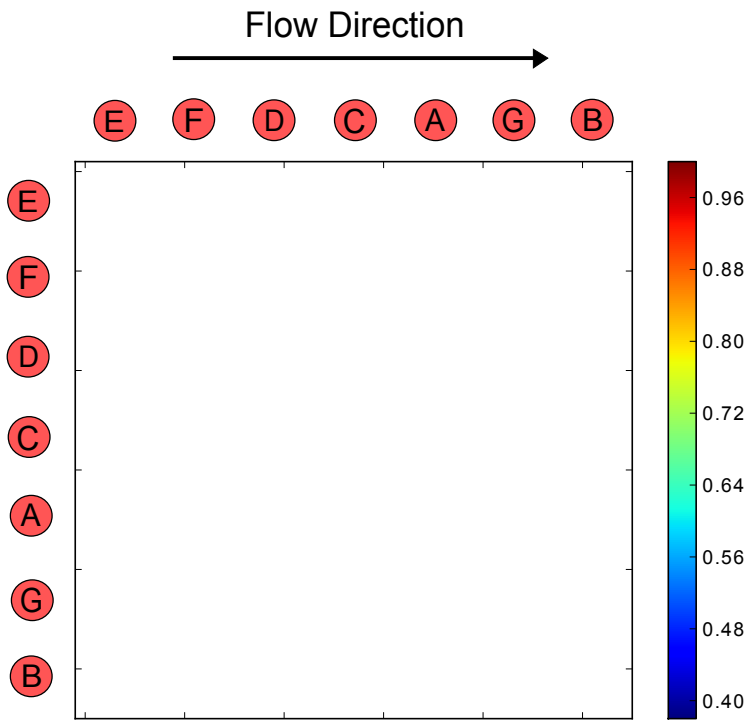

Fig. 2. Spatiotemporal correlations in the training set data. There are 28 boolean congestion variables corresponding to 7 measurement stations and 4 lead times. Rows and columns are lexicographically sorted on the station-lead time pair. With the exception of station E, spatial correlations among nearby stations are very strong and we can observe the spatiotemporal propagation of the congestion state along the direction of flow (traffic is westbound).

a time interval of four hours. We used two months of data (Jan-Feb 2008) as training set, one month (Mar 2008) as tuning set, and two months (Apr-May 2008) for test. Time intervals of four hours containing missing measurements due to temporary faults in the sensors were discarded from the data set. The tuning set was used to choose the C and γ parameters for the SVM predictor, and to perform early stopping for the GS-MLNs.

The inter-dependencies between nodes which are close in the transportation network are evident from the simple correlation diagram shown in Figure 2.

3.2 Experimental setup

The GS-MLN model was trained under the learning from interpretations setting. An interpretation in this case corresponds to a typical forecasting session, where at time t we want to forecast the congestion state of the network at future lead times, given previous measurements. Hence interpretations are indexed by

their time stamp t, which is therefore be omitted in all formulae (the temporal index $\mathtt{h}$ in the formulae below refers to the time lead of the prediction, i.e. 1,2,3, and 4 for 15,30,45, and 60 minutes ahead). Interpretations are assumed to be independent, and this essentially follows the setting of other supervised learning approaches such as [24, 18, 17]. However, in our approach congestion states at different lead times and at different sites are predicted collectively. Dependencies are introduced by spatiotemporal neighborhood rules, such as

$$\begin{aligned}&\mathtt{Congestion}(+\mathtt{s},\mathtt{h}) \wedge \mathtt{Weekday}(+\mathtt{wd}) \wedge \mathtt{TimeSlot}(+\mathtt{ts})\\ &\Rightarrow \mathtt{Congestion}(+\mathtt{s},\mathtt{h}+1)\end{aligned} \tag{3}$$

$$\mathtt{Congestion}(+\mathtt{s1},\mathtt{h}) \wedge \mathtt{Next}(\mathtt{s1},\mathtt{s2}) \Rightarrow \mathtt{Congestion}(+\mathtt{s2},\mathtt{h}+1) \tag{4}$$

where $\mathtt{Congestion}(\mathtt{S},\mathtt{H})$ is true of the velocity at site $\mathtt{S}$ and lead time $\mathtt{H}$ falls below the 50mph threshold, and the + symbol before a site variable assigns a different weight to each site or site pair. The predicate $\mathtt{Next}(\mathtt{s1},\mathtt{s2})$ is true if site $\mathtt{s2}$ follows site $\mathtt{s1}$ in the flow direction. The predicate $\mathtt{Weekday}(\mathtt{wd})$ distinguishes between workdays and holidays, while $\mathtt{TimeSlot}(\mathtt{ts})$ encodes the part of the day (morning, afternoon, etc.) of the current timestamp.

