Graphical Models for Statistical Inference and Data Assimilation

Alexander T. Ihler [a] Sergey Kirshner [b] Michael Ghil [c,d]
Andrew W. Robertson [e] Padhraic Smyth [a]

[a] *Donald Bren School of Information and Computer Science, University of California, Irvine, U.S.A.*

[b] *Alberta Ingenuity Centre for Machine Learning, Department of Computing Science University of Alberta, Edmonton, Alberta, Canada T6G 2E8*

[c] *Department of Atmospheric and Oceanic Sciences and Institute of Geophysics and Planetary Physics, University of California, Los Angeles, U.S.A.*

[d] *Département Terre-Atmosphère-Océan and Laboratoire de Métórologie Dynamique/IPSL, Ecole Normale Supérieure, F-75231 Paris Cedex 05, FRANCE*

[e] *International Research Institute for Climate Prediction, The Earth Institute at Columbia University, Palisades, NY, U.S.A.*

Abstract

In data assimilation for a system which evolves in time, one combines past and current observations with a model of the dynamics of the system, in order to improve the simulation of the system as well as any future predictions about it. From a statistical point of view, this process can be regarded as estimating many random variables, which are related both spatially and temporally: given observations of some of these variables, typically corresponding to times past, we require estimates of several others, typically corresponding to future times.

Graphical models have emerged as an effective formalism for assisting in these types of inference tasks, particularly for large numbers of random variables. Graphical models provide a means of representing *dependency structure* among the variables, and can provide both intuition and efficiency in estimation and other inference computations. We provide an overview and introduction to graphical models, and describe how they can be used to represent statistical dependency and how the resulting structure can be used to organize computation. The relation between statistical inference using graphical models and optimal sequential estimation algorithms such as Kalman filtering is discussed. We then give several additional examples of how graphical models can be applied to climate dynamics, specifically estimation using multi-resolution models of large–scale data sets such as satellite imagery, and learning hidden Markov models to capture rainfall patterns in space and time.

Key words: data assimilation, graphical models, hidden Markov models, Kalman

filtering, statistical inference
PACS:

1 Introduction

This paper provides a tutorial introduction to graphical models in the broad context of geoscientific data analysis. Graphical models provide a graph-theoretic framework for representing dependencies among sets of random variables as well as general purpose algorithms for inference and estimation. The paper focuses in particular on how graphical models can be used as a general representation for a broad class of state–space models.

1.1 Background and Motivation

Data assimilation can be viewed as the procedure by which observed data measurements are mapped into the space of a dynamical model, in order to update (or improve) the model's estimate of the state–space vector. An estimate of the state vector x_{t-1} at time $t-1$ is propagated forward in time by the dynamical model to yield an estimate x_t at time t, which is updated using observed data measurements y_t at time t. This approach can then be applied recursively to propagate information about the updated state forward to time $t+1$, combine this with data measurements obtained at time $t+1$, and so on. In atmospheric and oceanic applications the data observations are usually spatial in nature, the state vector represents the true (but unknown) state of the atmosphere and ocean in the model space, and the dynamical model captures the physics of how the state evolves over time. Given the inherent uncertainties in the problem (such as measurement noise and errors due to approximations in the physical model) it is often worthwhile to adopt a probabilistic viewpoint and work with the conditional probability distribution [1] of the state at time t given all observations up to time t, denoted $p(x_t|y_1, \ldots, y_t)$ (e.g., see [1–4]). This tracking of (or monitoring) of the state vector over time is known as *filtering* in inference terminology.

For models with linear dynamics and Gaussian noise the probabilistic computations for sequential state estimation can be performed using the Kalman filter recursions [5–7]. The Kalman filter model can be represented by a graphical model that depicts the evolution of the model state vector x_t according

[1] We will abuse notation slightly by using $p(x_t)$ to indicate either the probability distribution of x_t when x_t takes on a continuous set of values, or the probability mass function of x_t when it takes on one of a discrete set of values.

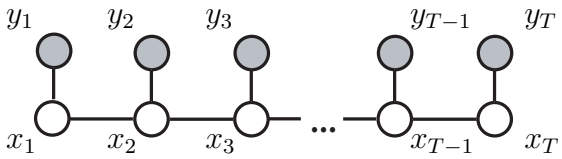


Fig. 1. The Kalman filter operates on a graphical model called a hidden Markov chain (the x_t state variables form a Markov chain that is unobserved or hidden), in which the observations y_t at each time t are used to update the current estimate of the model state x_t. Shaded nodes indicate variables with observed values.

to the model dynamics, together with the updating by observational data y_t (see Figure 1). The graphical model shows dependencies (and independencies) between variables, here between a dynamical model's evolution and available observations, as a basis for inference.

While linear and Gaussian models are a useful starting point, they have well–known limitations when applied to real data assimilation problems in the atmospheric and oceanic sciences. The main limitations are dynamical and computational: first, planetary flows, atmospheric and oceanic, are nonlinear [8–10], and hence give rise to non-Gaussian processes; second, the number of variables necessary to simulate these flows in detail is huge, and the requirement of computing the evolution of the covariance matrices is therefore prohibitive.

Efforts to overcome the dynamical obstacles include methods such as the extended Kalman filter to capture non-linearities [11–15] and ensemble methods to characterize non-Gaussian distributions (e.g., [16,2,17,18]). Methods to overcome the computational difficulties arising from large covariance matrices involve low–rank and partitioning approximations [19,20], as well as parallelization techniques [21,22]; reviews include [23,24]. At the same time, data assimilation is perceived more and more as a powerful strategy in advancing understanding, simulation and prediction in the earth sciences as a whole [25] and beyond [26].

Data assimilation in the atmospheric and oceanic sciences is widely used for the initialization of numerical weather prediction (NWP) models with observed data. However, its relevance within climate science is much broader. As pointed out by Stephenson et al. [27], the observable variables that we are interested in predicting, like precipitation at a particular location, are not the same mathematical quantities used by NWP models to predict weather or by general circulation models (GCMs) to simulate climate. These models resolve large–scale climatic fields spectrally or on a grid. Thus, the state vector of the observation state space is different from the state vector of the GCM state space. The latter is the one addressed by traditional data assimilation, while the former is one we are often more interested in, if we want to make use of the climate forecasts. Stephenson et al. [27] coin the term *forecast assimilation* for the procedure of mapping GCM predictions into the observational

space. They demonstrate the mathematical duality with traditional data assimilation, used to map observations into the model state space to initialize the NWP model or GCM and make a weather or climate prediction. Forecast assimilation refers to the calibration and tailoring of GCM forecasts, to the appropriate real–world space, which may vary by context.

The graphical models described in this paper are often motivated by the desire to build simplified probabilistic models of this observed state space, and tend to be characterized by simple assumptions, rather than sets of partial differential equations. The real–world observational space consists of data in ever increasing quantities, where identification of dependencies between variables is a key step toward producing a concise description of the data, and one amenable to prediction. Graphical models provide a principled way to analyze such dependencies and incorporate them into predictive models. For example, a Markovian assumption in time, while highly simplifying, yields a surprisingly good description of daily time dependence of rainfall occurrence, as illustrated in one of the examples in this paper.

The real–world observational space is not limited to meteorological variables. In the application of forecasts to climate–related risk management in agriculture or health, for example, the space of interest could be malaria incidence over a particular region [28]. Dependencies between disease and meteorological variables would need to be isolated and modeled, perhaps in terms of GCM model outputs. The assimilation of these GCM outputs together with observed data into a probabilistic model of disease incidence is a problem in data assimilation, for which the structure of the model may be empirical, or involve physically–based equations.

1.2 Graphical Models

The focus in this paper is somewhat different to that of traditional work in data assimilation in that we are interested in a broader and more general characterization of the nature of probability computations (or *inference*) involving an observed sequence of measurements and a parallel unobserved state–sequence. Important inference goals include not only the estimation of hidden (unobservable) random variables x_t such as the complete model state, whether in the past (called *smoothing*), present (*filtering*), or future (*prediction*), but also goals such as the prediction or simulation of the observable variables y_t in the future (which by definition have not yet been observed). The paper's motivation arises from questions such as "how could we add more complex (e.g., non-Markov) dependencies into a state–space model?", "what is the appropriate systematic approach to handling missing data observations?", and "how should parameter estimation be performed in such models?".

