Mach Learn (2013) 93:115–139
DOI 10.1007/s10994-013-5399-7

Spatio-temporal random fields: compressible representation and distributed estimation

Nico Piatkowski · Sangkyun Lee · Katharina Morik

Received: 3 March 2013 / Accepted: 24 June 2013 / Published online: 25 July 2013

Abstract Modern sensing technology allows us enhanced monitoring of dynamic activities in business, traffic, and home, just to name a few. The increasing amount of sensor measurements, however, brings us the challenge for efficient data analysis. This is especially true when sensing targets can interoperate—in such cases we need learning models that can capture the relations of sensors, possibly without collecting or exchanging all data. Generative graphical models namely the Markov random fields (MRF) fit this purpose, which can represent complex spatial and temporal relations among sensors, producing interpretable answers in terms of probability. The only drawback will be the cost for inference, storing and optimizing a very large number of parameters—not uncommon when we apply them for real-world applications.

In this paper, we investigate how we can make discrete probabilistic graphical models practical for predicting sensor states in a spatio-temporal setting. A set of new ideas allows keeping the advantages of such models while achieving scalability. We first introduce a novel alternative to represent model parameters, which enables us to compress the parameter storage by removing uninformative parameters in a systematic way. For finding the best parameters via maximum likelihood estimation, we provide a separable optimization algorithm that can be performed independently in parallel in each graph node. We illustrate that the prediction quality of our suggested method is comparable to those of the standard MRF and a spatio-temporal k-nearest neighbor method, while using much less computational resources.

Keywords Regularization · Graphical models · Spatio-temporal

Editors: Hendrik Blockeel, Kristian Kersting, Siegfried Nijssen, and Filip Železný.

N. Piatkowski (✉) · S. Lee · K. Morik
TU Dortmund University, Dortmund 44227, Germany
e-mail: nico.piatkowski@tu-dortmund.de

S. Lee
e-mail: sangkyun.lee@tu-dortmund.de

K. Morik
e-mail: katharina.morik@tu-dortmund.de

1 Introduction

Sensor-based monitoring and prediction has become a hot topic in a large variety of applications. According to the slogan *Monitor, Mine, Manage* (Campbell 2011), series of data from heterogeneous sources are to be put to good use in diverse applications. A view of data mining towards *distributed sensor measurements* is presented in the book on ubiquitous knowledge discovery (May and Saitta 2010). There are several approaches to distributed stream mining based on work like, e.g., Wolff et al. (2009) or Sagy et al. (2011). The goal in these approaches is a general model (or function) which is built on the basis of local models while restricting communication costs. Most often, the global model allows to answer threshold queries, but also clustering of nodes is sometimes handled. Although the function is more complex, the model is global and not tailored for the prediction of measurements at a particular location. In contrast, we want to predict some sensor's state at some point in time given relevant previous and current measurements of itself and other sensors.

Since his influential book, David Luckham has promoted *complex event processing* successfully (Luckham 2002). Detecting events in streams of data has accordingly been modeled, e.g. in the context of monitoring hygiene in a hospital (Wang et al. 2011). However, in our case, the monitoring does not imply certain events. We do not aim at finding patterns that define an event, although they may show up as a side effect. We rather want to predict a certain state at a particular sensor or set of sensors taking into account the context of other locations and points in time. Although related, the tasks differ.

The analysis of mobile sensor measurements has been framed as *spatio-temporal trajectory mining* by, e.g., Giannotti et al. (2007). There, frequent patterns are mined from movements of pedestrians or cars. The places are not given a priori, but interesting places could be derived from frequent crossings. It does not deliver a prediction of states and no probabilities. Trajectory mining is a complementary task to that of state prediction.

Let us illustrate the task of *spatio-temporal state prediction* by an example from traffic modeling. The structure of the model is given by a street network, which represents spatial relationships. Nodes within the network represent places, where the traffic is measured over time. The state of a node is the congestion at this street segment. At training time, we do not know which place at which time needs to be predicted as "jam". Given observations of the state variables at the nodes, a model is trained. The model must answer queries for all parts of the network and all points in time. Examples of such predictions are:

- Given the traffic densities of all roads in a street network at discrete time points t_1, t_2, t_3 (e.g., Monday, Tuesday, Wednesday 8 o'clock): indicate the probabilities of traffic levels on a particular road A at another time point, not necessarily following the given ones (e.g., Thursday 7 o'clock).
- Given a traffic jam at place A at time t_s: output other places with a probability higher than 0.7 for the state "jam" in the time interval of $t_s < t < \rho$.

One particular interest lies in learning probabilistic models for answering such queries in resource constrained environments. This addresses huge amounts of data on quite fast compute facilities as well as a rather moderate data volume on embedded or ubiquitous devices.

1.1 Previous work

In this section, an overview of previous contributions to spatio-temporal modeling is given. The task of *traffic forecasting* is often solved by simulations (Marinosson et al. 2002). This presupposes a model instead of learning it. In the course of urban traffic control, events are

merely propagated that are already observed, e.g., a jam at a particular highway section results in a jam at another highway section, or the prediction is based on a physical rule that predicts a traffic jam based on a particular congestion pattern (Hafstein et al. 2004). Many approaches apply statistical time series methods like auto-regression and moving average (Williams and Hoel 2003). They do not take into account spatial relations but restrict themselves to the prediction of the state at one location given a series of observations at this particular location. An early approach is presented by Whittaker et al. (1997), using a street network topology that represents spatial relations. The training is done using simply Kalman filters, which is not as expressive as is necessary for queries like the ones above. A statistical relational learning approach to traffic forecasting uses explicit rules for modeling spatio-temporal dependencies like $congestion(+s_1, h) \wedge next(s_1, s_2) \Rightarrow congestion(+s_2, h + 1)$ (Lippi et al. 2010). Training is done by a Markov Logic Network delivering conditional probabilities of congestion classes. The discriminative model is restricted to binary classification tasks and the spatial dependencies need to be given by hand-tailored rules. Moreover, the model is not sparse and training is not scaleable. Even for a small number of sensors, training takes hours of computation. When the estimation of models for spatio-temporal data on ubiquitous devices is considered, e.g. learning to predict smartphone usage patterns based on time and visited places, minutes are the order of magnitude in demand. Hence, also this advanced approach does not yet meet the demands of the spatio-temporal prediction task in resource constrained environments.

Some geographically weighted regression or non-parametric k-nearest neighbour (kNN) methods model a task similar to spatio-temporal state prediction (Zhao and Park 2004; Gong and Wang 2002; May et al. 2008). The regression expresses the temporal dynamics and the weights express spatial distances. Another way to introduce the spatial relations into the regression is to encode the spatial network into a kernel function (Liebig et al. 2012). The kNN method by Lam et al. (2006) models correlations in spatio-temporal data not only by their spatial but also by their temporal distance. As stated for spatio-temporal state prediction task, the particular place and time in question need not be known in advance, because the lazy learner kNN determines the prediction at query time. However, also this approach does not deliver probabilities along with the predictions. For some applications, for instance, traffic prognoses for car drivers, a probabilistic assertion is not necessary. However, in applications of disaster management, the additional information of likelihood is wanted.

As is easily seen, generative models fit the task of spatio-temporal state prediction. For notational convenience, let us assume just one variable x. The *generative model* $p(x, y)$ allows to derive $p(y|x) = \frac{p(x,y)}{p(x)}$ as well as $p(x|y) = \frac{p(x,y)}{p(y)}$. In contrast, the *discriminative model* $p(y|x)$ must be trained specifically for each y. In our example, for each place, a distinct model would need to be trained. Hence, a huge set of discriminative models would be necessary to express one generative model. A discussion of discriminative versus generative models can be found in a study by Ng and Jordan (2002). Here, we refer to the capability of interpolation (e.g., between points in time) of generative models and their informativeness in delivering probability estimates instead of mere binary decisions.

Spatial relations are naturally expressed by *graphical models*. For instance, discriminative graphical models—as are conditional random fields (CRF)—have been used for object recognition over time (Douillard et al. 2007), but also generative graphical models such as the Markov random field (MRF) have been applied to video or image data (Yin and Collins 2007; Huang et al. 2008). The number of training instances does not influence the model complexity of MRF. However, the number of parameters can exceed millions easily. In particular when using MRF for spatio-temporal state prediction, the many spatial and temporal relations soon lead to inefficiency.

Now we have argued in favor of using generative graphical models that model spatial and temporal dependencies at the same time. However, there are problems which until now have prohibited this:

– the original parametrization is not well suited for producing sparse models,
– trained models tend to overfit to the training data, and
– training high-dimensional models is not feasible.

In the following, we shall recapitulate graphical models (Sect. 1.2) and regularization methods (Sect. 1.3) so that we can then introduce a new method for spatio-temporal state prediction, that does no longer suffer from the listed disadvantages.

1.2 Graphical models

The formalism of probabilistic graphical models provides a unifying framework for capturing complex dependencies among random variables, and building large-scale multivariate statistical models (Wainwright and Jordan 2007). Let $G = (V, E)$ be an undirected graph with the set of vertices V and the set of edges $E \subset V \times V$. For each node (or vertex) $v \in V$, let X_v be a random variable, taking values x_v in some space $\mathcal{X}_v$. The concatenation of all $n = |V|$ variables yields a multivariate random variable $\boldsymbol{X}$ with state space $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_n$. Training delivers a full probability distribution over the random variable $\boldsymbol{X}$. Let $\boldsymbol{\phi}$ be an indicator function or *sufficient statistic* that indicates if a configuration $\boldsymbol{x}$ obeys a certain event $\{\boldsymbol{X}_\alpha = \boldsymbol{x}_\alpha\}$ with $\alpha \subseteq V$. We use the short hand notation $\{\boldsymbol{x}_\alpha\}$ to denote the event $\{\boldsymbol{X}_\alpha = \boldsymbol{x}_\alpha\}$. The functions of $\boldsymbol{x}$ defined in the following can be also considered as functions of $\boldsymbol{X}$—we replace $\boldsymbol{x}$ by $\boldsymbol{X}$ when it makes their meaning clearer. Restricting α to vertices and edges,[1] one gets

$$\boldsymbol{\phi}_{\{v=x\}}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x}_v = x \\ 0 & \text{otherwise,} \end{cases} \qquad \boldsymbol{\phi}_{\{(v,w)=(x,y)\}}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } (\boldsymbol{x}_v, \boldsymbol{x}_w) = (x, y) \\ 0 & \text{otherwise} \end{cases}$$

with $\boldsymbol{x} \in \mathcal{X}$, $\boldsymbol{x}_v \in \mathcal{X}_v$ and $y \in \mathcal{X}_w$. Let us now define vectors for collections of those indicator functions:

$$\begin{aligned} \boldsymbol{\phi}_v(\boldsymbol{x}) &:= \big[\boldsymbol{\phi}_{\{v=x\}}(\boldsymbol{x})\big]_{x \in \mathcal{X}_v}, \\ \boldsymbol{\phi}_{(v,w)}(\boldsymbol{x}) &:= \big[\boldsymbol{\phi}_{\{(v,w)=(x,y)\}}(\boldsymbol{x})\big]_{(x,y) \in \mathcal{X}_v \times \mathcal{X}_w}, \\ \boldsymbol{\phi}(\boldsymbol{x}) &:= \big[\boldsymbol{\phi}_v(\boldsymbol{x}), \boldsymbol{\phi}_e(\boldsymbol{x}) : \forall v \in V,\ \forall e \in E\big]. \end{aligned} \tag{1}$$

The vectors are constructed for fixed but arbitrary orderings of V, E and $\mathcal{X}$. The dimension of $\boldsymbol{\phi}(\boldsymbol{x})$ is thus $d = \sum_{v \in V} |\mathcal{X}_v| + \sum_{(v,u) \in E} |\mathcal{X}_v| \times |\mathcal{X}_u|$. Now, consider a data set $\mathcal{D} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N\}$ with instances $\boldsymbol{x}^i$. Each $\boldsymbol{x}^i$ consists of an assignment to every node in the graph. It defines a full joint state of the random variable $\boldsymbol{X}$. The quantities

$$\hat{\boldsymbol{\mu}}_{\{v=x\}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\phi}_{\{v=x\}}\big(\boldsymbol{x}^i\big), \qquad \hat{\boldsymbol{\mu}}_{\{(v,w)=(x,y)\}} = \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\phi}_{\{(v,w)=(x,y)\}}\big(\boldsymbol{x}^i\big) \tag{2}$$