Of course the road congestion state also depends on previously observed velocity or flow. Indeed, literature results [24, 18, 17] suggest that good local forecasts can be obtained as a nonlinear function of the recent sequence of observed traffic flow or speed. Using GS-MLNs, continuous attributes describing the observed time series can be introduced within the model, using a set of grounding-specific formulae, e.g.:

$$\mathtt{SpeedSeries}(\mathtt{SD},\$\mathtt{SeriesD}) \Rightarrow \mathtt{Congestion}(\mathtt{SD},1) \tag{5}$$

where the grounding-specific weights are computed by a neural network taking as input a real vector associated with constant $\mathtt{Series_SD}$ (being $\mathtt{SD}$ the station identifier), containing past speed measurements during the previous 12 time steps. Note that a separate formula (and a separate neural network) is employed for each site and for each lead time.

Seasonality was encoded by the predicate $\mathtt{SeasonalCongestion}(\mathtt{s})$, which is true if, on average, station $\mathtt{s}$ presents a congestion at the time of the day referred to by the current interpretation (this information was extracted from averages on the training set). Other pieces of background knowledge were encoded in the MLN. For example, the number of lanes at a given site can be influence bottleneck behaviors:

$$\begin{aligned}&\mathtt{Congestion}(\mathtt{s1},\mathtt{h}) \wedge \mathtt{NodeClose}(\mathtt{s1},\mathtt{s2}) \wedge \mathtt{NLanes}(\mathtt{s1},\mathtt{l1}) \wedge \mathtt{NLanes}(\mathtt{s2},\mathtt{l2}) \wedge\\ &\mathtt{l2} < \mathtt{l1} \Rightarrow \mathtt{Congestion}(\mathtt{s2},\mathtt{h}+1)\end{aligned}$$

The MLN contained 14 formulae in the background knowledge and 125 parameters after grounding variables prefixed by a +. The 28 neural networks had 12 continuous inputs and 5 hidden units each, yielding about 2000 parameters in total. Our software implementation is a modified version of the Alchemy system to

incorporate neural network as pluggable components. Inference was performed by MaxWalkSat algorithm. Twenty epochs of stochastic gradient ascent were performed, with a learning rate $\epsilon = 0.03$ for the MLN weights, and $\mu = \frac{0.00003}{n}$ for the neural networks, being n the number of misclassifications in the current minibatch. In order to further speed up the training procedure, all neural networks were pre-trained for a few epochs (using the congestion state as the target) before plugging them into the GS-MLN jointly and tuning the whole set of parameters.

We compared the obtained results against three competitors:

Trivial predictor The seasonal average classifier predicts, for any time of the day, the congestion state observed on average in the training set at that time. Although it is a baseline predictor, it is widely used in literature as a competitor.

SVM We used SVM as a representative of state-of-the-art propositional classifiers. A different SVM with RBF kernel was trained for each station and for each lead time, performing a separated model selection for the C and γ values to be adopted for each measurement station. The measurements used by the SVM predictor consist in the speed time series observed in the past 180 minutes, aggregated at 15 minutes intervals, hence producing 12 inputs, plus an additional one representing the seasonal average at current time. A gaussian standardization was applied to all these inputs.

Standard MLN When implementing the classifier based on standard MLNs, the speed time series had to be discretized in order to be used within the model. Five different speed classes were used, and the quantization thresholds were chosen by following a maximum entropy strategy. The trend of the speed time series was modeled by the following set of formulae that were used in place of formula 5:

$$\begin{aligned}&\texttt{Speed_Past_1}(\texttt{n}, +\texttt{v}) \Rightarrow \texttt{Congestion}(\texttt{n}, 1)\\&\dots\\&\texttt{Speed_Past_k}(\texttt{n}, +\texttt{v}) \Rightarrow \texttt{Congestion}(\texttt{n}, 1)\end{aligned}$$

where predicate $\texttt{Speed_Past_j}(\texttt{node}, \texttt{speed_value})$ encodes the discrete values of the speed at the j-th time step before the current time. Note that an MLN containing only the above formulae essentially represents a logistic regression classifier taking the discretized features as inputs. All remaining formulae were identical to those used in conjunction with the GS-MLN.