In this context, the goal of the paper is to provide a tutorial review of *graphical models*, a general framework for representing and manipulating joint distributions defined over sets of random variables [29–34]. Graphical models have been adopted in recent years across a wide variety of application areas, including genetics [35,36], error–correcting codes [37], speech processing [38], image analysis [39–41], and computational biology [42,43]. Indeed, graphical models (in the form of Markov random fields) have been used in data assimilation problems as well, for example [44]. These applications have in common the fact that they involve complex models with large numbers of random variables. However, despite the typically large numbers of variables involved, the dependency relations among these variables are often highly structured. This structure can in turn be leveraged by the graphical model framework to yield highly efficient computational algorithms for making inferences about unobserved quantities given observed measurements. Both the representational aspects of graphical models (how to efficiently represent complex independence relations) and the computational aspects (how to use the independence relations to infer information about state variables given data) make graphical models a potentially valuable tool in the computational geoscience toolbox.

To motivate the rest of the paper we briefly review the main aspects of graphical models below, leaving a more detailed description to subsequent sections in the paper. The principles that underlie graphical models can be summarized as follows:

- each variable of interest is associated with a node (vertex) in a graph. More generally, nodes can be associated with sets or vectors of random variables when this is appropriate. For example, in Figure 1, each of the x_t nodes could represent d variables at time t in a d-dimensional state vector x_t, and similarly the y_t nodes could also represent a multidimensional variable.
- edges in the graph represent dependencies between random variables in the joint distribution model; or, conversely, absence of edges in the graph represent conditional independence relations among random variables;
- the joint probability model can be factored into products of local functions defined on nodes and their immediate neighbors;
- problems involving computation of quantities related to the joint probability model can be profitably cast within a graph–theoretic framework. Such computations include calculation of conditional probabilities or expectations of variables of interest given observations of others. In particular, as we will show later in this paper, the underlying structure of the graph is closely related to the underlying computational complexity of an inference problem.

Thus, a graphical model consists of two parts: (1) a graph structure (nodes and edges) such that the connectivity of the graph characterizes the dependency relations in an underlying probability model, and (2) functions defined on local neighborhoods of nodes on the graph that parameterize local components of

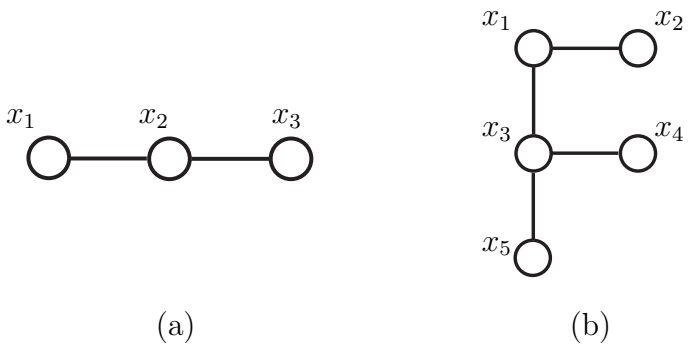


Fig. 2. Two simple examples of graphical models.

the overall joint probability model.

1.3 Examples of Graphical Models

As a simple example, consider three random variables that form a Markov chain such that $p(x_1, x_2, x_3) = p(x_1)p(x_2|x_1)p(x_3|x_2)$. We leave the functional forms of $p(x_2|x_1)$, etc., unspecified at this point and focus instead on the structure of the model. The associated graphical model has 3 nodes, one for each variable, and the graph structure is a chain (Figure 2(a)): x_1 and x_2 have direct dependence, as do x_2 and x_3, and x_1 and x_3 are conditionally independent given x_2. If we want to compute the probability of x_3 given a particular value for x_1 we can do so by marginalizing [2] over x_2, i.e., $p(x_3|x_1) = \int p(x_3, x_2|x_1)dx_2 = \int p(x_3|x_2)p(x_2|x_1)dx_2$. In terms of the graph we can view this computation as representing information flowing from x_1 through x_2 to x_3, a graphical interpretation of the well–known recursion for updating probabilities in a Markov chain.

For the most part, we shall keep the interpretation of these variables x_i somewhat abstract, with specific examples in Section 4. These variables may represent quantities related by time, space, or both, and may be scalar or vector-valued. In general, there is a trade-off between graph complexity and the dimensionality of the variables (see Section 3.3).

Consider now the more complex model in Figure 2(b), a graph that is more tree-like in structure (i.e., it cannot be arranged in a linear fashion, but also does not contain any loops, or *cycles*). We can represent the joint distribution for this model as a product of local factors, such as,

$$p(x_1, x_2, x_3, x_4, x_5) = p(x_5|x_3)p(x_4|x_3)p(x_3|x_1)p(x_2|x_1)p(x_1). \qquad (1)$$

Now consider, for example computing $p(x_5|x_1)$, i.e., the probability distribu-

[2] Of course, if x_2 is discrete–valued, this integration is simply a sum.

tion for x_5 conditioned on a specific value for x_1. By the law of total probability,

$$p(x_5|x_1) = \iiint p(x_2, x_3, x_4, x_5|x_1)dx_2dx_3dx_4.$$

In this form this equation requires integration over the 3-dimensional space x_2, x_3, x_4—more generally, if there were k variables in the model, we would have to integrate over a $(k-2)$-dimensional space to compute $p(x_5|x_1)$. However, it is clear that we can use the structure of the graph to reduce the amount of computation that is needed: specifically, by using the conditional independence relations that are implicitly defined by the graph structure (and by Equation (1)) we get

$$\begin{aligned}
p(x_5|x_1) &= \iiint p(x_2|x_1)p(x_3, x_4, x_5|x_1)dx_2dx_3dx_4 \\
&= \iint \left[\int p(x_2|x_1)dx_2\right] p(x_5|x_3)p(x_4|x_1)p(x_3|x_1)dx_3dx_4 \\
&= \int p(x_5|x_3)p(x_3|x_1)\left[\int p(x_4|x_1)dx_4\right]dx_3 \\
&= \int p(x_5|x_3)p(x_3|x_1)dx_3
\end{aligned}$$

which only requires a 1-dimensional integration over x_3. Note that while we computed here the conditional distribution $p(x_5|x_1)$ (by *marginalization* over the other unobserved random variables), we could instead have computed other quantities of interest about x_5 conditioned on x_1, such as the conditional mean or the conditional mode. The same type of factorization used above to compute a conditional density can also be leveraged to derive quantities such as conditional means or modes in an efficient manner. We will refer to these types of computations (whether computing distributions, expectations, modes, etc.) as *inference*. A key point is that inference algorithms can directly leverage structure in the underlying dependency graph for the purposes of efficient computation.

In the simple example above it is visually intuitive from the structure of the graph in Figure 2(a) that only a single variable x_3 needs to be marginalized to update x_5 given x_1. However, when there are many variables in the problem (for example thousands or even millions) and they involve complex graph structures (such as various forms of chains, tree structures, and even graphs with cycles) it is not necessarily obvious *a priori* (1) how to derive a computationally–efficient inference scheme, or (2) whether an efficient computational scheme even exists.

As we will illustrate later in the paper, the framework of graphical models allows to answer both of these questions in a general manner. The structure of the underlying graph provides the answer to the second question: generally

speaking, graphs without cycles lead to very efficient inference schemes. For graphs with cycles the situation is more complex, but sparsity (having fewer edges in the graph) helps—the sparser the graph the less computational effort is generally required to perform inference. The impact of cycles and sparsity on inference algorithms are discussed more fully in Section 3.3. The first question, how to find an efficient computational scheme given a graph (if one exists), is straightforward for graphs without cycles, and again, for graphs with cycles tends to be easier when the graph is sparse.