[1] In general, one may consider indicator functions not only for nodes and edges, but for all cliques (fully connected subgraphs) in G. Our description still applies to higher order models, since we can convert them into models using solely nodes and edges (Wainwright and Jordan 2007, Appendix E).

are known as *empirical moments* and they reflect the empirical frequency estimates of the corresponding events. We say that a given probability mass function p with base measure ν and expectation $\mathbb{E}_p[\boldsymbol{\phi}_{\{\boldsymbol{x}_\alpha\}}(\boldsymbol{x})]$ is *locally consistent* with data $\mathcal{D}$ if and only if p satisfies the *moment matching condition*

$$\mathbb{E}_p\big[\boldsymbol{\phi}_{\{\boldsymbol{x}_\alpha\}}(\boldsymbol{x})\big] = \hat{\boldsymbol{\mu}}_{\{\boldsymbol{x}_\alpha\}}, \quad \forall \alpha \in V \cup E,$$

i.e. the density p is consistent with the data w.r.t. the empirical moments. This problem is underdetermined, in that there are many densities p that are consistent with the data, so that we need a principle for choosing among them. The principle of maximum entropy is to choose, from among the densities consistent with the data, the densities p^* whose *Shannon entropy* $\mathcal{H}(p)$ is maximal. $\mathcal{H}$ is given by

$$\mathcal{H}(p) := -\int_{\mathcal{X}} p(\boldsymbol{x}) \log_2\big(p(\boldsymbol{x})\big)\nu(d\boldsymbol{x}).$$

This is turned into the constrained optimization problem

$$\max_{p\in\mathbb{P}} \mathcal{H}(p) \text{ subject to } \mathbb{E}_p\big[\boldsymbol{\phi}_{\{\boldsymbol{x}_\alpha\}}(\boldsymbol{x})\big] = \hat{\boldsymbol{\mu}}_{\{\boldsymbol{x}_\alpha\}}, \quad \forall \alpha \in V \cup E.$$

It can be shown that the optimal solution p^* takes the form of an exponential family of densities

$$p_{\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x}) = \exp\big[\big\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\boldsymbol{x})\big\rangle - A(\boldsymbol{\theta})\big],$$

parametrized by a vector $\boldsymbol{\theta} \in \mathbb{R}^d$. Note that the parameter vector $\boldsymbol{\theta}$ and the sufficient statistics vector $\boldsymbol{\phi}(\boldsymbol{x})$ have the same length d. The term $A(\boldsymbol{\theta})$ is called the *log partition function*,

$$A(\boldsymbol{\theta}) := \log \int_{\mathcal{X}} \exp\big[\big\langle \boldsymbol{\theta}, \boldsymbol{\phi}(\boldsymbol{x})\big\rangle\big]\nu(d\boldsymbol{x}),$$

which is defined with respect to a reference measure ν such that $P[X \in S] = \int_S p_{\boldsymbol{\theta}}(x)\nu(dx)$ for any measurable set S. Expanding $\boldsymbol{\phi}(\boldsymbol{x})$ by means of (1) reveals the usual density of pairwise undirected graphical models, also known as *pairwise Markov random field*

$$\begin{aligned} p_{\boldsymbol{\theta}}(\boldsymbol{X} = \boldsymbol{x}) &= \frac{1}{\exp A(\boldsymbol{\theta})} \prod_{v\in V} \exp\big[\big\langle \boldsymbol{\theta}_v, \boldsymbol{\phi}_v(\boldsymbol{x})\big\rangle\big] \prod_{(v,w)\in E} \exp\big[\big\langle \boldsymbol{\theta}_{(v,w)}, \boldsymbol{\phi}_{(v,w)}(\boldsymbol{x})\big\rangle\big] \\ &= \frac{1}{\Psi(\boldsymbol{\theta})} \prod_{v\in V} \psi_v(\boldsymbol{x}) \prod_{(v,w)\in E} \psi_{(v,w)}(\boldsymbol{x}). \end{aligned}$$

Here, $\Psi = \exp A$ is the cumulant-generating function of $p_{\boldsymbol{\theta}}$, and ψ_α are the so-called *potential functions*.

Inference, that is computing the marginal probabilities or maximum a-posteriori states of each vertex, can be carried out by message propagation algorithms (Kschischang et al. 2001; Wainwright et al. 2005; Pearl 1988) or variational methods (Wainwright and Jordan 2007). In order to fit the model on some data set, the model parameters have to be estimated. If the data set contains solely fully observed instances, the parameters may be estimated by the maximum likelihood principle. The estimation of parameters in the case of partially unobserved data is a challenging topic on its own. Here, we assume that the data set $\mathcal{D}$

contains only fully observed instances. The *likelihood* $\mathcal{L}$ and the *average log-likelihood* ℓ of parameters $\boldsymbol{\theta}$ given a set of i.i.d. data $\mathcal{D}$ are defined as

$$\mathcal{L}(\boldsymbol{\theta};\mathcal{D}) := \prod_{i=1}^{N} p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^i\right) \quad \text{and} \quad \ell(\boldsymbol{\theta};\mathcal{D}) := \frac{1}{N}\sum_{i=1}^{N} \log p_{\boldsymbol{\theta}}\left(\boldsymbol{x}^i\right) = \langle \boldsymbol{\theta}, \hat{\boldsymbol{\mu}} \rangle - A(\boldsymbol{\theta}). \tag{3}$$

The latter is usually maximized due to numerical inconveniences of $\mathcal{L}$. The most frequently applied optimization methods are iterative proportional fitting (Darroch and Ratcliff 1972), gradient descent and quasi-newton methods like LBFGS or conjugate gradient (Nocedal and Wright 2006). Section 2 will show, how to model spatio-temporal dependencies within this formalism.

1.3 Regularization

As we can see, the number of parameters in $\boldsymbol{\theta}$ grows quite rapidly as we consider more complex graphical models. A large number of parameters is generally not preferable, since it may lead to overfitting, not to mention that it becomes hard to implement a memory efficient predictor. Therefore some regularization would be necessary to achieve a sparse and robust model.

Popular choices of regularizers are the ℓ_1 and ℓ_2 norms of the parameter vector, $\|\boldsymbol{\theta}\|_1$ and $\|\boldsymbol{\theta}\|_2$. By minimizing the ℓ_1 norm, we coerce the values for less informative parameters to zero (similarly to LASSO (Tibshirani 1996)), and by the ℓ_2 norm we find smooth functions parametrized by $\boldsymbol{\theta}$ (similarly to the penalized splines (Pearce and Wand 2006)). Using both together is often referred to as the *elastic net* (Zou and Hastie 2005), which we also use in our work. For graphical models, elastic nets have been used for the task of structure learning (estimating the neighborhoods) by Cucuringu et al. (2011) in a manner similar to the approach of Meinshausen and Buehlmann (2005). For the state prediction task, there exist two short workshop papers (Piatkowski 2012; Piatkowski et al. 2012) using the elastic net. However, the analytical and empirical validation of such an approach is rather limited there.

1.4 Overview

Given the introduction of the spatio-temporal prediction task, graphical models, and regularization, we can now propose how to keep the advantages of a generative graphical model and achieve robust, sparse models by scalable training, at the same time. The major contributions of this article are:

- an interpretable model that truly captures spatio-temporal structures,
- a new parametrization that results in sparse models with regularization,
- and scalable training even for high dimensional models.

Section 2 shows how to represent spatio-temporal structures in a graphical model. Section 3 presents the parametrization that yields sparse models and then shows a parallel formulation of a scaled gradient descend with soft thresholding that enhances the scalability of training. We evaluate the performance of the suggested method in Sect. 4.

2 From linear chains to spatio-temporal models

Sequential undirected graphical models, also known as linear chains, are a popular method in the natural language processing community (Lafferty et al. 2001; Sutton and McCallum

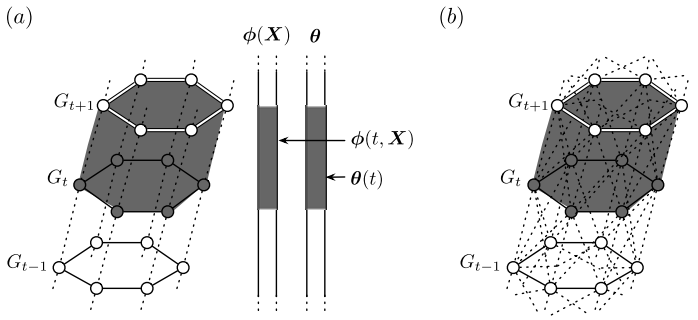

Fig. 1 A spatio-temporal model consisting of multiple snapshot graphs G_t for $t = 1, 2, \ldots, T$. The spatial and temporal edges are represented by *solid* and *dotted lines*, respectively. (**a**) A layer L_t is shown as the shaded region with simple temporal edges (L_t does not include the elements of G_{t+1}), along with the corresponding sufficient statistic and parameter subvectors $\boldsymbol{\phi}(t, X)$ and $\boldsymbol{\theta}(t)$. (**b**) An extended model with "crossing" temporal edges between consecutive snapshots. This extended model is adopted in our experiments

2007). There, consecutive words or corresponding word features are connected to a sequence of labels that reflects an underlying domain of interest like entities or part of speech tags. If we consider a sensor network G that generates measurements over space as a word, then it would be appealing to think of the instances of G at different timepoints, like words in a sentence, to form a temporal chain $G_1 - G_2 - \cdots - G_T$. We will now present a formalization of this idea followed by some obvious drawbacks. Afterwards we will discuss how to tackle those drawbacks and derive a tractable class of generative graphical models for the spatio-temporal state prediction task.

We first define the part of the graph corresponding to the time t as the *snapshot graph* $G_t = (V_t, E_t)$, for $t = 1, 2, \ldots, T$. Each snapshot graph G_t replicates a given *spatial graph* $G_0 = (V_0, E_0)$, which represents the underlying physical placement of sensors, i.e., the spatial structure of random variables that does not change over time. We also define the set of spatio-temporal edges $E_{t-1;t} \subset V_{t-1} \times V_t$ for $t = 2, \ldots, T$ and $E_{0;1} = \emptyset$, that represent dependencies between adjacent snapshot graphs G_{t-1} and G_t, assuming a Markov property among snapshots, so that $E_{t;t+h} = \emptyset$ whenever $h > 1$ for any t. Note that the actual time gap between any two time frames t and $t + 1$ can be chosen arbitrarily.

The entire graph, denoted by G, consists of the snapshot graphs G_t stacked in order for time frames $t = 1, 2, \ldots, T$ and the temporal edges connecting them: $G := (V, E)$ for $V := \bigcup_{t=1}^{T} V_t$ and $E := \bigcup_{t=1}^{T} \{E_t \cup E_{t-1;t}\}$. We sketch the structure of G in Fig. 1.