As for the predictor based on GS-MLNs, there is no need to use discretized features, but the same vectors of features used by the SVM classifier can be adopted.

4 Results and discussion

4.1 Performance analysis

The congestion state in the analyzed highway segment is a very unbalanced task even at the 50mph threshold. Table 2 shows the percentage of positive query

atoms in the training set and in the test set, for each station. The last two columns report the percentage of days containing at least one congestion. The data distribution shows that the stations present different behaviors, corroborating the choice of using different neural networks for each station.

Table 2. Percentage of true ground atoms, for each measurement station. The percentage of days in the train/test set containing at least one congestion is reported in the last two columns.

Station	% pos train	% pos test	% pos days train	% pos days test
A	11.8	9.2	78.3	70.7
B	5.8	4.9	60.0	53.4
C	16.8	13.7	66.6	86.9
D	3.4	2.3	45.0	31.0
E	28.2	22.9	86.7	72.4
F	3.9	1.8	51.6	31.0
G	1.9	1.7	30.0	22.4

Given the unbalanced data set, we compare the predictors on the F_1 measure, as the harmonic mean between precision $P = \frac{TP}{TP+FP}$ and recall $R = \frac{TP}{TP+FN}$: $F_1 = \frac{2PR}{P+R}$. Table 3 shows the F_1 measure, averaged per station. The advantages of the relational approach are much more evident when increasing the prediction horizon: at 45 and 60 minutes ahead, the improvement of the GS-MLN model is statistically significant, according to a Wilcoxon paired test, with p-value< 0.05. Detailed comparisons for each sensor station at 15, 30, 45, and 60 minutes ahead are reported in Tables 4 , Tables 5 , Tables 6 and 7, respectively. These tables show that congestion at some of the sites are clearly "easier" to predict than at other sites. Comparing Tables 4-7 to Table 2 we see that the difficulty strongly correlates with the data set imbalance, an effect which is hardly surprising. It is also often the case that GS-MLN significantly outperforms the SVM classifier for "difficult" sites. The comparison between the standard MLN and the GS-MLN shows that input quantization can significantly deteriorate performance, all other things being equal. This supports the proposed strategy of embedding neural networks as a key component of the model.

An interesting performance measure considers only those test cases in which traffic conditions are anomalous with respect to the typical seasonal behavior. To this aim, we restricted the test set, by collecting only those interpretations for which the baseline seasonal average classifier would miss the prediction of the current congestion state. Table 8 shows that the advantage of the relational approach is still evident for long prediction horizons.

The experiments were performed on a 3GHz processor with 4Mb cache. The total training time for SVM is 40 minutes, and 7-8 hours for GS-MLNs. As for testing times, both systems perform in real-time.

Table 3. Comparison between the tested predictors. Results show the F_1 on the positive class, averaged on the seven nodes. The symbol $\ominus$ indicates a significant loss of the method with respect to GS-MLN, according to a Wilcoxon paired test (p-value<0.05).

	15 m		30 m		45 m		60 m	
Seasonal Avg	38.3	$\ominus$	38.3	$\ominus$	38.3	$\ominus$	38.3	$\ominus$
SVM	81.7		68.6		56.4	$\ominus$	51.8	$\ominus$
MLN	59.5	$\ominus$	56.5	$\ominus$	53.6	$\ominus$	50.4	$\ominus$
GS-MLN	80.9		69.2		61.6		56.9	

Table 4. Details on the predictions per station, at 15 minutes ahead.

	SVM	MLN	GS-MLN
A	82.9	64.0	80.4
B	78.0	50.8	74.5
C	91.2	66.5	89.1
D	77.5	51.9	79.5
E	92.0	69.4	92.9
F	70.6	51.9	66.7
G	80.0	61.7	83.4

Table 5. Details on the predictions per station, at 30 minutes ahead.