1.4 Outline of the Paper

To orient the reader, we give a short outline of the sections which follow. Section 2 begins with a tutorial discussion on how graphical models provide a general framework for efficiently representing dependency relations among sets of random variables. Section 3 then builds on these ideas to show how inference calculations can be performed that take advantage of the underlying graphical structure in a model. In Section 4 we illustrate these ideas in the context of two applications of graphical models to real–world data, involving assimilation of ocean data using tree–structured models, and modeling and simulation of station precipitation data using hidden Markov models. Section 5 concludes the paper with a brief discussion and summary.

2 Graphical Models as a Representation Language for Sets of Random Variables

Graphical models provide a convenient framework for representing structure within a probability distribution over a collection of random variables. In particular, for a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ we associate the nodes $\mathcal{V}$ with the random variables of interest, and use the edges $\mathcal{E}$ to indicate the factorization of their distribution. This factorization, or decomposition of the joint distribution over all variables into a product of smaller functions, can then be used to organize the computations required to perform inference, e.g., estimating the likely values of some variables given observations of others.

Since each node s is associated with a random variable x_s, we will frequently make no distinction between the two. Additionally, we can correspond sets of nodes with their (collection of) random variables, so that for any set $A \subseteq \mathcal{V}$ we have $x_A = \{x_s : s \in A\}$.

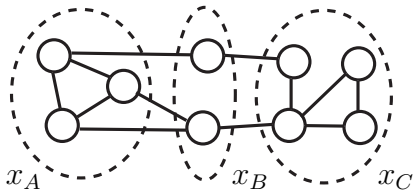


Fig. 3. Graph separation indicates conditional independence. Since every path between sets A and C passes through B, we have that the distribution factors as $p(x_A, x_B, x_C) = p(x_B)p(x_A|x_B)p(x_C|x_B)$.

2.1 Markov Random Fields

Graphical models may be formulated in any of several closely related ways. We shall restrict our attention to *undirected* graphical models, specifically Markov random fields, in which we make no distinction between the edge (s, t) and the edge (t, s). If $(s, t) \in \mathcal{E}$, we say that s and t are *adjacent*, and those nodes which are adjacent to t are called its *neighbors*, denoted $\Gamma_t \subseteq \mathcal{V}$.

It is also possible to define graphical models using directed edges (e.g., Bayes nets [30]), in which case many of the same (or similar) results hold as for undirected graphs, though there are some minor differences. For example, directed graphs are sometimes better able to represent causative relationships, while undirected graphs may be better able to represent purely correlative relationships. However, it is easy to convert any directed graphical model into an undirected one, although unfortunately some of the useful structure may be lost in the process [29,45].

The relationship between the structure of the joint distribution $p(\cdot)$ over all variables $x_\mathcal{V}$ and the graph $\mathcal{G}$ is succinctly described by a Markov property—the distribution p is said to be Markov with respect to $\mathcal{G}$ if *separation* within the graph implies conditional independence in the distribution. To be precise, let $A, B, C \subset \mathcal{V}$ be three sets of nodes, and x_A, x_B, x_C be the random variables associated with each set. Then, if every path, or sequence of non-repeating, adjacent nodes (i.e., $v_1, \ldots, v_L$ with $(v_i, v_{i+1}) \in \mathcal{E}$ and $v_i \neq v_j$ for $i \neq j$) from any node in A to any node in C pass through at least one node in B (see Figure 3), we require the joint distribution to factor as

$$p(x_A, x_B, x_C) = p(x_B)p(x_A|x_B)p(x_C|x_B),$$

i.e., that the variables x_A and x_C are conditionally independent given the variables x_B.

Given a factorization of $p(\cdot)$, it is easy to guarantee this Markov property with a simple constructive procedure for $\mathcal{G}$, by connecting each pair of nodes by an edge if their associated random variables are both arguments to a common

factor [3]. Turning this statement around, we have that given a graph $\mathcal{G}$, $p(\cdot)$ is Markov with respect to $\mathcal{G}$ if $p(\cdot)$ can be written as a product of functions defined on the *cliques*, or fully–connected subsets of nodes, as

$$p(x_{\mathcal{V}}) = \prod_{C \in \mathcal{C}} \psi_C(x_C) \tag{2}$$

where $\mathcal{C}$ is the set of all cliques in $\mathcal{G}$ and the ψ_C are non-negative functions called *potentials*. Moreover, when $p(\cdot)$ is strictly positive (as, for example, in jointly Gaussian models) the form (2) is a necessary condition as well [47]. Interestingly, for jointly Gaussian models, the structure of the graph $\mathcal{G}$ is echoed in the structure of the *inverse* of the covariance matrix of $x_{\mathcal{V}}$, in the sense that the $(s,t)^{th}$ entry of this matrix is non-zero if and only if $(s,t) \in \mathcal{E}$ [31,48,44].

Thus, the graph $\mathcal{G}$ specifies a factorization of $p(\cdot)$, while the potential functions ψ define the terms of that factorization. To illustrate this more clearly, we examine a Gaussian auto–regressive process (whose graphical model is a simple Markov chain) in Section 2.3. First, however, we describe the important subclass of graphical models formed when $\mathcal{G}$ is required to be a tree (and of which the Markov chain is a member).

2.2 Tree–Structured Graphical Models

It turns out that many probabilistic and statistical operations are greatly simplified when the graph structure associated with the random variables possesses no *cycles*. Specifically, a cycle in $\mathcal{G}$ is any non-trivial path which begins and ends at the same node. If a graph $\mathcal{G}$ contains no cycles, it is called *tree–structured* [4].

Tree–structured graphs have a number of nice properties which we shall exploit in later sections. For example, if $\mathcal{G}$ is tree–structured, then each pair of nodes s and t are joined by a unique path (if any such path exists), and thus by the graph separation property, x_s and x_t are conditionally independent given any of the variables along this path. Additionally, the cliques $\mathcal{C}$ of a tree–structured graph consist solely of single nodes and pairs of nodes, so that each term in (2)

[3] In some graphical models these "factors" of the joint distribution are explicitly represented in the graph as additional nodes; see, e.g., factor graphs [46]. However, for most graphical models (including those considered here) the factors are associated implicitly with sets of edges in the graph.

[4] If $\mathcal{G}$ is connected, i.e., there is a path between every pair of nodes s and t, it is called a *tree*; if not, it is called a *forest* (a collection of trees). Since the distinction is not important for statistical graphs, we refer to both cases alike as "tree–structured".

is a function of at most two variables. We use this fact to write each ψ_C in (2) as either $\psi_s(x_s)$ for some $s \in \mathcal{V}$ or $\psi_{st}(x_s, x_t)$ for some $(s,t) \in \mathcal{E}$.

Finally, any tree–structured graph can be used to provide a partial ordering of its nodes. Designating an arbitrary node s in the graph as the *root*, we can designate its neighbors $t \in \Gamma_s$ as its *children* (s is similarly called the *parent* of t), the neighbors of each t except s ($\Gamma_t \setminus s$) as the children of t, and so forth. Nodes which have no children are called *leaf* nodes of the graph.

These properties make inference in tree-structured graphs relatively efficient. In particular, in Section 3.2 we shall see how the ordering defined by parents and children can be used to construct efficient schedules of computation in the graph, while path uniqueness allows us to easily construct compact (low–dimensional) sufficient statistics for the computations.

2.3 Markov Chains

To see how a graphical model can be specified in practice, we examine the familiar example of a linear, auto–regressive state space model with Gaussian noise. In particular, let us define a collection of (possibly vector–valued) random variables by the recursion

$$\begin{aligned} x_{t+1} &= Ax_t + w_t \\ y_t &= Cx_t + v_t \end{aligned} \tag{3}$$

for t in some fixed interval $\{0, \ldots, T\}$, and where the quantities w_t and v_t are samples of zero–mean white Gaussian noise with covariances Q and R (respectively) and the uncertainty on the initial vector x_0 is also Gaussian with mean μ_0 and covariance Σ_0. As is common, here the variables x_t represent a "hidden" state of the system which can only be observed indirectly through the quantities y_t.