For simple description, we define a *layer* L_t as the partial subgraph of G containing all vertices of V_t and all edges of $E_t \cup E_{t;t+1}$, for $t = 1, 2, \ldots, T$. For instance, a layer L_t is depicted as a shaded region in Fig. 1. We also define the subvectors of $\boldsymbol{\phi}(X)$ and $\boldsymbol{\theta}$ that correspond to a layer L_t as follows,

$$\begin{aligned} \boldsymbol{\phi}(t, \boldsymbol{X}) &:= \big(\boldsymbol{\phi}_{v=a}(\boldsymbol{X}_v), \boldsymbol{\phi}_{(v,w)=(a,b)}(\boldsymbol{X}_v, \boldsymbol{X}_w) \mid v \in L_t,\ (v, w) \in L_t,\ a \in \mathcal{X}_v,\ b \in \mathcal{X}_w\big), \\ \boldsymbol{\theta}(t) &:= \big(\boldsymbol{\theta}_{v=a}, \boldsymbol{\theta}_{(v,w)=(a,b)} \mid v \in L_t,\ (v, w) \in L_t,\ a \in \mathcal{X}_v,\ b \in \mathcal{X}_w\big). \end{aligned} \tag{4}$$

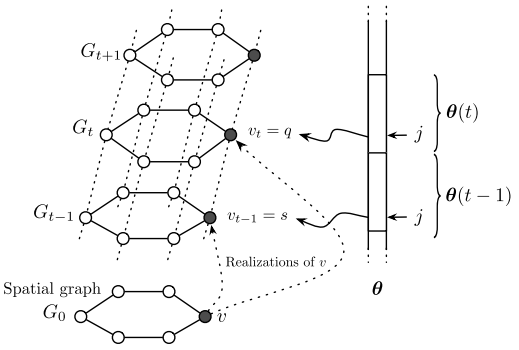

Fig. 2 An example of indexing for a node and state pair over time. A sensor modeled by the node v in the spatial graph G_0 shows its measurements v_{t-1} and v_t at time frames $t-1$ and t, respectively. The pairs $v_{t-1} = s$ and $v_t = q$ are located at the same index j in the subvectors $\boldsymbol{\theta}(t-1)$ and $\boldsymbol{\theta}(t)$

By construction, the layers $L_1, L_2, \ldots, L_T$ define a non-overlapping partitioning of a graph G, which allows us writing

$$\big\langle \boldsymbol{\phi}(\boldsymbol{X}), \boldsymbol{\theta} \big\rangle = \sum_{t=1}^{T} \big\langle \boldsymbol{\phi}(t, \boldsymbol{X}), \boldsymbol{\theta}(t) \big\rangle.$$

The subvectors $\boldsymbol{\phi}(t, \boldsymbol{X})$ and $\boldsymbol{\theta}(t)$ have the same length $d' := d/T$ for all $t = 1, 2, \ldots, T$. Note that the subvectors should be "aligned", in the sense that the jth elements in all subvectors must point to the same node:state or edge:states pair over time. We illustrate this in Fig. 2.

The spatial graph G_0 and the sizes of the vertex state spaces $\mathcal{X}_v$ determine the number of model parameters d. In order to compute this quantity, we consider the construction of G (as shown in Fig. 1(b)) from G_0. First, all vertices v and all edges (u, v) from G_0 are copied exactly T times and added to $G = (V, E)$, whereas each copy is indexed by time t, i.e. $v \in V_0 \Rightarrow v_t \in V, 1 \le t \le T$ and likewise for the edges. Then, for each vertex $v_t \in V$ with $t \le T-1$, a temporal edge (v_t, v_{t+1}) is added to G. Finally, for each edge $(v_t, u_t) \in E$ with $t \le T-1$, the two spatio-temporal edges (v_t, u_{t+1}) and (v_{t+1}, u_t) are also added to G. The number of parameters per vertex v is $|\mathcal{X}_v|$ and accordingly $|\mathcal{X}_v||\mathcal{X}_u|$ per edge (v, u). Thus, the total number of model parameters is

$$\begin{aligned} d = {} & \sum_{v \in V_0} \sum_{t=1}^{T} |\mathcal{X}_{v_t}| + \sum_{v \in V_0} \sum_{t=1}^{T-1} |\mathcal{X}_{v_t}||\mathcal{X}_{v_{t+1}}| + \sum_{(u,v) \in E_0} |\mathcal{X}_{v_T}||\mathcal{X}_{u_T}| \\ & + \sum_{(u,v) \in E_0} \sum_{t=1}^{T-1} \big(|\mathcal{X}_{v_t}||\mathcal{X}_{u_{t+1}}| + |\mathcal{X}_{v_{t+1}}||\mathcal{X}_{u_t}| + |\mathcal{X}_{v_t}||\mathcal{X}_{u_t}| \big). \end{aligned} \tag{5}$$

If we assume that all vertices $v, u \in V$ share a common state space and that state spaces do not change over time, i.e. $\mathcal{X}_{v_t} = \mathcal{X}_{u_{t'}}, \forall v, u \in V, 1 \le t, t' \le T$, the expression simplifies to

$$d = \underbrace{T|V_0||\mathcal{X}_{v_t}|}_{\text{\# of vertex parameters}} + \underbrace{\big[(T-1)\big(|V_0| + 3|E_0|\big) + |E_0|\big]|\mathcal{X}_{v_t}|^2}_{\text{\# of edge parameters}}$$

with some arbitrary but fixed vertex v_t. Note that the last two assumptions are only needed to simplify the computation of d, the spatio temporal random field that is described in the following section is not restricted by any of these assumptions.

This model now truly expresses temporal and spatial relations between all locations and points in time for all features. However, the memory requirements of such models are quite high due to the large problem dimension. Even loading or sending models may cause issues when mobile devices are considered as a platform. Furthermore, the training does not scale well because of stepsize adaption techniques that are based on sequential (i.e., non-parallel) algorithms.

3 Spatio-temporal random fields

Now we describe how we modify the naive spatio-temporal graphical model discussed above. We have two goals in mind: (i) to achieve compact models retaining the same prediction power, and (ii) to find the best of such models via scalable distributed optimization.

3.1 Toward better sparsification

The memory consumption of the MRF is dominated by the size of its parameter vector: the graph G can be stored within $\mathcal{O}(|V| + |E|)$ space (temporal edges do not have to be constructed explicitly), and the size of intermediate variables required for inference takes $\mathcal{O}(2|E||\mathcal{X}_v|)$. That is, if $|\mathcal{X}_v| \geq 2$ for all v, the dimension d in (5) and therefore the memory consumption of the parameter vector is always a dominant factor. Also, since each parameter is usually accessed multiple times during inference, it is desirable to have them in a fast storage, e.g. a cache memory.

An important observation of the parameter subvector $\boldsymbol{\theta}(t)$ is that it is unlikely to be a zero vector when it models an informative distribution. For example, if the nodes can have one of the two states {high, low}, suppose that the corresponding parameters at time t satisfy $[\boldsymbol{\theta}(t)]_v = 0$ for all v and equally for all edge weights. Then it implies $P(v = \text{high}) = P(v = \text{low})$, a uniform marginal distribution. The closer the parameters of a classical MRF are towards $\mathbf{0}$, the closer are the corresponding marginals to the uniform distribution.

When all consecutive layers are sufficiently close in time, the transition of distributions over the layers will be smooth in many real world applications. But the optimal $\boldsymbol{\theta}$ is likely to be a dense vector, and it will require large memory and possibly long time to make predictions with it as we deal with large graphical models. This brings us the call for another parametrization.

3.1.1 Reparametrization

In our reparametrization, we consider a piecewise linear representation of $\boldsymbol{\theta}(t)$ with new parameter vectors $\boldsymbol{Z}_{\cdot i} \in \mathbb{R}^{d'}$ for $i = 1, 2, \ldots, T$,

$$\boldsymbol{\theta}(t) = \sum_{i=1}^{t} \frac{1}{t - i + 1} \boldsymbol{Z}_{\cdot i}, \quad t = 1, 2, \ldots, T. \tag{6}$$

Our motivation is best shown by the differences in $\boldsymbol{\theta}$ between two consecutive layers, $\Delta_{(t-1):t} := \boldsymbol{\theta}(t) - \boldsymbol{\theta}(t-1) = \boldsymbol{Z}_{\cdot t} - \sum_{i=1}^{t-1} \frac{1}{(t-i+1)(t-i)} \boldsymbol{Z}_{\cdot i}$. That is, the difference (slope) is mostly captured by the first term $\boldsymbol{Z}_{\cdot t}$, and by the remainder terms $\boldsymbol{Z}_{\cdot(t-i)}$ with quadratically decaying weights in $\mathcal{O}(i^{-2})$, for $i = 1, 2, \ldots, t$. We note that a simpler alternative might be setting $\boldsymbol{\theta}(t) = \sum_{i=1}^{t} \boldsymbol{Z}_{\cdot i}$, but our approach leads to better conditions in optimization which allow for faster convergence. More details will be discussed in Sect. 3.2.1.

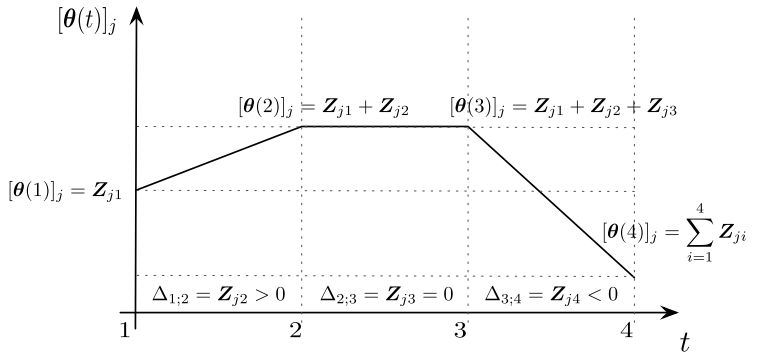

Fig. 3 A simplified example of the reparametrization of $[\boldsymbol{\theta}(t)]_j$, the jth element in the subvector $\boldsymbol{\theta}(t)$, over the timeframes $t = 1, 2, 3, 4$. We store slopes $\boldsymbol{Z}_{jt}$ instead of the actual values of the piecewise linear function $[\boldsymbol{\theta}(t)]_j$, between two consecutive timeframes $t-1$ and t (except for $\boldsymbol{Z}_{j1}$ which works as an intercept). Near-zero slopes $\boldsymbol{Z}_{jt} \approx 0$ ($\boldsymbol{Z}_{j3} = 0$ above) can be removed from computation and memory

Now with the new parameters, if the changes between two consecutive layers are near zero, that is, $\boldsymbol{\theta}(t) \approx \boldsymbol{\theta}(t-1)$, then we expect $\boldsymbol{Z}_{\cdot t} \approx 0$. This is a novel property of the new parametrization, since with the classical parameters $\boldsymbol{\theta}$ the condition does not necessarily entail $\boldsymbol{\theta}(t) \approx 0$. In other words, $\boldsymbol{Z}_{\cdot t} = 0$ implies no changes in distribution from $t-1$ to t, but $\boldsymbol{\theta}(t) = 0$ implies the distribution at t suddenly becomes a uniform distribution, regardless of the previous state at layer $t-1$. An example is illustrated in Fig. 3.

Since we have defined $\boldsymbol{\theta}$ as a concatenation of vectors $\boldsymbol{\theta}(1), \boldsymbol{\theta}(2), \ldots, \boldsymbol{\theta}(T)$, the reparametrization reads as follows,

$$\boldsymbol{\theta} = \begin{bmatrix} \boldsymbol{\theta}(1) \\ \boldsymbol{\theta}(2) \\ \vdots \\ \boldsymbol{\theta}(T) \end{bmatrix} = \begin{bmatrix} \boldsymbol{Z}_{\cdot 1} \\ \frac{1}{2}\boldsymbol{Z}_{\cdot 1} + \boldsymbol{Z}_{\cdot 2} \\ \vdots \\ \sum_{i=1}^{T} \frac{1}{t-i+1} \boldsymbol{Z}_{\cdot i} \end{bmatrix}, \quad \boldsymbol{Z} := \begin{bmatrix} | & | & & | \\ \boldsymbol{Z}_{\cdot 1} & \boldsymbol{Z}_{\cdot 2} & \cdots & \boldsymbol{Z}_{\cdot T} \\ | & | & & | \end{bmatrix}.$$

For convenience, we define the *slope matrix* $\boldsymbol{Z} \in \mathbb{R}^{d' \times T}$ as above, which contains $\boldsymbol{Z}_{\cdot 1}, \boldsymbol{Z}_{\cdot 2}, \ldots, \boldsymbol{Z}_{\cdot T}$ as its columns. In the following we sometimes use the notations $\boldsymbol{\theta}(\boldsymbol{Z})$ and $\boldsymbol{\theta}(t, \boldsymbol{Z})$, whenever it is necessary to emphasize the fact that $\boldsymbol{\theta}$ and $\boldsymbol{\theta}(t)$ are functions of $\boldsymbol{Z}$ under the new parametrization. Finally, another property of our reparametrization is that it is linear. Therefore an important property for optimization carries over: $A(\boldsymbol{\theta}(\boldsymbol{Z}))$ is convex in $\boldsymbol{Z}$ as $A(\boldsymbol{\theta})$ is convex in $\boldsymbol{\theta}$ (Wainwright and Jordan 2007).

We note that our reparametrization with $\boldsymbol{Z}$ introduces some overhead, due to the summation in (6), compared to the classical parametrization with $\boldsymbol{\theta}$. In particular, whenever an algorithm has to read a value from $\boldsymbol{\theta}$, it has do be decompressed instantly, which adds asymptotic complexity $\mathcal{O}(T)$ to every access. However, if we obtain sparse representation with $\boldsymbol{Z}$, then it can be stored in small memory (possibly even in CPU cache memory), and therefore the chances for cache misses or memory swapping will be reduced. This becomes an important factor when we deploy a learned model to applications running on mobile devices, for instance.