	SVM	MLN	GS-MLN
A	76.2	50.6	74.2
B	60.9	46.5	60.5
C	85.6	81.5	86.0
D	64.4	57.0	65.5
E	85.7	74.3	86.0
F	36.0	30.4	45.6
G	71.6	55.5	66.7

Table 6. Details on the predictions per station, at 45 minutes ahead.

	SVM	MLN	GS-MLN
A	74.3	71.1	73.5
B	41.6	29.3	44.5
C	82.7	75.1	83.9
D	46.2	49.9	59.4
E	80.7	78.2	82.9
F	33.8	28.7	37.3
G	35.5	43.2	50.0

Table 7. Details on the predictions per station, at 60 minutes ahead.

	SVM	MLN	GS-MLN
A	72.5	71.6	72.1
B	29.9	27.9	37.0
C	83.5	80.9	84.7
D	38.0	41.0	52.4
E	79.7	75.4	79.9
F	26.0	21.0	29.9
G	33.3	32.6	42.4

Table 8. Comparison between the tested predictors, only on those cases where the seasonal average predictor fails. Results show the F_1 on the positive class, averaged on the seven nodes.

	15 m	30 m	45 m	60 m
SVM	81.4	69.1	59.1	59.2
MLN	39.9	47.6	48.4	41.6
GS-MLN	78.4	68.2	68.4	65.5

4.2 Dealing with missing data

The problem of missing or incomplete data is crucial in all time series forecasting applications [3, 4]: in the case of punctual missing information, a reconstruction algorithm might be employed in order to interpolate the signal, so that prediction methods might be applied unchanged. Occasionally, sensor faults can last several time steps, and when this happens, a large part of the input can be unavailable to a standard propositional predictor until the sensor recovers from the failure state. Of course, cases containing missing data can be filtered from the training set as we did for our previous experiments. However, in order to deploy a predictor on a real-time task, it is necessary also to handle the case of missing values at prediction time. A relational model can be in principle more robust than its propositional counterpart by exploiting information from nearby sites.

In this section we report results obtained by simulating the absence of several values within the observed time series, using the trivial seasonal average predictor (Section 3.2) as reconstruction algorithm for these unobserved data . Producing an accurate model of sensor faults is clearly beyond the scope of this paper and we built a naive observation model based on a two states first-order Markov chain with $P(\text{observed} \mapsto \text{observed}) = 0.99$ and $P(\text{reconstructed} \mapsto \text{reconstructed}) = 0.9$. The performance of the predictors on this task are shown in Table 9.

5 Conclusions

We have proposed a statistical relational learning approach to traffic forecasting, in order to collectively classify the congestion state at several nodes of a

Table 9. Comparison between the tested predictors, using a test set containing missing values, reconstructed using the seasonal average. Results show the F_1 on the positive class.

	15 m	30 m	45 m	60 m
SVM	79.0	63.2	53.6	48.8
GS-MLN	80.5	70.4	62.6	58.1

transportation network, and at multiple lead times in the future, exploiting the relational structure of the domain. Our method is based on grounding-specific Markov logic networks, which extend the framework of Markov logic in order to include discriminative classifiers and generic vectors of features within the model. Experimental results performed on a case study extracted from the Californian PeMS data set show that the relational approach outperforms the propositional one, in particular when the prediction horizon grows.

Although we performed experiments on a binary classification task, we plan to extend the framework also to the case of multiclass classification or ordinal regression. As a further direction of research, the use of Markov logic gives the possibility to extend the model by applying structure learning algorithms to learn relations and dependencies directly from data in an automatic way.

The proposed methodology is not restricted to traffic management, but it can be applied to several different time series application domains, such as ecologic time series, for air pollution monitoring, or economic time series, for marketing analysis.

Acknowledgments

This research is partially supported by grant SSAMM-2009 from the Foundation for Research and Innovation of the University of Florence.