It is well–known that for such a state–space model, the variables $\{x_t\}$ for all t form a *Markov chain*—specifically, the joint distribution factors as

$$p(x_0, y_0, \ldots, x_T, y_T) = p(x_0)p(y_0|x_0) \prod_{t=1}^{T} p(x_t|x_{t-1})p(y_t|x_t) \tag{4}$$

We can graphically represent this structure by associating open circles with the hidden random variables x_t and filled circles with the observed random variables y_t, connecting by an edge each pair of variables if they are related by one of the terms in (4), as described previously. This procedure results in the same graphical model depicted in Figure 1.

From this graphical model, the graph separation condition makes it easy to see a well–known property of Markov chains, that given the state x_{t_1} at some time t_1, the "past" and "future" states are conditionally independent:

$$p(x_{t_0}, x_{t_1}, x_{t_2}) = p(x_{t_0}|x_{t_1})p(x_{t_1})p(x_{t_2}|x_{t_1}) \qquad \text{where} \quad t_0 \leq t_1 \leq t_2$$

since (in the graph) any path from x_{t_0} to x_{t_2} must pass through the node associated with x_{t_1}.

As exhibited by (4), we could specify the potential functions ψ in terms of conditional distributions. However, this choice is not unique—for example, we could also choose the "symmetric" form

$$p(x_0, y_0, \ldots, x_T, y_T) = \prod_{t=0}^{T} p(x_t, y_t) \prod_{t=1}^{T} \frac{p(x_t, x_{t-1})}{p(x_t)p(x_{t-1})}. \tag{5}$$

In fact, there are many equivalent ways to specify the potentials ψ (in the sense that both represent the same joint distribution p). Different choices may affect the interpretation of the intermediate quantities computed during inference (see, for example, Section 3.2.1), but do not alter the final results.

3 Inference in Graphical Models

It turns out that the structure of the graphical model $\mathcal{G}$ can be used to create efficient methods of performing inference over the collection of random variables. Typically, of course, we are primarily interested in computing an *estimate* of the values of any unobserved variables given the observations. Typical estimates include, for example, the maximum a posteriori (MAP) estimate, the maximum posterior marginal (MPM) estimates, and the least–squares (LS) estimates of x_s. From a Bayesian point of view we would like not only *estimates* of the x_s but also some measure of uncertainty on those estimates, i.e., we would prefer to have the *distribution* of likely values for x_s, rather than just the most likely one.

These goals fall generally into two tasks—maximization of the joint likelihood function (MAP), and marginalization (MPM). It turns out that both these operations may be performed quite efficiently in tree–structured graphical models, using quite similar algorithms. Before discussing the more general case, however, we consider the special case of the Markov chain of Section 2.3, for which optimal inference may be performed using the Kalman filtering, or Rauch–Tung–Striebel (RTS) smoothing, algorithms.

A third possible inference goal, which we do not explore in detail, is to obtain *samples* from the joint distribution given the observed values. In this

problem as well, the graph structure plays a similar role, determining the complexity of the procedure and enabling one to design both exact and approximate sampling methods which operate using the graph structure. These samples can then be used to provide Monte Carlo estimates of other quantities (for example, means, variances, and other expectations). For more details, see e.g., [49,50].

3.1 Kalman Filtering and Smoothing

In the system described by (3), it is well–known that optimal inference on the hidden variables x_t given a collection of past observations can be accomplished using a recursive estimation procedure called the Kalman filter [5]. Specifically, the Kalman filter recursively computes the quantities

$$\begin{array}{ll} \hat{x}_{t|t-1} = A\hat{x}_{t-1|t-1} & \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(y_t - C\hat{x}_{t|t-1}) \\ P_{t|t-1} = AP_{t-1|t-1}A^{\mathrm{T}} + Q & P_{t|t} = (I - K_tC)P_{t|t-1} \end{array} \tag{6}$$

where I is the identity matrix, A^{T} is the transpose of A, and K_t is the Kalman filter gain, given by

$$K_t = P_{t|t-1}C^{\mathrm{T}}(CP_{t|t-1}C^{\mathrm{T}} + R)^{-1} \tag{7}$$

taking initial conditions $P_{0|-1} = \Sigma_0$ and $\hat{x}_{0|-1} = \mu_0$. Then, $\hat{x}_{t|t}$ provides the best estimate of x_t given the observations up to time t under a wide variety of criteria (including the MAP, MPM, and LS criteria mentioned previously). An important point here is that the quantities $\hat{x}_{t|t}$ and $P_{t|t}$ are *sufficient statistics* of the past observations $\{y_0, \ldots, y_t\}$ for computing estimates or expectations over future states such as x_{t+1}. In fact, these statistics are simply the parameters of the posterior marginal, i.e.,

$$p(x_t|y_0, \ldots, y_t) = \mathcal{N}(x_t\,;\, \hat{x}_{t|t}, P_{t|t}) \tag{8}$$

where $\mathcal{N}(x\,;\,\mu, \Sigma)$ is a Gaussian distribution with mean μ and covariance Σ.

Efficient estimates of x_t given *all* observations (i.e., smoothing) can be accomplished using a two–pass version of the Kalman filter, typically called the Rauch–Tung–Striebel (RTS) smoother [6]. One way to understand this computation is that (assuming A is invertible) we may also run a second Kalman filter *backward* in time, so that

$$\begin{array}{ll} \hat{x}^-_{t|t+1} = A^{-1}\hat{x}^-_{t+1} & \hat{x}^-_{t|t} = \hat{x}^-_{t|t+1} + K^-_t(y_t - C\hat{x}^-_{t|t+1}) \\ P^-_{t|t+1} = A^{-1}P^-_{t+1|t+1}(A^{-1})^{\mathrm{T}} + Q & P^-_{t|t} = (I - K_tC)P^-_{t|t+1} \end{array} \tag{9}$$

with "initial" conditions $\hat{x}^-_{T|T+1} = 0$ and $(P^-_{T|T+1})^{-1} = 0$. Notice that (like the initial condition) the matrices $P^-_{t|t+1}$ produced by this recursion will not nec-

essarily be positive definite, and thus may not correspond to the covariance of a valid Gaussian density. In fact, the quantities $\hat{x}^-_{t|t+1}$ and $P^-_{t|t+1}$ are the parameters of the likelihood function $p(y_t, \ldots, y_T|x_t)$ (which also has a quadratic form similar to the Gaussian distribution, but is not necessarily normalizable). The best estimate of x_t and its posterior marginal are then given by

$$\begin{aligned} p(x_t|y_0, \ldots, y_T) &\propto p(x_t|y_0, \ldots, y_t)p(y_{t+1}, \ldots, y_T|x_t) \\ &\propto \mathcal{N}(x_t\,;\, \hat{x}_{t|t}, P_{t|t})\mathcal{N}(x_t\,;\, \hat{x}^-_{t|t+1}, P^-_{t|t+1}) \end{aligned}$$

i.e., a product of the quantities computed in each of the two passes.

The Kalman filter uses the (chain) structure of $\mathcal{G}$ to derive a method of inference which is efficient (linear) in the number of variables T (the length of the Markov chain). However, if the dimension d of the individual state variables x_t is high, each step of the Kalman filter may remain computationally difficult, since (6) involves the multiplication of matrices of size $d \times d$. Since the dimension of the variables in atmospheric and ocean modeling problems may be 10^6 or larger, the cubic cost of this multiplication may be intractable. In fact, for such high dimensions even the quadratic cost of representing the covariance matrices such as $P_{t|t}$ and $P^-_{t|t+1}$ may be impossible. In this case, the Markov chain structure does not in itself yield an efficient solution. It may, however, be possible to treat each dimension of x_t as an individual random variable to expose and exploit further independence structure in the graph (see Section 4.2). For the Gaussian case, the dependence structure encompassed in the covariance matrix can often be utilized to speed up the matrix inversion (e.g., [51,48]). In general, the issues of structure and dimensionality are closely related and together determine the complexity of inference (see Section 3.3).