3.1.2 Regularizers revisited

We define the ℓ_1 and ℓ_2 regularizers for the slope matrix $\boldsymbol{Z}$ as follows,

$$\|\boldsymbol{Z}\|_1 := \sum_{j=1}^{d'} \|\boldsymbol{Z}_{j\cdot}\|_1, \qquad \|\boldsymbol{Z}\|_F^2 := \sum_{j=1}^{d'} \|\boldsymbol{Z}_{j\cdot}\|_2^2. \tag{7}$$

The latter definition is consistent with that of the Frobenius inner product between two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of the same size, $\langle \boldsymbol{A}, \boldsymbol{B}\rangle_F := \sum_i \sum_j \boldsymbol{A}_{ji}\boldsymbol{B}_{ji}$ (so that $\|\boldsymbol{A}\|_F^2 := \langle \boldsymbol{A}, \boldsymbol{A}\rangle_F$).

The two regularizers induce sparsity and smoothness respectively, as we have discussed in Sect. 1.3. The difference is that due to the reparametrization, now differences between parameters $\boldsymbol{\theta}(t-1)$ and $\boldsymbol{\theta}(t)$ are penalized, not the actual values therein—they are unlikely to be zero in general.

3.2 High-dimensional scalable learning

Maximum likelihood estimation as described in Sect. 1.2 is the basis of training. Let $\mathcal{D} := \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N\}$ be an i.i.d. data set, composed of N training instances, where each $\boldsymbol{x}^i \in \mathcal{X}$ is a fully observed joint configuration of all nodes in a given graph G. Together with the regularizers defined in (7), the optimization problem becomes as follows for given $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$,

$$\min_{\boldsymbol{Z}\in\mathbb{R}^{d'\times T}} h(\boldsymbol{Z}) := \underbrace{-\ell\big(\boldsymbol{\theta}(\boldsymbol{Z}); \mathcal{D}\big) + \frac{\lambda_2}{2}\|\boldsymbol{Z}\|_F^2}_{=:f(\boldsymbol{Z})} + \lambda_1\|\boldsymbol{Z}\|_1. \tag{8}$$

For ease of notation, we denote the "smooth" part of the objective by $f(\boldsymbol{Z})$, which is continuously differentiable everywhere. In contrast, $\|\boldsymbol{Z}\|_1$ is not differentiable at $\boldsymbol{Z} = \boldsymbol{0}$ and therefore is not smooth.

3.2.1 Separable subproblems

The parameter estimation problem in (8) is a convex minimization problem, because of the convexity of $A(\boldsymbol{\theta}(\boldsymbol{Z}))$, $\|\boldsymbol{Z}\|_1$ and $\|\boldsymbol{Z}\|_F^2$ in Z. In order to speed up the computation, distributing the optimization over several computing units would be desirable. However, in its current form the optimization of (8) is not separable over the components of $\boldsymbol{Z}$.

Following the general framework in the SpaRSA approach (Wright et al. 2009), we construct subproblems with the second-order approximation to the non-separable but smooth part $f(\boldsymbol{Z})$ of the objective function, keeping the non-smooth part $\|\boldsymbol{Z}\|_1$ intact.

In order to achieve separability, we need the property that the gradient of $f(\boldsymbol{Z})$ for each node or edge in the spatial graph G_0 can be computed independently, since otherwise we have to incur costly communication among graph elements in every iteration. The next lemma shows that we do have this property.

Lemma 1 *The partial gradient of $f(\boldsymbol{Z}) = -\ell(\boldsymbol{\theta}(\boldsymbol{Z}); \mathcal{D}) + \frac{\lambda_2}{2}\|\boldsymbol{Z}\|_F^2$ can be computed independently for each element in $\boldsymbol{Z}$, which is given for $i, i' = 1, 2, \ldots, T$ and $j = 1, 2, \ldots, d'$ by,*

$$\frac{\partial f(\boldsymbol{Z})}{\partial \boldsymbol{Z}_{ji}} = \sum_{t=i}^{T} \frac{[\boldsymbol{\mu}(t) - \hat{\boldsymbol{\mu}}(t)]_j}{t-i+1} + \lambda_2 \boldsymbol{Z}_{ji}, \tag{9a}$$

$$\frac{\partial^2 f(\boldsymbol{Z})}{\partial \boldsymbol{Z}_{ji}\partial \boldsymbol{Z}_{ji'}} = \sum_{t=i}^{T}\sum_{t'=i'}^{T} \frac{\operatorname{Cov}([\boldsymbol{\phi}(X,t)]_j, [\boldsymbol{\phi}(X,t')]_j)}{(t-i+1)(t'-i'+1)} + \lambda_2 \mathbf{1}_{\{i=i'\}}, \tag{9b}$$

where $\operatorname{Cov}(A,B)$ *is the covariance of two random variables* A *and* B*, and* $\mathbf{1}_{\{i=i'\}}$ *is an indicator function returning one if* $i=i'$*, and zero otherwise.*

Proof From (3), we have $-\ell(\boldsymbol{\theta};\mathcal{D}) = A(\boldsymbol{\theta}) - \langle \boldsymbol{\theta}, \hat{\boldsymbol{\mu}}\rangle$. For the first term, we begin with the definition of $A(\boldsymbol{\theta})$ to derive,

$$\begin{aligned}
\frac{\partial A(\boldsymbol{\theta})}{\partial \boldsymbol{Z}_{jk}} &= \frac{\partial}{\partial \boldsymbol{Z}_{jk}}\left\{\log \int_{\mathcal{X}} \exp \sum_{t=1}^{T} \big\langle \boldsymbol{\theta}(t), \boldsymbol{\phi}(\boldsymbol{x},t)\big\rangle \nu(d\boldsymbol{x})\right\} \quad \text{(using (4))} \\
&= \frac{\partial}{\partial \boldsymbol{Z}_{jk}}\left\{\log \int_{\mathcal{X}} \exp \sum_{t=1}^{T} \left\langle \sum_{i=1}^{t} \frac{1}{t-i+1}\boldsymbol{Z}_{\cdot i}, \boldsymbol{\phi}(\boldsymbol{x},t)\right\rangle \nu(d\boldsymbol{x})\right\} \quad \text{(using (6))} \\
&= \frac{\partial}{\partial \boldsymbol{Z}_{jk}}\left\{\log \int_{\mathcal{X}} \exp \sum_{i=1}^{T}\sum_{t=i}^{T} \frac{\langle \boldsymbol{Z}_{\cdot i}, \boldsymbol{\phi}(\boldsymbol{x},t)\rangle}{t-i+1} \nu(d\boldsymbol{x})\right\} \quad \text{(rearrangements)} \\
&= \int_{\mathcal{X}} \sum_{t=k}^{T} \frac{1}{t-k+1}\big[\boldsymbol{\phi}(\boldsymbol{x},t)\big]_j \frac{\exp \sum_{i=1}^{T}\sum_{t=i}^{T} \frac{\langle \boldsymbol{Z}_{\cdot i}, \boldsymbol{\phi}(\boldsymbol{x},t)\rangle}{t-i+1}}{\int_{\mathcal{X}} \exp \sum_{i=1}^{T}\sum_{t=i}^{T} \frac{\langle \boldsymbol{Z}_{\cdot i}, \boldsymbol{\phi}(\boldsymbol{x},t)\rangle}{t-i+1} \nu(d\boldsymbol{x})} \nu(d\boldsymbol{x}) \\
&= \sum_{t=k}^{T} \frac{1}{t-k+1}\mathbb{E}\big(\big[\boldsymbol{\phi}(\boldsymbol{X},t)\big]_j\big) = \sum_{t=k}^{T} \frac{1}{t-k+1}\big[\boldsymbol{\mu}(t)\big]_j.
\end{aligned}$$

The differentiation of the second term $\langle \boldsymbol{\theta}, \hat{\boldsymbol{\mu}}\rangle$ becomes clear when we use the definitions (4) and (6), and similar rearrangements,

$$\langle \boldsymbol{\theta}, \hat{\boldsymbol{\mu}}\rangle = \sum_{t=1}^{T}\big\langle \boldsymbol{\theta}(t), \hat{\boldsymbol{\mu}}(t)\big\rangle = \sum_{t=1}^{T}\left\langle \sum_{i=1}^{t} \frac{1}{t-i+1}\boldsymbol{Z}_{\cdot i}, \hat{\boldsymbol{\mu}}(t)\right\rangle = \sum_{i=1}^{T}\sum_{t=i}^{T} \frac{1}{t-i+1}\big\langle \boldsymbol{Z}_{\cdot i}, \hat{\boldsymbol{\mu}}(t)\big\rangle.$$

Therefore, we have $\frac{\partial}{\partial \boldsymbol{Z}_{jk}}\langle \boldsymbol{\theta}, \hat{\boldsymbol{\mu}}\rangle = \sum_{t=k}^{T} \frac{1}{t-k+1}[\hat{\boldsymbol{\mu}}(t)]_j$. Combining the above two results, together with the simple differentiation of the last term in $f(\boldsymbol{Z})$, leads to the first claim.

Next, starting from the derivations of $\frac{\partial A(\boldsymbol{\theta})}{\partial \boldsymbol{Z}_{jk}}$ above, we have that

$$\begin{aligned}
\frac{\partial^2 A(\boldsymbol{\theta})}{\partial \boldsymbol{Z}_{jk}\partial \boldsymbol{Z}_{jk'}} &= \int_{\mathcal{X}} \sum_{t=k}^{T} \frac{[\boldsymbol{\phi}(\boldsymbol{x},t)]_j}{t-k+1} \frac{\partial}{\partial \boldsymbol{Z}_{jk'}}\left[\frac{\exp \sum_{i=1}^{T}\sum_{t=i}^{T} \frac{\langle \boldsymbol{Z}_{\cdot i}, \boldsymbol{\phi}(\boldsymbol{x},t)\rangle}{t-i+1}}{\int_{\mathcal{X}} \exp \sum_{i=1}^{T}\sum_{t=i}^{T} \frac{\langle \boldsymbol{Z}_{\cdot i}, \boldsymbol{\phi}(\boldsymbol{x},t)\rangle}{t-i+1} \nu(d\boldsymbol{x})}\right] \nu(d\boldsymbol{x}) \\
&= \sum_{t=k}^{T}\sum_{t'=k'}^{T} \frac{\mathbb{E}([\boldsymbol{\phi}(\boldsymbol{x},t)]_j[\boldsymbol{\phi}(\boldsymbol{x},t')]_j) - \mathbb{E}([\boldsymbol{\phi}(\boldsymbol{x},t)]_j)\mathbb{E}([\boldsymbol{\phi}(\boldsymbol{x},t')]_j)}{(t-k+1)(t'-k'+1)}.
\end{aligned}$$

Using the definition of covariance and a simple differentiation of the linear term $\lambda_2 \boldsymbol{Z}_{ji}$ in (9a), we obtain the second claim. □

We define the gradient matrix $\boldsymbol{J}(\boldsymbol{Z}) := \nabla f(\boldsymbol{Z})$, where each element is given by (9a). Given an iterate $\boldsymbol{Z}^k$, we construct the kth subproblem progressively as follows to compute

the next iterate $\boldsymbol{Z}^{k+1}$:

$$\boldsymbol{Z}^{k+1} = \arg\min_{\boldsymbol{Z}\in\mathbb{R}^{d'\times T}} \langle \boldsymbol{Z} - \boldsymbol{Z}^k, J(\boldsymbol{Z}^k)\rangle_F + \frac{1}{2}\langle \boldsymbol{Z} - \boldsymbol{Z}^k, \boldsymbol{H}^k \circ (\boldsymbol{Z} - \boldsymbol{Z}^k)\rangle_F + \lambda_1 \|\boldsymbol{Z}\|_1. \tag{10}$$

Here $\boldsymbol{A} \circ \boldsymbol{B}$ is the Hadamard product between the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of the same size, and $\boldsymbol{H}^k \in \mathbb{R}^{d'\times T} > 0$ is a matrix approximating the curvature of $f(\boldsymbol{Z})$.