References

1. B. Abdulhai, H. Porwal, and W. Recker. Short-term freeway traffic flow prediction using genetically optimized time-delay-based neural networks. *Transportation Research Board, 78th Annual Meeting, Washington D.C*, 1999.
2. L. Bottou. Stochastic learning. In Olivier Bousquet and Ulrike von Luxburg, editors, *Advanced Lectures on Machine Learning*, Lecture Notes in Artificial Intelligence, LNAI 3176, pages 146–168. Springer Verlag, Berlin, 2004.
3. G. Box, G. M. Jenkins, and G. Reinsel. *Time Series Analysis: Forecasting & Control (3rd Edition)*. Prentice Hall, 3rd edition, February 1994.
4. C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman & Hall/CRC, sixth edition, July 2003.
5. T. G. Dietterich, P. Domingos, L. Getoor, S. Muggleton, and P. Tadepalli. Structured machine learning: the next ten years. *Machine Learning*, 73(1):3–23, 2008.
6. P. Domingos, S. Kok, D. Lowd, H. Poon, M. Richardson, and P. Singla. Markov logic. In *Probabilistic Inductive Logic Programming*, pages 92–117, 2008.

7. B. Ghosh, B. Basu, and M. O'Mahony. Multivariate short-term traffic flow forecasting using time-series analysis. *Trans. Intell. Transport. Sys.*, 10(2):246–254, 2009.
8. C. W. J. Granger and P. Newbold. *Forecasting Economic Time Series (Economic Theory and Mathematical Economics)*. Academic Press, 1977.
9. Z. Jia, C. Chen, B. Coifman, and P. Varaiya. The pems algorithms for accurate, real-time estimates of g-factors and speeds from single-loop detectors. pages 536 –541, 2001.
10. Y. Kamarianakis and P. Prastacos. Space-time modeling of traffic flow. *Comput. Geosci.*, 31:119–133, March 2005.
11. B. S. Kerner, C. Demir, R. G. Herrtwich, S. L. Klenov, H. Rehborn, M. Aleksic, and A. Haug. Traffic state detection with floating car data in road networks. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 44–49, 2005.
12. M. Lippi and P. Frasconi. Prediction of protein beta-residue contacts by markov logic networks with grounding-specific weights. *Bioinformatics*, 25(18):2326–2333, 2009.
13. R. B. Noland and J. W. Polak. Travel time variability: a review of theoretical and empirical issues. *Transport Reviews: A Transnational Transdisciplinary Journal*, 22:39–54, 2002.
14. J. Peng, L. Bo, and J. Xu. Conditional neural fields. In Y. Bengio, D. Schuurmans, J. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 1419–1427. 2009.
15. D. L. Selby and R. Powell. Urban traffic control system incorporating scoot: design and implementation. *Institution of Civil Engineers Proceedings*, 82:903–20, Oct 1987.
16. AG. Sims. S.c.a.t. the sydney co-ordinated adaptive traffic system. *Symposium on Computer Control of Transport 1981: Preprints of Papers*, pages 22–26, 1981.
17. B. L. Smith and M. J. Demetsky. Short-term traffic flow prediction: neural network approach. *Transportation Research Record*, 1453:98–104, 1997.
18. B. L. Smith and M. J. Demetsky. Traffic flow forecasting: Comparison of modeling approaches. *Journal of Transportation Engineering-Asce*, 123(4):261–266, Jul-Aug 1997.
19. B. L. Smith, B.M. Williams, and R. Keith Oswald. Comparison of parametric and nonparametric models for traffic flow forecasting. *Transportation Research Part C*, 10(4):303–321, 2002.
20. S. Sun, C. Zhang, and G. Yu. A bayesian network approach to traffic flow forecasting. *IEEE Transactions on Intelligent Transportation Systems*, 7(1):124–132, 2006.
21. P. Varaiya. Freeway Performance Measurement System: Final Report. PATH Working Paper UCB-ITS-PWP-2001-1, University of California Berkley, 2001.
22. S. Watson. Combining kohonen maps with arima time series models to forecast traffic flow. *Transportation Research Part C: Emerging Technologies*, 4:307–318(12), October 1996.
23. J. Whittaker, S. Garside, and K. Lindveld. Tracking and predicting a network traffic process. *International Journal of Forecasting*, 13(1):51–61, 1997.
24. C. H. Wu, J. M. Ho, and D. T. Lee. Travel-time prediction with support vector regression. *Ieee Transactions On Intelligent Transportation Systems*, 5(4):276–281, December 2004.