As mentioned previously, the Kalman filter and RTS smoother can be thought of as simultaneously computing the maximum a posteriori (MAP) estimates of each x_t given observations $\{y_t\}$, *and* computing the posterior marginal distributions $p(x_t|y_1, \ldots, y_T)$. Although described for the Markov chain, the Kalman filtering algorithm generalizes relatively easily to jointly Gaussian distributions defined over any tree–structured graphical model. However, when the random variables are not jointly Gaussian, these inference goals are not in general equivalent, and thus are not both achieved using the same algorithm. However, each can be performed using algorithms which are similar in spirit to the Kalman filter. For example, for models with discrete–value unobserved variables in the Markov chain (often called a hidden Markov model [38]), the inference is similarly performed in a two–pass forward–backward manner [52].

The Kalman filter and RTS smoother can be viewed as special cases of the more general inference algorithms alluded to previously. It turns out that similarly efficient forms can be derived for any tree–structured graph, in essence taking advantage of the partial ordering of variables defined by the tree.

One useful way of framing these inference algorithms is as a set of iterative *message–passing* operations. In particular, each node t computes a message m_{ts} to send to its neighbor s, where m_{ts} is a function of the messages received by the other neighbors of t, $\Gamma_t \setminus s$. On a tree, this iterative procedure is guaranteed to converge in a finite number of steps. In fact, there exists an efficient order in which to compute these messages, so that each message is only computed once.

Specifically, beginning with the leaf nodes, we compute messages to parents (an upward sweep through the tree), with each node computing the outgoing message to its parent only after it has received messages from all its children. Then, beginning with the root node, we compute messages to children in a similar, downward sweep through the tree. This ordering can be thought of as a generalization of the forward and backward sweeps used in smoothing on a Markov chain (Section 3.1). We first describe how the operations of marginalization can be performed in this manner, then describe the modifications necessary to perform joint maximization.

3.2.1 Marginalization

The posterior marginal distributions of the variables at each node can be computed efficiently on a tree using a message–passing algorithm (sometimes called the *sum–product* algorithm), in which the outgoing message from t to s is computed using the incoming messages from t's other neighbors, as

$$m_{ts}(x_s) = \alpha_{ts} \int_{x_t} \psi(x_s, x_t)\psi(x_t) \prod_{u \in \Gamma_t \setminus s} m_{ut}(x_t) dx_t. \tag{10}$$

On a tree, the previously described ordering (upward from leaves to root, then downward from root to leaves) can be used to compute these messages in a well-posed manner. Each message m_{ts} is a function of the state variable x_s, defined on the same domain as x_s itself; it may sometimes be interpreted as a conditional distribution or a likelihood function, as illustrated by the example later in this section.

The posterior marginal at x_t is then given by the product of all incoming

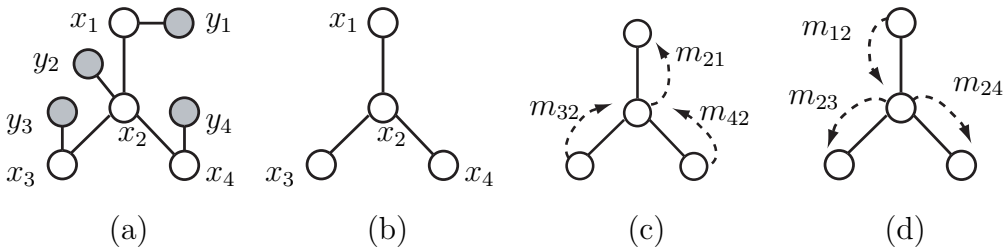


Fig. 4. (a) Given a tree–structured graphical model with observed variables $\{y_t\}$, or (b) an equivalent graphical model of only the hidden variables $\{x_t\}$, the messages required for inference can be computed efficiently using a two–pass algorithm. Taking x_1 to be the root node, (b) we compute upward messages m_{32} and m_{42}, then message m_{21}, followed by (c) a downward pass, computing m_{12} followed by m_{23} and m_{24}.

messages,

$$p(x_t|\{y_s\}) = \alpha_t \psi(x_t) \prod_{u \in \Gamma_t} m_{ut}(x_t). \tag{11}$$

Both definitions (10) and (11) contain a scalar proportionality constant α. Although the posterior (11) should be normalized so as to integrate to unity (making it a valid probability distribution), the constants α_{st} are essentially arbitrary, and chosen to avoid numerical underflow in the computations, often again by normalization. However, it is important to note that in the most general case, the individual messages m_{ts} may not even be finitely integrable (for example, the likelihood functions computed in the RTS smoother). For Gaussian uncertainty, the messages have a convenient closed form (an exponentiated quadratic), while in discrete-valued systems the messages are finite–length vectors and thus may always be normalized. In more general systems, one may use Monte Carlo techniques [50] to approximate the integrals involved. For Markov chains, examples include particle and ensemble filtering [53,54], and similar methods can also be applied to more general graph structures [55].

To see how this procedure works, let us consider the graph in Figure 4, and compute the posterior marginal distributions for each x_t. Although our original graph, shown in Figure 4(a), contains both hidden variables $\{x_t\}$ and observed variables $\{y_t\}$, we will first convert it to a simpler graph (Figure 4(b)) of only the hidden variables. In this new graph, each potential function $\psi_t(x_t)$ is an implicit function of the observation y_t (whose value is known). Specifically, let us choose to parameterize the potential functions as conditional distributions, so that the root has a single–node potential given by $\psi(x_1) = p(x_1, y_1)$, the pairwise potentials are conditionals, e.g., $\psi(x_1, x_2) = p(x_2|x_1)$, and the remaining single–node potentials are the observation likelihoods, e.g., $p(y_2|x_2)$.

By Bayes' rule, the posterior marginal of x_1 is

$$p(x_1|\{y_t\}) = p(x_1, \{y_t\})/p(\{y_t\}).$$

The denominator $p(\{y_t\})$ is simply a constant (the data likelihood); then, using the law of total probability and the factorization implied by the graph structure we have

$$\propto \iiint p(x_1, y_1)p(x_2|x_1)p(y_2|x_2)p(x_3|x_2)p(x_4|x_2)p(y_4|x_4)p(y_3|x_3)\, dx_2 dx_3 dx_4.$$

By the distributive law, this is

$$= p(x_1, y_1) \int p(x_2|x_1)p(y_2|x_2) \left(\int p(x_3|x_2)p(y_3|x_3)dx_3 \right) \left(\int p(x_4|x_2)p(y_4|x_4)dx_4 \right) dx_2$$

and applying the definition of the sum–product messages we have

$$\begin{aligned} &\propto p(x_1, y_1) \int p(x_2|x_1)\, p(y_2|x_2)\, m_{32}(x_2)\, m_{42}(x_2) dx_2 \\ &\propto p(x_1, y_1)\, m_{21}(x_1), \end{aligned}$$

which matches (11). From this it is relatively easy to see that, like the backward pass of the Kalman filter example in Section 3.1, the upward messages in this graphical model are likelihood functions, e.g.,

$$m_{32}(x_2) \propto p(y_3|x_2) \qquad\qquad m_{21}(x_1) \propto p(y_2, y_3, y_4|x_1);$$

one can show similarly that the downward messages are conditionals, e.g.,

$$m_{12}(x_2) \propto p(x_2|y_1) \qquad\qquad m_{23}(x_3) \propto p(x_3|y_1, y_2, y_4)$$

Thus the messages $m_{ts}(x_s)$ provide sufficient statistics for each marginal computation in the graph. Other choices of ψ, such as the symmetric potentials (5), result in different interpretations of the messages m_{ts}, but still provide sufficient statistics using precisely the same computational steps [56].

Higher–order posterior marginal distributions may also be estimated using these sufficient statistics; for example, the posterior over two nodes s and t with $(s, t) \in \mathcal{E}$ is given by

$$p(x_s, x_t|\{y_u\}) \propto \psi(x_s)\psi(x_t)\psi(x_s, x_t) \prod_{u \in \Gamma_s \setminus t} m_{us}(x_s) \prod_{u' \in \Gamma_t \setminus s} m_{u't}(x_t). \qquad (12)$$

The resulting marginal distributions can be used not only to create estimates of the x_t, but for other inference goals as well; for example, to compute expectations over functions of the variables.