It is important to note that the objective in the subproblem (10) is now separable in terms of the rows of $\boldsymbol{Z}$, which correspond to the elements in the spatial graph G_0. Therefore each element in G_0 can perform its own optimization, without communicating with other elements. For each row $j = 1, 2, \ldots, d'$, we have a separated subproblem,

$$\boldsymbol{Z}_{j\cdot}^{k+1} = \arg\min_{\boldsymbol{Z}_{j\cdot}\in\mathbb{R}^T} (\boldsymbol{Z}_{j\cdot} - \boldsymbol{Z}_{j\cdot}^k)^T \boldsymbol{J}_{j\cdot}(\boldsymbol{Z}^k) + \frac{1}{2}(\boldsymbol{Z}_{j\cdot} - \boldsymbol{Z}_{j\cdot}^k)^T \boldsymbol{D}_j^k (\boldsymbol{Z}_{j\cdot} - \boldsymbol{Z}_{j\cdot}^k) + \lambda_1 \|\boldsymbol{Z}_{j\cdot}\|_1. \tag{11}$$

Here $\boldsymbol{D}_j^k$ is a diagonal matrix such that $\operatorname{diag}(\boldsymbol{D}_j^k) = \boldsymbol{H}_{j\cdot}^k$. In this way each row of $\boldsymbol{Z}^{k+1}$ can be computed independently, given the corresponding row of $\boldsymbol{J}(\boldsymbol{Z}^k)$. Also, each separated subproblem has a closed form solution:

$$\boldsymbol{Z}_{ji}^{k+1} = \operatorname{soft}\left(\boldsymbol{Z}_{ji}^k - \frac{\boldsymbol{J}_{ji}(\boldsymbol{Z}^k)}{\boldsymbol{H}_{ji}^k}, \frac{\lambda_1}{\boldsymbol{H}_{ji}^k}\right), \quad i = 1, 2, \ldots, T,$$

where $\operatorname{soft}(\cdot,\cdot)$ is often called as the *soft-thresholding* function defined by $\operatorname{soft}(z, a) := \operatorname{sgn}(z)\max\{0, |z| - a\}$ for scalars z and a, where the *signum* function $\operatorname{sgn}(z)$ returns $+1$ if $z > 0$, -1 if $z < 0$, and 0 otherwise.

When we use $\boldsymbol{D}_j^k > 0$, then the objective in (11) is strongly convex, and therefore the new iterate $\boldsymbol{Z}_{j\cdot}^{k+1}$ is uniquely determined for each row.

Before proceeding to the next topic, we discuss why our reparametrization in (6) may lead to better conditions for optimization, compared to a simple alternative setting $\boldsymbol{\theta}(t) = \sum_{i=1}^t \boldsymbol{Z}_{\cdot i}$. Since $\boldsymbol{\phi}(X, t) \in [0, 1]$ by definition, from the expression of the second derivative of $f(\boldsymbol{Z})$ in (9b), we have for any two time points $i \neq i'$ that

$$\left|\frac{\partial^2 f(\boldsymbol{Z})}{\partial \boldsymbol{Z}_{ji}\partial \boldsymbol{Z}_{ji'}}\right| \leq \sum_{t=i}^T \sum_{t'=i'}^T \frac{1}{(t-i+1)(t'-i'+1)} \in \left[1, \left(1+\log(T)\right)^2\right].$$

Here we have used the fact that the summation in the middle has the maximum value $(\sum_{t=1}^T 1/t)(\sum_{t'=1}^T 1/t') \leq (1 + \int_1^T 1/t\,dt)^2$ at $(i, i') = (1, 1)$. On the other hand, if we have used the simple alternative, we instead have that

$$\left|\frac{\partial^2 f_{\text{simple}}(\boldsymbol{Z})}{\partial \boldsymbol{Z}_{ji}\partial \boldsymbol{Z}_{ji'}}\right| \leq \sum_{t=i}^T \sum_{t'=i'}^T 1 \in \left[1, T^2\right],$$

which can be easily checked by following the arguments in the proof of Lemma 1. This implies that with the simple alternative, the differences of curvature over timepoints can be up to $\mathcal{O}(T^2)$, which grows much faster with T than the upper bound of curvature differences $\mathcal{O}((\log(T))^2)$ with our parametrization. Considering the contours of the two objective functions $f(\boldsymbol{Z})$ and $f_{\text{simple}}(\boldsymbol{Z})$ resulted from our parametrization and from the simple alternative, $f(\boldsymbol{Z})$ has better scaling: in other words, the $f_{\text{simple}}(\boldsymbol{Z})$ will look more elliptical than $f(\boldsymbol{Z})$. Since the estimation of curvature we will discuss in the next section can be poor, and

in such cases our optimization procedure reverts to steepest descent, it is important to have an objective function with better scaling for faster convergence. We refer to the textbook by Nocedal and Wright (2006, Sect. 3.3) for the relation between the scaling of objective functions and convergence rates.

3.2.2 Estimation of curvature

The subproblem (11) is constructed using a linear approximation of the non-separable smooth function $f(\boldsymbol{Z})$, to create a separable objective. The second term $\frac{1}{2}(\boldsymbol{Z}_{j\cdot} - \boldsymbol{Z}^k_{j\cdot})^T \boldsymbol{D}^k_j (\boldsymbol{Z}_{j\cdot} - \boldsymbol{Z}^k_{j\cdot})$ in the objective ensures that the next iterate $\boldsymbol{Z}^{k+1}_{j\cdot}$ is not too far from the current iterate $\boldsymbol{Z}^k_{j\cdot}$, since the linear approximation becomes less accurate on farther points.

In another view, the term augments the linear approximation with a quadratic function whose curvature is represented by a positive definite diagonal matrix $\boldsymbol{D}^k_j$. Each element of $\boldsymbol{D}^k_j$ can be computed separately, capturing the curvature between two consecutive iterates for all $i = 1, 2, \ldots, T$ and $j = 1, 2, \ldots, d'$:

$$\left[\boldsymbol{D}^k_j\right]_{ii} = \arg\min_{s\in\mathbb{R}} \left\| s\left(\boldsymbol{Z}^k_{ji} - \boldsymbol{Z}^{k-1}_{ji}\right) - \left(\boldsymbol{J}^k_{ji} - \boldsymbol{J}^{k-1}_{ji}\right)\right\|_2^2 = \frac{\langle \boldsymbol{J}^k_{ji} - \boldsymbol{J}^{k-1}_{ji}, \boldsymbol{Z}^k_{ji} - \boldsymbol{Z}^{k-1}_{ji}\rangle}{\|\boldsymbol{Z}^k_{ji} - \boldsymbol{Z}^{k-1}_{ji}\|_2^2}.$$

This is motivated from the works by Barzilai and Borwein (1988) and Wright et al. (2009), but we use a diagonal matrix to approximate the Hessian, not a scaled identity matrix as in the previous works. The difference is more important than it appears, since our gradient components have different scales over time due to our reparametrization, and the components (over time) are expected to behave differently (many becoming zero), comparing to the original parametrization.

Since the spectrum computed above is only an approximation, we project each value onto an interval defined by $0 < D_{\min} < D_{\max}$ to avoid numerical issues. In the worst case, for a certain j, we could have all (over time) curvature estimates to be $D_{\min}$ (or $D_{\max}$)—in such cases our method performs the steepest descent optimization for $\boldsymbol{Z}_{j\cdot}$.

3.2.3 Approximate line search

The line search is particularly challenging with spatio-temporal random fields, since the evaluation of $f(\boldsymbol{Z})$ at each trial point requires calling the belief propagation routine, which by itself is an iterative procedure to run until convergence. We avoid such load by considering a separable linear approximation to the difference of objective function values instead, evaluated at a trial point $\boldsymbol{Z}$ with a given iterate $\boldsymbol{Z}^k$, which becomes the lower bound of the true difference $h(\boldsymbol{Z}) - h(\boldsymbol{Z}^k)$ due to the convexity of h, that is,

$$h(\boldsymbol{Z}) - h\left(\boldsymbol{Z}^k\right) \geq \sum_{j=1}^{d'} \left\langle \boldsymbol{Z}_{j\cdot} - \boldsymbol{Z}^k_{j\cdot}, \boldsymbol{J}_{j\cdot}\left(\boldsymbol{Z}^k\right)\right\rangle + \lambda_1 \sum_{j=1}^{d'} \left(\|\boldsymbol{Z}_{j\cdot}\|_1 - \left\|\boldsymbol{Z}^k_{j\cdot}\right\|_1\right).$$

We perform an Armijo line search procedure (Nocedal and Wright 2006) until the expression in the right-hand side above becomes negative, decreasing the step size by half whenever the test fails. Due to the Taylor's theorem, our line search becomes more exact as the trial stepsizes becomes smaller.

In fact we can separate the sums into partial sums for the elements in G_0 as before, and check if a partial sum corresponding to each graph element becomes negative, performing

line search in parallel as well. This may endow a stronger condition than the lower bound itself becoming negative, but at least in our experiments it seems to compensate the inaccuracy of our approximate line search.

3.2.4 Stopping criterion and convergence

In order to check if an iterate $\boldsymbol{Z}$ satisfies the optimality condition $\mathbf{0} \in \partial h(\boldsymbol{Z})$, we define an optimality measure $\sigma(\boldsymbol{Z})$,

$$\sigma(\boldsymbol{Z}) := \frac{1}{\sqrt{d}} \min_{\boldsymbol{g} \in \partial h(\boldsymbol{Z})} \|\boldsymbol{g}\|_F. \tag{12}$$

To compute this measure, we need to find the subgradient $\boldsymbol{g}^* \in \partial h(\boldsymbol{Z})$ that has the minimal norm, and it can be constructed in a closed form as follows,

$$\left[\boldsymbol{g}^*\right]_{ji} = \begin{cases} \boldsymbol{J}_{ji}(\boldsymbol{Z}) + \operatorname{sgn}(\boldsymbol{Z}_{ji})\lambda_1 & \text{if } |\boldsymbol{Z}_{ji}| > 0, \\ \operatorname{soft}(\boldsymbol{J}_{ji}(\boldsymbol{Z}), \lambda_1) & \text{otherwise.} \end{cases}$$

We terminate our optimization with $\boldsymbol{Z}^k$ when $\sigma(\boldsymbol{Z}^k)$ falls below a given threshold.

For the result that our separated optimization eventually finds an optimal solution, we refer to Theorem 1 of Wright et al. (2009). The theorem requires that the smooth part of the objective is convex and Lipschitz continuously differentiable, and our $f(\boldsymbol{Z})$ satisfies the conditions (the second condition is easily verifiable from our Lemma 1 and from the fact that $\|\boldsymbol{\mu}(t) - \hat{\boldsymbol{\mu}}(t)\|_\infty \leq 1$ by construction).

Algorithm 1 presents the outline of our algorithm. The first parallel loop computes the empirical moments $\hat{\boldsymbol{\mu}}$ defined in (2). This computation is required once for a given data set $\mathcal{D}$, and therefore can be computed separately—in that case, we consider $\hat{\boldsymbol{\mu}}$ as another input. The second part optimizing $\boldsymbol{Z}$ summarizes individual procedures described in this section. Note that we set an arbitrary ordering to the nodes in the spatial graph G_0, in order to avoid updating parameters twice for each edge.

4 Experiments

We evaluate the performance of our suggested method on two real-world data sets, where each set is described by a spatial graph $G_0 = (V_0, E_0)$ with a set of sensors V_0 and connections E_0, and a set of historical sensor readings $\mathcal{D}$. We evaluate the two approaches Markov random field with the original parametrization (MRF) and the spatio temporal random field[2] (STRF) proposed in the present paper.

First we discuss about model training. We investigate the prediction quality and sparsity of resulting models with respect to regularization parameters. Also, the impact of separable optimization on training time is presented. Next, the quality of prediction on test sets is discussed, regarding the sparsity (and thereby the size in memory) of trained models. Finally, a qualitative analysis in terms of interpretability (non-overfitting) of our model is presented.

All experiments have been performed on a Linux system with four Intel Xeon CPUs (each with 8 cores at 2.00 GHz and 18 MB cache) and 256 GB of main memory, except for measuring test performance in Sect. 4.4—there we have used a Linux machine with smaller

[2]Our C++ source code is available at http://sfb876.tu-dortmund.de/strf.

Algorithm 1: STRF Algorithm

input: A spatial graph $G_0 = (V_0, E_0)$, its spatio-temporal graph $G = (V, E)$, and a data set $\mathcal{D} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^N\}$.

// Compute empirical moments $\hat{\boldsymbol{\mu}}$ in (2) (once for a given $\mathcal{D}$):
(parallel) for each node $v \in V_0$
  for $t = 1, 2, \ldots, T$ **do**
    // Update node moments (v_t : the realization of v at time t)
    $\hat{\boldsymbol{\mu}}_{\{v_t = x\}} \leftarrow \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\phi}_{\{v_t = x\}}(\boldsymbol{x}^i),\ \forall x \in \mathcal{X}_{v_t}$;
    // Update edge moments ($\mathcal{N}_G(v_t)$: the neighboring nodes of v_t in G)
    for each $w \in \mathcal{N}(v_t) \setminus \{v_t\}$ **do**
      $\hat{\boldsymbol{\mu}}_{\{(v_t, w) = (x, y)\}} \leftarrow \frac{1}{N} \sum_{i=1}^{N} \boldsymbol{\phi}_{\{(v_t, w) = (x, y)\}}(\boldsymbol{x}^i),\ \forall (x, y) \in \mathcal{X}_{v_t} \times \mathcal{X}_w$;

// Optimize $\boldsymbol{Z}$:
$\boldsymbol{Z}^1 \leftarrow \boldsymbol{0}$ // Initialization
Assign distinct index numbers $\mathrm{idx}(v)$ to the nodes $v \in V_0$ (in an arbitrary order);

for $k = 1, 2, \ldots$ **do**
  Compute marginal probability vector $\boldsymbol{\mu} \leftarrow \mathbb{E}_{\boldsymbol{\theta}(\boldsymbol{Z}^k)}[\boldsymbol{\phi}(X)]$:
  – Use loopy belief propagation (e.g. Kschischang et al. (2001)).
  – Parameters $\boldsymbol{\theta}(\boldsymbol{Z}^k)$ at time t has the form in (6)
  (parallel) for each node $v \in V_0$
    // To update the parameters for v and for the edges connected to v only once, we prepare the sets:
    – $\mathcal{N}^+_{G_0}(v) \leftarrow \{w : w \text{ is a neighboring node of } v \text{ in } G_0, \text{ and } \mathrm{idx}(w) > \mathrm{idx}(v)\}$
    – $\mathcal{U}(v) \leftarrow \{v\} \cup \{(v, w) : w \in \mathcal{N}^+_{G_0}(v)\}$
    for each $j \in \{1, 2, \ldots, d' = |\mathcal{U}(v)|\}$ **do**
      Compute the gradient $\boldsymbol{J}_{j\cdot}(\boldsymbol{Z}^k)$ by Lemma 1;
      Estimate diagonal curvature matrix $\boldsymbol{D}^k_j$ as in Sect. 3.2.2;
      Update parameters: for $i = 1, 2, \ldots, T$:

$$\boldsymbol{Z}^{k+1}_{ji} \leftarrow \mathrm{soft}\left(\boldsymbol{Z}^k_{ji} - \frac{\boldsymbol{J}_{ji}(\boldsymbol{Z}^k)}{[\boldsymbol{D}^k_j]_{ii}}, \frac{\lambda_1}{[\boldsymbol{D}^k_j]_{ii}}\right).$$

      Perform an approximate line search, if required, as in Sect. 3.2.3;
  If $\boldsymbol{Z}^{k+1}$ is convergent according to the optimality measure (12), then stop;

(16 GB) amount of memory and a single commodity Intel i7 CPU (at 3.40 GHz and with 8 MB cache), in order to better simulate low-memory situations.

Throughout the experiments, our STRF algorithm has produced solutions satisfying our target optimality of $\sigma(\boldsymbol{Z}) < 10^{-5}$ in term of the measure (12), within ten iterations.

4.1 Data sets

Traffic The traffic data from German North Rhine-Westphalia highways, are available at the Online Traffic Information System (http://autobahn.nrw.de) and consist of the number of vehicles and their average speed per minute, and the occupancy rate of the highway region covered by each sensor. Due to the amount of data, scalability is particularly an issue with this set. Here, the data from July to December 2010 is used, together more than 200 million sensor readings.

The highways are naturally partitioned into several segments by their departing locations. To prepare a spatial graph, we create nodes for notable segments, and consider the remaining segments as edges connecting the nodes. If there are more than one sensor assigned to the same vertex, we average the sensor readings as a measurement for the vertex. Different types of measurements are combined and discretized into four states following Marinosson et al. (2002), namly *green* which is defined by high average speed and low traffic density, *yellow* which allows slightly higher traffic density, *orange* additionally limits the average speed and *red* represents a traffic jam. After removing malfunctioning sensors, the final spatial graph contained 174 nodes and 218 edges. It is assumed that each week of collected traffic data is generated by the same underlying distribution. Thus, the spatio-temporal graph has $144 \times 7 = 1008$ layers (i.e. each weekday is sampled at 144 time points), consists of $174 \times 1008 = 175392$ nodes and $((174 + 218 \times 3) \times 1007 + 218) = 834014$ edges. Notice that the corresponding MRF model has more than 10^7 parameters, if counted by (5).

Temperature The second data set consists of data collected in March 2004 from the sensors deployed in the Intel Berkeley Research lab (the data are available at http://db.csail.mit.edu/labdata/labdata.html). The measurements consist of humidity, temperature, light, and voltage values captured every 31 seconds. One half of two million sensor readings were considered for training, and the rest for testing.

The spatial layer is constructed by means of a nearest neighbor graph, where each node represents a sensor and edges are disconnected whenever there is a wall between neighboring sensors. We also excluded sensors reported faulty in (Apiletti et al. 2011). The final spatial graph contained 48 nodes and 150 edges. We use the temperature measurements every 30 minutes, since it was the finest resolution without too many missing values, and discretized them into 21 equally sized bins. We assume that each day of collected temperature data is generated by the same underlying distribution. As a result, the spatio-temporal graph models exactly one day of temperature measurements. It has 48 layers, contains 2304 nodes and 23556 edges. Although the number of layers is smaller than in the traffic data, the corresponding model still has more than 10^7 parameters, due to the larger state space size.

4.2 Measures

Each data set has been split into a training set $\mathcal{D}_{\text{Train}}$ and a test set $\mathcal{D}_{\text{Test}}$ in an obvious way, such that the states of future time points (the test set) are predicted, given those of the past (the training set).

We measure the *instance-wise prediction accuracy* as the number of correctly predicted vertex states.

$$\text{Acc}(x, \hat{x} \mid x_U) := \frac{1}{|V| - |U|} \sum_{v \in V \setminus U} \mathbf{1}_{\{x_v = \hat{x}_v\}}.$$

Here, x is the true state of a test instance and $\hat{x}$ is a prediction. The set V contains all spatio-temporal nodes and U is a proper subset of V. Their cardinalities are expressed by $|V|$ and

$|U|$, respectively. In case of STRF and MRF, we make predictions for the traffic data via *maximum-a-posteriori* (MAP) estimation,

$$\hat{x} = \arg\max_{x \in \mathcal{X}} p_{\boldsymbol{\theta}}(x_{V \setminus U} \mid x_U).$$

To compute this quantity, we apply the max-product algorithm (Kschischang et al. 2001), often referred to as loopy belief propagation (LBP). Although LBP computes only approximate marginals and therefore MAP estimation by LBP may not be perfect (Heinemann and Globerson 2011), it suffices our purpose. For the temperature data, the marginal probabilities, as estimated by LBP, were used to compute the expected temperature for each vertex.

Based on the generative nature of our method, each prediction can be conditioned on an arbitrary subset of vertices $U \subseteq V$ for which the values have to be known at prediction time. Notice that $U = \emptyset$ is a perfectly valid choice and represents the most probable joint state of all vertices in the graph.

For kNN, the same spatio-temporal graph as for STRF and MRF is considered. For two nodes u and v, their spatio-temporal distance $d(v, u)$ is given by the number of edges of the shortest path that connects both nodes. The kNN prediction for $\boldsymbol{x}_v \mid \boldsymbol{x}_U$ (with $v \notin U$) is made by (i) computing the distance $d(\boldsymbol{x}^i, \boldsymbol{x}_U)$ from each training instance $\boldsymbol{x}^i$ to $\boldsymbol{x}_U$, which is simply the cardinality of U minus the number of matching values in $\boldsymbol{x}^i$ and $\boldsymbol{x}_U$. (ii) For each vertex u in each training instance $\boldsymbol{x}^i$, compute the distance $d_u := d(v, u) + d(\boldsymbol{x}^i, \boldsymbol{x}_U)$. (iii) Sort all vertices by d_u and return the top k. If more than one of these k states have the maximum frequency, we select one of them at random. This simple algorithm looks appealing, but it has an obvious drawback that it needs access to the complete training data for prediction. Lastly, the random prediction method selects a uniformly distributed random state for each vertex.

Training and test data are further split into distinct pairs $(\mathcal{D}^i_{\text{Train}}, \mathcal{D}^i_{\text{Test}})$ per month indexed by i, in order to compute an estimator for the prediction quality of different models. Train and test sets are generated from consecutive time intervals, e.g. $\mathcal{D}^0_{\text{Train}}$ is the first half of the first month and $\mathcal{D}^0_{\text{Test}}$ is the second half. If k of such pairs are considered, the accuracy on $\mathcal{D}_{\text{Test}}$ given $U \subset V$ is computed as

$$\text{Acc}(\mathcal{D}_{\text{Test}} \mid U) := \sum_{i=1}^{k} \frac{1}{|\mathcal{D}^i_{\text{Test}}|} \sum_{x \in \mathcal{D}^i_{\text{Test}}} \text{Acc}\big(x^i, \hat{x} \mid U\big).$$

Here, x^i is the prediction of a model that has been trained on $\mathcal{D}^i_{\text{Train}}$. To reveal the *training progress* of models, $-\ell(\boldsymbol{\theta}; \mathcal{D})$, the negative log-likelihood (3), is reported. Notice that this value is not computed for kNN due to its non-probabilistic nature.

Finally, to investigate the compressibility of models, we compute the ratio of non-zero entries of a d-dimensional parameter ω.

$$\text{NNZ Ratio}(\omega) := \frac{1}{d} \sum_{i=1}^{d} \mathbf{1}_{\{|\omega_i| > \epsilon\}}.$$

Again, if many parameters are set to the zero value, we can implement inference algorithms with sparse data structures and sparse arithmetic, saving memory and computation time. For numerical stability, we declare all parameters with magnitude smaller than $\epsilon = 10^{-8}$ as zeroes, although it is not necessary for our method STRF since it sets the values to exactly the zero value whenever possible.

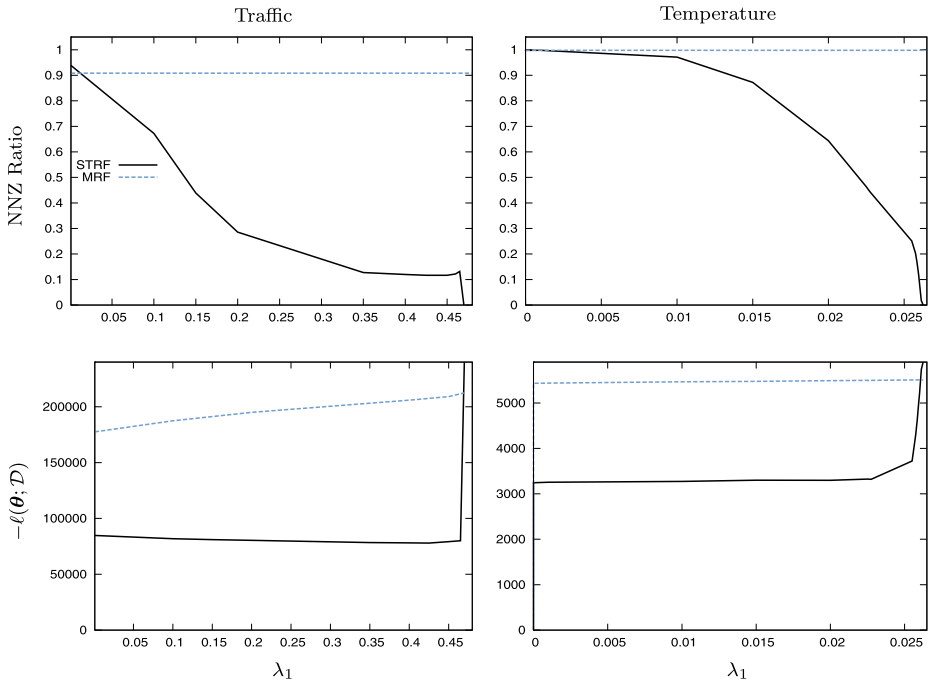

Fig. 4 The effect of regularization on models for varying sparsity parameter λ_1 (*left*: Traffic data, *right*: Temperature data, *top*: NNZ ratio, *bottom*: negative log-likelihood). All measurements were obtained after ten iterations, which was enough for STRF to reach tight optimality

4.3 Regularized training of spatio-temporal random fields

In our model, the ℓ_2 regularizer imposes "smoothness" to the dynamics of parameters over time, providing a controllable way to avoid overfitting noisy observations. The degree of smoothness is controlled by λ_2, whereas the compression ratio is controlled by λ_1. Positive values of λ_2 help in our method, since the curvature estimation in Sect. 3.2.2 becomes better conditioned.