Finally, note that the same algorithm (specifically, the upward pass) can be used to compute the likelihood of the observed data $\{y_t\}$. In particular, recall that the α_{ts} are arbitrarily chosen (typically for numerical convenience) and that α_t are chosen so as to normalize the posterior distribution. If we enforce

the choice $\alpha_{ts} = 1$ in our example, we find that the product on the right–hand side of (11) is the joint $p(x_t, \{y_s\})$, and thus the normalizing constant α_t is precisely the likelihood $p(\{y_s\})$. The same likelihood computation can, of course, be accomplished by simply keeping track of the α_{ts} at each stage, a process which (like many likelihood calculations) is often performed in the log–domain for numerical reasons.

3.2.2 Maximization

The principle of the distributive law which underlies the sum–product algorithm for marginalization can be extended to a general set of computations which can be made efficient on a tree [57]. Another inference goal in this category which is frequently of interest is to find a joint state $\hat{x}_{\mathcal{V}}$ which has maximum posterior probability (the MAP estimate).

By replacing the integrals of (10) and (11) by a maximum over the space of x_t, we obtain an alternate procedure, called the *max–product* algorithm [30,58]. Instead of the posterior marginal distributions, this algorithm computes the so–called "max–marginals" $\rho(x_t)$ over each variable x_t, given by

$$\rho(x_t) \propto \max_{x'_{\mathcal{V}\setminus t}} p(x_t, x'_{\mathcal{V}\setminus t}) \tag{13}$$

and pairwise max–marginals $\rho(x_s, x_t)$ defined similarly. (Note that the max–product operations can equivalently be written in terms of log–probabilities, in which case one obtains a "min–sum" algorithm.)

Using these quantities, it is a simple matter to find a joint MAP estimate for each x_t. To do so, we first compute the max–marginal at the root node of the tree, which we denote x_0 without loss of generality, by calculating the upward messages of the max–product algorithm. Let $\hat{x}_0$ be any value which maximizes $\rho(x_0)$. Now, proceeding downward through the tree, we compute a MAP estimate of x_t given the estimate of its parent x_s by choosing any $\hat{x}_t$ which maximizes $\rho(x_t, \hat{x}_s)$.

Once again we see that in a tree–structured graph, exact inference can be performed efficiently in two sweeps of a simple message–passing algorithm. The max–product algorithm is in fact an example of dynamic programming [59], and when the graph structure is a simple chain, becomes equivalent to the well–known Viterbi algorithm [60].

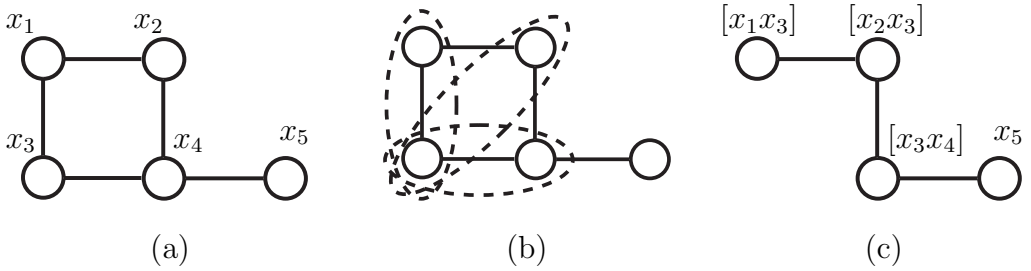


Fig. 5. Removing cycles from a graph. (a) Given a graph with cycles, one may (b) group variables together to form (c) a new graph with no cycles, but higher–dimensional variables associated with some nodes.

3.3 Inference Over Arbitrary Graph Structures

When the graphical model is *not* tree–structured, inference becomes somewhat more difficult. In particular, the presence of cycles means that the messages in Section 3.2 are no longer sufficient statistics: for any two variables x_s, x_u within a cycle, no single other variable x_t provides graph separation. One way to perform inference in graphs with cycles is to first convert them into a tree–structured graph via a process of variable augmentation, and then perform inference on the new graph using the methods described previously.

To see how graph augmentation can be used to remove a cycle, consider Figure 5(a). In this example, we have a graph with five random variables, four of which form a single cycle. If we define new random variables by concatenating x_3 with each of the other variables in the loop (Figure 5(b)), the independence structure of the new variables allows us to obtain the graph in Figure 5(c), which has no loops. In the original graph, x_2 and x_3 together were sufficient to separate x_1 and x_4, and this is captured in the new graph by treating x_2 and x_3 jointly. By successively applying such a procedure, one may convert any graph with cycles into a tree–structured graphical model [29].

However, this procedure comes at a computational price. Specifically, we have increased the dimensionality of several of the random variables in our graph. This means that the representation size of the potential functions and messages in the new graphical model are larger, increasing the storage and computational costs accordingly. In jointly Gaussian models, this increase is modest (since a d–dimensional covariance matrix is specified by $d^2/2$ entries) but can be unacceptable at sufficiently high dimensions; for discrete–valued random variables, however, the increase is exponential (since the concatenation of d binary random variables has 2^d possible values) and can very quickly become intractable.

Graphical models can also be used to define reasonable approximate alternatives when exact inference is deemed too computationally expensive. One of the most popular methods, "loopy" belief propagation (or loopy sum–

product), involves simply applying an iterative version of the sum–product equations (computing a message m_{ts}^i at iteration i in terms of its neighbor's messages m_{ut}^{i-1} at the previous iteration) despite the presence of loops in the graph [30]. Although this procedure does not necessarily converge to the correct estimates, and may in fact not converge at all, it has been shown to have very good empirical performance in many applications [57,61], and a number of methods with generally similar justifications have been developed [62–64]. Typically these algorithms perform well for graphs which are "close" to trees, i.e., are sparse and have relatively long cycles.

3.4 Learning Graphical Models

We have thus far assumed that both the graphical model's structure and its potential functions have been completely specified; but in many practical problems this is not necessarily the case. Often, either the functions ψ or both ψ and the graph structure $\mathcal{E}$ are unknown *a priori*, and must be learned from data.

In general, either learning task can be a formidable one. Specifically, the optimal choice of which edges to include in $\mathcal{G}$ given, say, a constraint on the complexity of inference in the resulting graph is NP-hard [65]. Given the structure of the graph, learning the model parameters (as specified by the potential functions ψ) remains a challenging problem. In general one resorts to local, likelihood–based search methods such as iterative proportional fitting [66], in which one performs a type of coordinate ascent by repeatedly optimizing each potential function to match the observed data given all the other potential functions.

However, once again the special case of tree–structured graphical models admits tractable solutions to both problems. Indeed, in any tree the maximum likelihood choice of the potential functions ψ have a simple closed form solution, which can be written in terms of the observed distributions over pairs of variables (either the conditional distributions as in (4) or in terms of the pairwise and single–variable marginals as in (5)). Because of this, and the ease of calculating likelihoods on a tree, it is also straightforward to select the optimal set of edges $\mathcal{E}$ such that $\mathcal{G}$ forms a tree. One may do so by following a simple maximum–weight spanning tree algorithm, where the weights are given by functions of the observed distributions over pairs of variables [67], as will be illustrated in Section 4.2.

Occasionally, even these pairwise distributions are unavailable (as, for example, in a hidden Markov model where the hidden state variables x_t are *never* observed), or the potentials ψ may be constrained to some limited parametric

form. In these cases we can again perform likelihood based optimization of the parameters of ψ. The expectation–maximization (EM) algorithm [68] is one popular means of performing this optimization. EM is an iterative algorithm which, in each step, first holds the parameters of ψ fixed and uses inference to compute the posterior of the hidden variables, $p(\{x_t\}|\{y_t\})$, then holds this distribution fixed while maximizing the likelihood of the y_t over the parameters of ψ. This process can be quite efficient, since the maximization step frequently has a simple, closed form solution. By repeating this procedure, one is guaranteed to find parameters of ψ which correspond to a local maximum of the likelihood under very mild conditions.

If we take a more Bayesian approach to parameter estimation, we may wish to determine not only the maximum likelihood setting of the parameters of ψ but also a *distribution* over these parameters (which may be more useful than a simple point–estimate). In this case, the parameters themselves can be included in the graphical model. We can then perform inference about quantities of interest, including both parameters and state variables, by applying the same algorithms to this graph. Parameter estimation, for example, becomes a computation on this graph: we can marginalize over the hidden state variables to obtain either point estimates of the parameters (conditional means or modes), or in a more Bayesian fashion compute posterior distributions over the parameters. Since this estimation process is the same as performing inference on a graph (a graph with nodes for both parameters and variables), the same principles apply in terms of computational schemes: graphs without cycles are straightforward and graphs with cycles become more complex, with sparsity again playing a role. This can be regarded as a generalization of the "augmented state vector" approach to parameter estimation, in which the unknown parameters of the model dynamics or system noise are added to the state vector in sequential estimation (and are typically assumed to evolve in time in a very simple way, e.g., by staying constant in time [69,24]).

4 Illustrative Examples of Graphical Models

In this section, we provide two illustrative examples of how graphical models have already been applied to atmospheric and climatological modeling. The first example describes how multi-scale graphical models can be used to capture the interactions among very large numbers of Gaussian random variables, such as might arise in satellite imagery. In the second example, we consider how graphical models can be used to model spatial and temporal patterns of rainfall observed at a collection of stations. In both cases, we shall see that tree–like graphical models can be used to both represent statistical dependency among the variables and perform the required inferences (estimation, etc.) efficiently. Note that both examples below are intended to highlight how

the concepts described earlier in the paper can be applied to observational geoscience data—for more complete details on multi-scale graphical models and their applications to geoscience data readers can consult [70], and [71,72] for details on spatio-temporal graphical models applied to precipitation data.

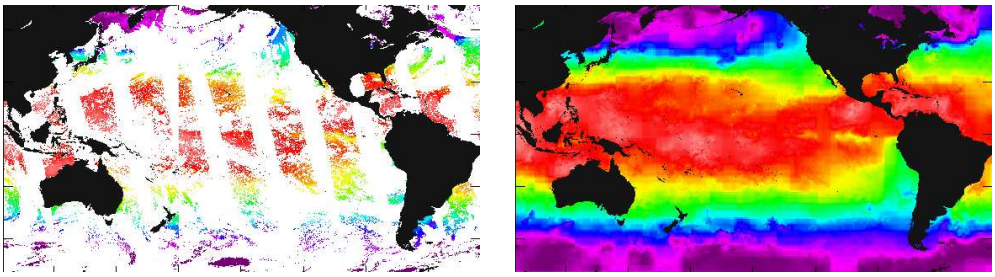
4.1 Multi-Scale Tree–Structured Models

Multi-scale or multi-resolution models and algorithms have been used successfully in numerous application areas, including satellite [73], tomographic [74], and natural image processing [75]. Graphical models provide one means of describing the statistical dependence relationships in multi-scale models which enable efficient representation and computation even for high–dimensional data. Here, we describe one example of how tree–structured graphical models can be used to solve large–scale data interpolation problems.

Figure 6(a) shows an image of the sea–surface temperature measurements recorded on a single day by the NASA MODIS/Aqua satellite [76]. Suppose that we wish to recover an estimate of the temperature, at some reasonable granularity (for example, the image in Figure 6(a) involves some 10^5 scalar variables, one for each pixel). However, the available observation data are extremely sparse (approximately 4000 observations total); the rest are completely unobserved due to the limited satellite path or the presence of cloud cover. We wish to interpolate these few observations into an estimate of the complete surface temperature (Figure 6(b)), and we may also wish to be informed of our estimate's reliability.

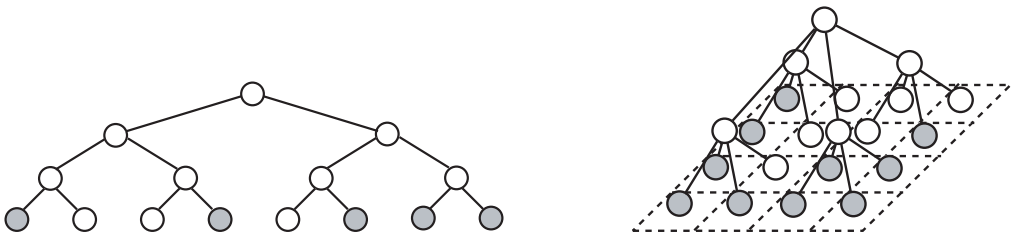
Of course, there exist many possible methods for interpolating the observed data [77]; but the method on which we shall focus relies on treating the temperature as a (large) collection of jointly Gaussian random variables, and applying a multi-scale model to lend structure to the covariance matrix and inform us of how to perform optimal estimation efficiently within this model. Note first, however, that the full, joint covariance matrix has $\approx 10^{10}$ entries, and is therefore too large to represent explicitly, much less manipulate. By using a graphical model, we can enforce a form of *sparseness* which gives both an efficient representation and enables tractable inference algorithms.

We assume that the process which generated the temperature data in Figure 6(a) can be described using a simple multi-scale model, specifically (since the data is two–dimensional) a quad-tree structure as shown in Figure 7(b). This type of multi-scale model (see Figure 7) has been successfully applied to the very similar problem of estimating sea–surface height using sparse satellite imagery [73]. We associate the finest scale of the tree structure with the original variables of interest (i.e., temperature), while the higher levels of the tree



(a) (b)

Fig. 6. Estimating sea–surface temperature. (a) Given only sparse satellite observations, we would like to produce both (b) an interpolated image of temperature estimates at each point.



(a) (b)

Fig. 7. Tree–structured multi-scale models. (a) A binary tree multi-scale model of a one dimensional process $x_1 \ldots x_8$, some of which have been observed (shaded nodes); (b) a quad-tree model of a two dimensional process (such as an image), again with partial observations.

are taken to be new variables representing the "state" of a broader (larger) region which decouple the state of their children. For example, the root node might be thought of as representing a kind of average global temperature, its children as capturing deviations from this mean in each of the four quadrants, and so forth at increasingly finer scales. The graph structure then represents our belief that the child nodes are statistically independent given the parent's (higher–scale) state, and the potential functions ψ define a (scale–dependent) amount of process noise. By choosing this noise level to increase geometrically with scale, the process takes on a $1/f$-like fractal self–similarity, similar to the self–similarity observed in many real–world applications [70].

Specifically, for each node u and its parent node t, we take $\psi_{ut} = \mathcal{N}(x_u - x_t\,;\, 0, \sigma^2 \cdot 2^{2\,\mathrm{sc}(u)})$, where $\mathrm{sc}(u)$ is the scale of node u—zero for the finest-scale, leaf nodes and increasing by one at each coarser scale—and $\sigma \approx .13$ for state measured in degrees Celsius. The observations themselves are assumed for this example to be noise–free, producing only an interpolation effect rather than smoothing. However, a more sophisticated model with observation noise is equally feasible; this observation noise could be learned from the data or set to the measured accuracy level of the sensing instrument (in this case, about

.3° Celsius).

Given the tree-structured graphical model, it is then easy to perform optimal estimation on the hidden variables (e.g., to compute MAP estimates of the unobserved leaf (pixel) nodes in the model) using the two-pass inference process described in Section 3.2. The computational cost of this algorithm is linear in the number of variables to be estimated, versus (for example) the quadratic or even cubic cost of directly manipulating the joint covariance matrix. These MAP estimates can be used as an interpolated estimate of the temperature, shown in Figure 6(b). Moreover, in a tree-structured model, it is equally easy (linear in the number of variables) to also obtain estimates of our uncertainty in these values (such as their marginal variance).

One issue that frequently arises for such multi-scale Markov trees is the presence of block artifacts along the edges of the quad-tree. We typically believe that a process such as temperature should be spatially smooth, but our model may not be—for instance, there are nodes which are spatially close, but are far from one another in the tree, indicating that they will have relatively low correlation (for example x_4 and x_5 in Figure 7(a)). A number of methods can be used to fix this. For the images in Figure 6, we used the computationally expedient technique of simply averaging the results of several such trees, each with centers slightly shifted from the others [70]. Other, more sophisticated approaches include creating a hidden mixture model of several such shifted trees and performing inference over the mixture (producing a weighted average rather than a simple average); using a single tree with overlapping leaf regions [78]; or introducing one or more extra edges (thus creating cycles in the graph) [79].