4.3.1 Sparsity of trained models and their training accuracy

The performance of STRF (our method) and MRF (classical parametrization) in terms of the negative log-likelihood and the NNZ ratio, to a range of values for λ_1, is shown in Fig. 4. The parameter λ_2 was fixed to 10^{-1} (the characteristics were almost identical for various λ_2 values we tried in the range of $[0, 1]$). For MRF, we augmented the objective with ℓ_1 and ℓ_2 regularizers discussed in Sect. 1.3, then applied a subgradient descent method with fixed stepsize ($\eta = 10^{-2}$). Our results show that (i) the subgradient method does not properly perform regularization for MRF, regardless of the choices of (λ_1, λ_2); (ii) the negative log-likelihood decreases as λ_1 is increased, as expected, since at the strongest ℓ_1 regularization will force all marginals to be uniform distributions; (iii) our method STRF identifies sparse models accordingly to given regularization strength, while retaining similar likelihood values to MRF. More precisely, focusing on the curves for STRF, likelihood keeps improving until λ_1 reaches 0.47 (beyond this value, the model is compressed too much, losing prediction power). Overall, the pair $(\lambda_1, \lambda_2) = (0.4655, 1.0)$ with NNZ ratio 0.101573 has been

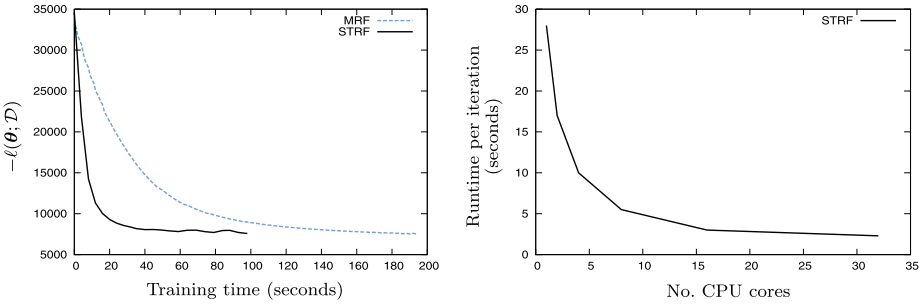

Fig. 5 The effect of separable optimization. *Left*: comparison of convergence rate in function values (using 32 CPU cores for both methods), *right*: the scalability of STRF w.r.t. the number of cores. The runtime per iteration for optimizing $\boldsymbol{Z}$ in Algorithm 1 (enumerated by k) is shown. Both STRF and MRF share the same code base and are executed on the same hardware without any additional load

identified as a good choice for the traffic data, and $(\lambda_1, \lambda_2) = (0.0255, 1.0)$ with NNZ ratio 0.248136 for the temperature data, since both lead to sparse models with reasonable likelihood values. We use these values in the following experiments.

Since the number of edge parameters is a dominant factor in the dimension d of the parameter space, it would be desirable if STRF compresses edge parameters well enough. Considering the NNZ ratio of vertex and edge parameters separately, it turns out that STRF has such a property: with the good parameter values above, the NNZ ratio of vertices is about 0.95, whereas that of edges is about 0.09.

4.3.2 Scalability of separable optimization

The number of parameters can grow quite rapidly with the size of the spatial graph, the number of states, and the number of layers. Therefore the scalability of training such models is important for practical applications, where models may have to be updated frequently for new data.

We show the characteristic of our algorithm STRF in Fig. 5, comparing it to the standard gradient descent method for MRF. For comparison, we first run MRF for 100 iterations on the traffic data with 144 layers, record the final objective value, then run STRF with $\lambda_1 = \lambda_2 = 0.1$ until it reaches a similar objective. As we can see, our method converges much faster than the gradient descent: in an ideal situation, we expect quadratic convergence from our method, compared to the linear convergence of gradient descent. The plot on the right shows how the runtime of our optimization scales with different numbers of CPU cores, illustrating the benefit of separable optimization. Experiments with other graphs and different numbers of layers showed no qualitative difference in the convergence behavior.

4.4 Prediction on test sets

Here we investigate (i) the test set performance of the sparse models, obtained with the good parameter values of λ_1 and λ_2 found in training, and (ii) how the sparsity of trained models affect the testing time.

4.4.1 Prediction quality of sparse models

The test set accuracy of the models, obtained by the regularization parameters described in Sect. 4.3.1, is presented in Fig. 6. Here our method STRF, the classical MRF, the kNN

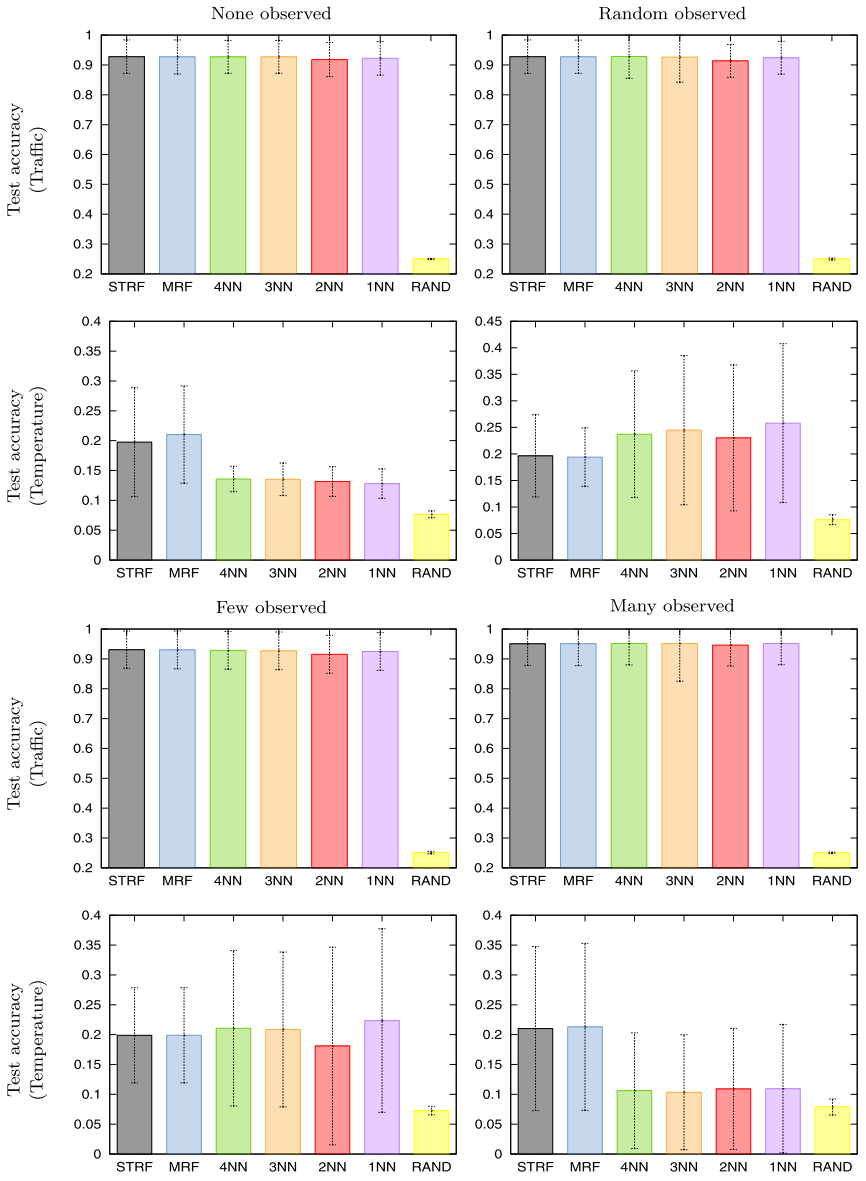

Fig. 6 Test accuracy of STRF, MRF, and k-nearest neighbor algorithm on the traffic data set for four scenarios: unconditioned (*first column*, *first two rows*), random observed layers (*second column*, *first two rows*), conditioned on Monday (*first column*, *last two rows*), conditioned on Monday to Saturday (*first column*, *last two rows*)

algorithm with several values of k, and the random guessing method, are compared. The prediction quality of the models produced by STRF is almost identical to that of MRF, although the STRF models are much smaller in size (10.2 % and 24.8 % of the MRF models in size, respectively for Traffic and Temperature). The kNN algorithm sometimes performs

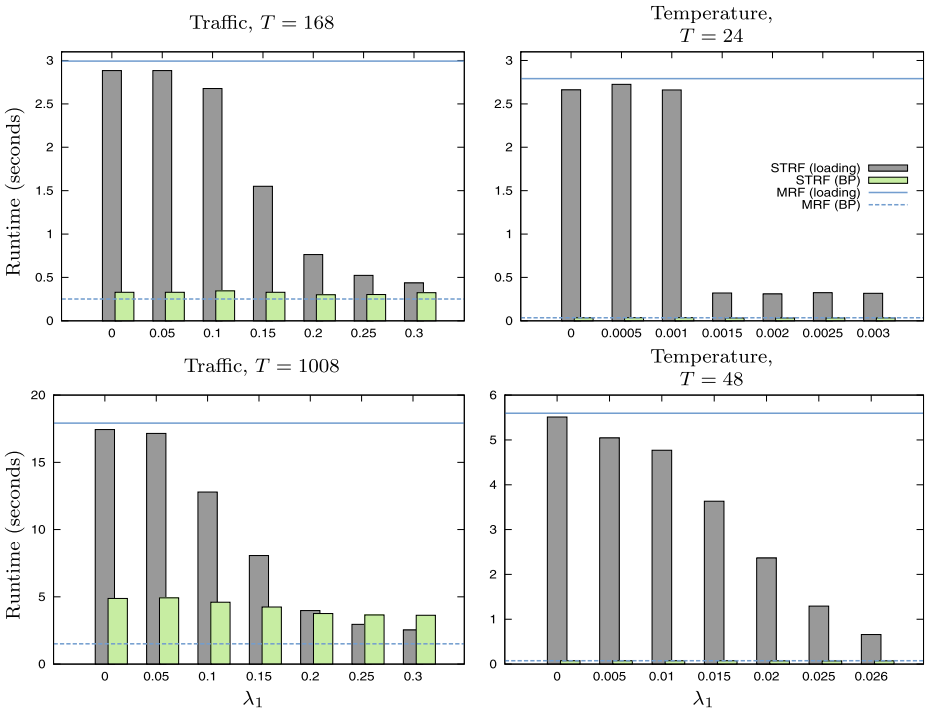

Fig. 7 Effects of sparsity to loading model parameters and belief propagation (BP) computation time for prediction. *Left*: Traffic, *right*: Temperature, *top*: small graphs, *bottom*: large graphs. MRF models do not depend on sparsity, and therefore their corresponding values are represented as *straight lines*: *solid* (loading) and *dashed* (BP)

better than STRF and MRF, but remember that kNN cannot capture probabilistic relations and requires the access to full training data, which is not the case for STRF and MRF.

4.4.2 The effect of sparsity on prediction time

As discussed in Sect. 3.1.1, our reparametrization brings in some computational overhead, especially up to $\mathcal{O}(T)$ to the message computation of the belief propagation (BP) algorithm, which is used for answering queries. However, it is not the entire picture. For example, STRF could produce a tiny model that can be stored in CPU caches, whereas MRF produces a much larger model that has to be swapped out. Then, fetching the model parameters would take much longer for MRF than for STRF, even making the extra computation time for STRF negligible—which is a likely scenario for small ubiquitous devices.

For a preliminary investigation, we use a standard Linux PC with 8 MB of cache and 16 GB of main memory, to monitor the effect of sparsity to the loading times of model parameters from disks, as well as the computation time per BP iteration. The results are shown in Fig. 7. Since loading and prediction time of MRF is not affected by regularization, they are represented as straight lines. As expected, the loading time is reduced quite significantly by sparsifying the model parameters. For instance, the MRF traffic model with $T = 168$ requires 16.18 MB and the corresponding STRF model with $(\lambda_1, \lambda_2) = (0.2, 0.1)$ only 4.04 MB. The time for a single BP iteration also reduces slightly with increased sparsity. For

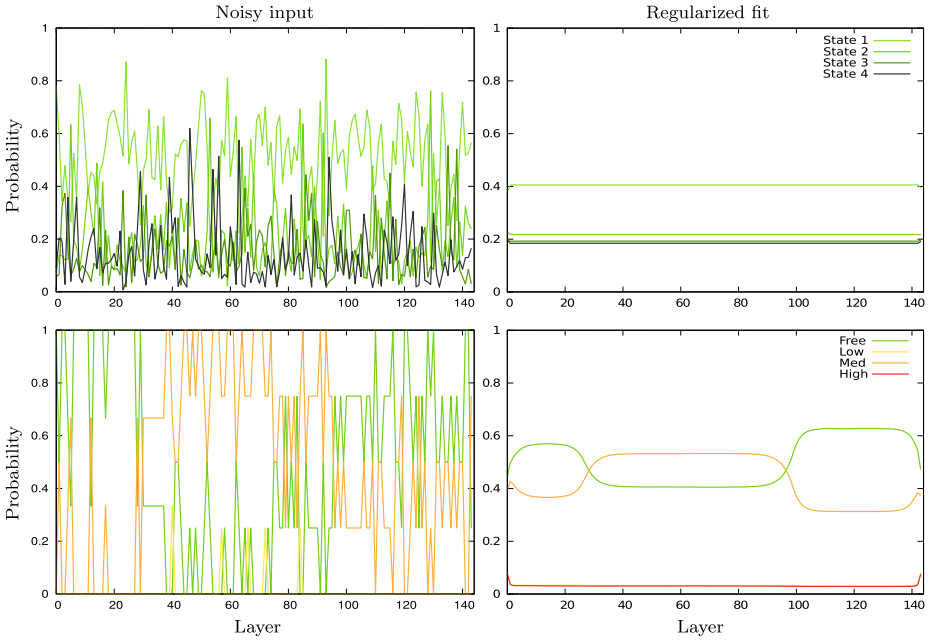

Fig. 8 Smooth marginal probabilities on synthetic data (*top*) and traffic data (*bottom*). *X-Axis*: Layer from 1 to 144, each of 10 min width. *Y-Axis*: Probability. Smoothed marginals are generated by our smooth reparametrization after 5 training iterations, $\lambda_1 = \lambda_2 = 10$

small graphs, the difference in prediction time of STRF and MRF is negligible. In case of larger graphs, STRF takes about two times longer than MRF in BP computation. Considering the loading time however, we expect that the characteristic will change more favorably to STRF, when host systems have smaller memory. Also, stronger sparsification can be tried, although it would lead to a loss in prediction accuracy.

4.5 Smooth probabilistic modelling

Graphical models can produce probabilistic answers for queries, and it is one motivation why we choose such models. Instead of the MAP prediction, the marginal probabilities $p_v(\boldsymbol{x}_v)$ may be considered for analysis.

In case of the traffic data set, the vertex marginals represent the probabilities that a certain vertex is in one of the four states {green, yellow, orange, red}. Figure 8 shows two noisy input distributions and their smooth estimates. The two plots at the bottom show empirical and smooth estimated marginal probability for a certain node from the traffic graph. One can clearly see the day and night pattern in the estimates, which is existing but not easy to see in the empirical data. The other plots show a noisy synthetic marginal for the state space $\mathcal{X}_v = \{0, 1, 2, 3\}$ with $p_v(\boldsymbol{x}_v) \propto \mathcal{N}(\exp(-x^2), 1)$ and its smooth estimate.

These very smooth results can be achieved by considering the reparametrization $\bar{\boldsymbol{\theta}}(t) := \sum_{i=1}^{t} \boldsymbol{Z}_{\cdot i}$ as mentioned in Sect. 3.1.1 in conjunction with strong regularization, i.e. $\lambda_1 = \lambda_2 = 10$. It has to be clear, that those models represent strong generalizations of the data and as a result, the corresponding test accuracies are strictly worse, compared to reparametrization (6). Compared to the STRF results from Sect. 4.4.1, the just mentioned model gets

around 20 % less accuracy. We also note that these robust models cannot be achieved with the classical MRF parametrization, due to its rigidness to regularization.

5 Conclusions

We presented an improved graphical model designed for efficient probabilistic modelling of spatio-temporal data. It is based on a novel parametrization that allows, for the first time, a regularization of spatio-temporal graphical models, such that the estimated parameters are sparse and the estimated marginal probabilities are smooth without loosing prediction accuracy. We investigated sparsity, smoothness, prediction accuracy and scalability of our model on two real world data sets. The experiments showed that our model with around 10 % of the original size retained almost the same prediction accuracy. Our method is designed to run in parallel, and scales very well with an increasing number of CPUs. Future research will consider other reparametrizations of graphical models as well as specialized inference algorithms for spatio-temporal data.

Acknowledgements We thank the reviewers for their suggestions that have helped us improve our manuscript. This work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Data Analysis", projects A1 and C1.

References

Apiletti, D., Baralis, E., & Cerquitelli, T. (2011). Energy-saving models for wireless sensor networks. *Knowledge and Information Systems*, *28*, 615–644.

Barzilai, J., & Borwein, J. M. (1988). Two-point step size gradient methods. *IMA Journal of Numerical Analysis*, *8*(1), 141–148.

Campbell, D. (2011). Is it still Big Data if it fits in my pocket? In *Proceedings of the VLDB endowment* (Vol. 4, p. 694).

Cucuringu, M., Puente, J., & Shue, D. (2011). Model selection in undirected graphical models with the elastic net.

Darroch, J. N., & Ratcliff, D. (1972). Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, *43*(5), 1470–1480.

Douillard, B., Fox, D., & Ramos, F. T. (2007). A spatio-temporal probabilistic model for multi-sensor object recognition. In *IEEE/RSJ international conference on intelligent robots and systems* (pp. 2402–2408).

Giannotti, F., Nanni, M., Pinelli, F., & Pedreschi, D. (2007). Trajectory pattern mining. In *Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 330–339).

Gong, X., & Wang, F. (2002). Three improvements on KNN-NPR for traffic flow forecasting. In *Proceedings of the 5th international conference on intelligent transportation systems* (pp. 736–740).

Hafstein, S. F., Chrobok, R., Pottmeier, A., & Mazur, M. S. F. (2004). A high-resolution cellular automata traffic simulation model with application in a freeway traffic information system. *Computer-Aided Civil and Infrastructure Engineering*, *19*(5), 338–350.

Heinemann, U., & Globerson, A. (2011). What cannot be learned with Bethe approximations. In *Proceedings of the 27th conference on uncertainty in artificial intelligence*, Barcelona, Spain.

Huang, R., Pavlovic, V., & Metaxas, D. (2008). A new spatio-temporal mrf framework for video-based object segmentation. In *The 1st international workshop on machine learning for vision-based motion analysis*.

Kschischang, F. R., Frey, B. J., & Loeliger, H. A. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, *47*(2), 498–519.

Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th international conference on machine learning* (pp. 282–289).

Lam, W. H. K., Tang, Y. F., & Tam, M. (2006). Comparison of two non-parametric models for daily traffic forecasting in Hong Kong. *Journal of Forecasting*, *25*(3), 173–192.

Liebig, T., Xu, Z., May, M., & Wrobel, S. (2012). Pedestrian quantity estimation with trajectory patterns. In *Lecture notes in computer science: Vol. 7524. Machine learning and knowledge discovery in databases* (pp. 629–643). Berlin: Springer.

Lippi, M., Bertini, M., & Frasconi, P. (2010). Collective traffic forecasting. In *Lecture notes in computer science: Vol. 6322. Machine learning and knowledge discovery in databases* (pp. 259–273). Berlin: Springer.

Luckham, D. (2002). *The power of events—an introduction to complex event processing in distributed enterprise systems*. Reading: Addison Wesley.

Marinosson, S. F., Chrobok, R., Pottmeier, A., Wahle, J., & Schreckenberg, M. (2002). Simulation framework for the autobahn traffic in North Rhine-Westphalia. In *Cellular automata—5th int. conf. on cellular automata for research and industry* (pp. 2977–2980). Berlin: Springer.

May, M. & Saitta, L. (Eds.) (2010). *Lecture notes in artificial intelligence: Vol. 6202. Ubiquitous knowledge discovery*. Berlin: Springer.

May, M., Hecker, D., Körner, C., Scheider, S., & Schulz, D. (2008). A vector-geometry based spatial knn-algorithm for traffic frequency predictions. In *Data mining workshops, international conference on data mining* (pp. 442–447).

Meinshausen, N., & Buehlmann, P. (2005). High dimensional graphs and variable selection with lasso. *The Annals of Statistics*, *34*(3), 1436–1462.

Ng, A. Y., & Jordan, M. I. (2002). On discriminative vs. generative classifiers: a comparison of logistic regression and naive Bayes. *Advances in Neural Information Processing Systems*, *14*, 841–848.

Nocedal, J., & Wright, S. J. (2006). *Numerical optimization* (2nd ed.). Berlin: Springer.

Pearce, N. D., & Wand, M. P. (2006). Penalized splines and reproducing kernel methods. *American Statistician*, *60*(3), 233–240.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. San Mateo: Morgan Kaufmann Publishers Inc.

Piatkowski, N. (2012). iST-MRF: interactive spatio-temporal probabilistic models for sensor networks. In *International workshop at ECML PKDD 2012 on instant interactive data mining*.

Piatkowski, N., Lee, S., & Morik, K. (2012). Spatio-temporal models for sustainability. In: *Proceedings of the SustKDD workshop in ACM KDD*.

Sagy, G., Keren, D., Sharfman, I., & Schuster, A. (2011). Distributed threshold querying of general functions by a difference of monotonic representation. In: *Proceedings of the VLDB endowment* (Vol. 4).

Sutton, C., & McCallum, A. (2007). An introduction to conditional random fields for relational learning. In *Introduction to statistical relational learning*. Cambridge: MIT Press.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, *58*, 267–288.

Wainwright, M., Jaakkola, T., & Willsky, A. (2005). A new class of upper bounds on the log partition function. *IEEE Transactions on Information Theory*, *51*(7), 2313–2335.

Wainwright, M. J., & Jordan, M. I. (2007). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, *1*(1–2), 1–305.

Wang, D., Rundensteiner, E. A., & Ellison, R. T. (2011). Active complex event processing of event streams. In: *Procs. of the VLDB endowment* (Vol. 4).

Whittaker, J., Garside, S., & Lindveld, K. (1997). Tracking and predicting a network traffic process. *International Journal of Forecasting*, *13*(1), 51–61.

Williams, B., & Hoel, L. (2003). Modeling and forecasting vehicular traffic flow as a seasonal arima process: theoretical basis and empirical results. *Journal of Transportation Engineering*, *129*(6), 664–672.

Wolff, R., Badhuri, K., & Kargupta, H. (2009). A generic local algorithm for mining data streams in large distributed systems. *IEEE Transactions on Knowledge and Data Engineering*, *21*(4), 465–478.

Wright, S. J., Nowak, R. D., & Figueiredo, M. A. T. (2009). Sparse reconstruction by separable approximation. *IEEE Transactions on Signal Processing*, *57*, 2479–2493.

Yin, Z., & Collins, R. (2007). Belief propagation in a 3D spatio-temporal MRF for moving object detection. *IEEE Computer Vision and Pattern Recognition*.

Zhao, F., & Park, N. (2004). Using geographically weighted regression models to estimate annual average daily traffic. *Journal of the Transportation Research Board*, *1879*(12), 99–107.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society, Series B*, *67*, 301–320.