Finally, imposing a simple statistical structure also makes it feasible to estimate the parameters of the model [80,70]. First, the statistical structure imposed greatly reduces the number of parameters necessary for the model, and additional prior knowledge can be brought to bear on how the parameters are themselves interrelated. Secondly, the tree-structured nature of the graph makes it easy to compute likelihoods and expectations (a necessary step in any likelihood-based parameter estimation) using the same techniques of Section 3.

4.2 Hidden Markov Models for Precipitation Modeling

Our second example application illustrates how graphical models can be used to model precipitation for networks of rain stations. Accurate representation of both spatial and temporal dependencies in rainfall across a region can be important for many hydrological applications. For example, simulations of rainfall patterns can be used in conjunction with hydrologic models to estimate

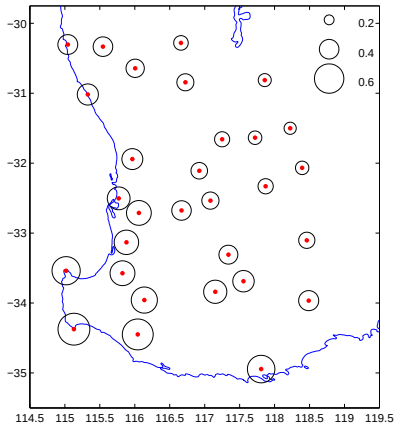


Fig. 8. Locations for stations in Southwestern Australia and their marginal probabilities of rainfall (> 0.3mm) as indicated by the circle radii.

flood risk under current or future conditions. We can view both the creation of a statistical model of rainfall and its initialization to current conditions as an assimilation of historical precipitation data (and we shall also describe how one may include other observed or simulated atmospheric measurements such as sea surface temperature or wind vectors). We focus our discussion on the models for precipitation *occurrence*, a multivariate binary process with 0 corresponding to the value of "dry" and 1 corresponding to "wet". [5] For this discussion, we assume that the precipitation data is collected at regular intervals (daily) for a period of T days for a given set of d stations. We will denote a d–dimensional binary vector of precipitation occurrence at day t by $y_t = [y_t^1 \dots y_t^d]$, with each dimension y_t^i corresponding to the observation at station i. By building a model of the time series $\{y_t\}$, we can predict or simulate realistic rainfall patterns in the future.

4.2.1 Example: Precipitation Data from Southwestern Australia

We use a data set collected at $d = 30$ stations in Southwestern Australia stations over the winter months of May–October, 1978–1992. [6] Figure 8 shows the geographic locations of the stations and their average rainfall occurrence probabilities.

Broadly speaking, any good model for rainfall patterns must capture both spatial correlations between the individual stations at time t, and temporal correlations between time t and $t + 1$. Due to the high dimensionality of the observations (each y_t can take on some 2^d possible values), the distributions involved are difficult to represent or manipulate directly. For example, modeling

[5] Many models for rainfall *amounts* also use occurrence models as a first step.

[6] This data set was first studied in [81] and provided to us courtesy of Dr. Stephen P. Charles of CSIRO, Australia.

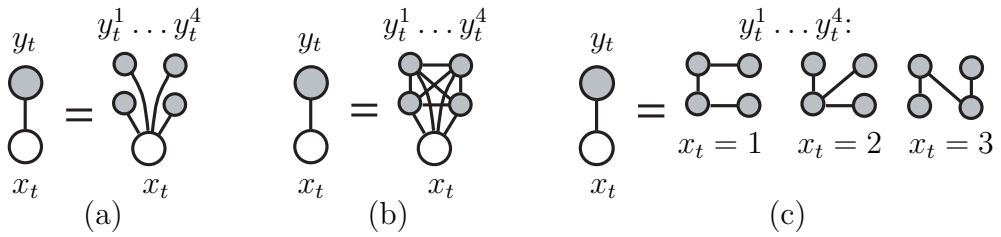


Fig. 9. Graphical models for describing rain station dependencies. (a) conditional independence, $\prod p(y_t^i|x_t)$ [83]; (b) all pairwise dependencies, so that $p(y_t|x_t)$ is fully connected [84,81]; (c) conditionally tree–structured, so that $p(y_t|x_t)$ is tree–structured, but that structure depends on the value of x_t.

of $p(y_t|y_{t-1})$ directly as a Markov chain would involve order of 2^{2d} parameters. As before, by structuring such distributions using a graphical model we can considerably reduce the complexity of such models and make the associated computations tractable.

4.2.2 Modeling Temporal Dependencies

Let us first consider the temporal structure of the data. One way we can create an efficient representation is to postulate the existence of a relatively low–dimensional hidden state x_t which captures the dependence of y_t on its past [82,81,83]. In particular, we can use a hidden Markov model (Figure 1) in which the dimensionality of x_t is much smaller than that of y_t. This bypasses the aforementioned problem of modeling $p(y_t|y_{t-1})$ directly and instead seeks to capture temporal dependence via a lower-dimensional Markov chain defined on the x_t state variable. The hidden variable x_t can be viewed as a "weather state" representing general, temporally–localized conditions about the present. This state might capture, for example, whether the present time is likely to be very wet or very dry, or even indicate the likelihood of particular spatial patterns.

Introducing the hidden state x_t allows us to decompose the problem into several, more manageable parts: (a) determine the temporal state transition distribution $p(x_t|x_{t-1})$; (b) determine the graph structure, or factorization of $p(y_t|x_t) = p\left(y_t^1, \ldots, y_t^d|x_t\right)$ describing the spatial distribution of rainfall at a particular time t and conditioned on a particular state x_t, and (c) learn the parameters, ψ, of the distributions in parts (a) and (b) from observed data.

4.2.3 Modeling Spatial Dependencies

We next turn to modeling $p\left(y_t^1, \ldots, y_t^d|x_t\right)$, the spatial distribution of rainfall patterns across stations, given the hidden state x_t. Perhaps the simplest possible graphical structure for the y_t^i is to make them conditionally independent given the state x_t [83]. This type of model is depicted in Figure 9(a). However, this model may be *too* simplistic as x_t can capture only *some* of the

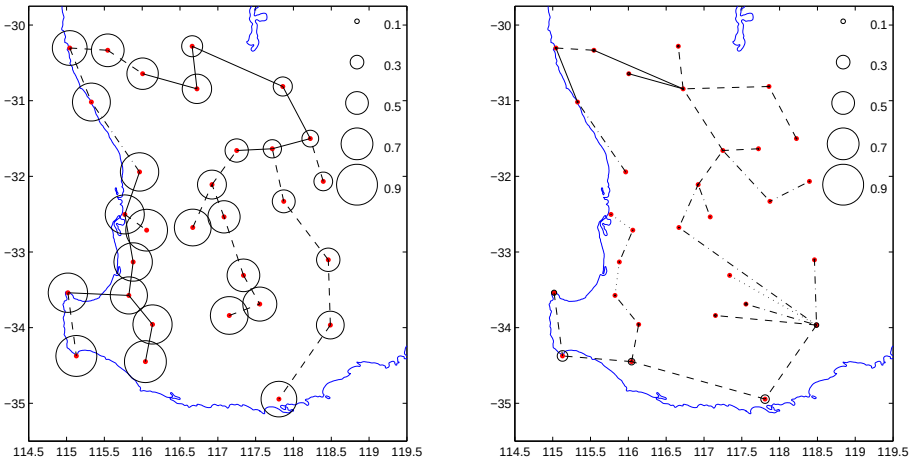
dependence between pairs of stations (y_t^i and y_t^j). Hughes and Guttorp [84,81] suggest modeling *all* of pairwise dependencies via an auto-logistic model with graph structure described in Figure 9(b). However, due to the dense nature of the dependence, exact inference and parameter estimation for this model are infeasible, and approximations are cumbersome and slow. An alternative approach is to learn a model with a sparse dependence structure for $p\left(y_t|x_t\right)$. For example, a tree structure is appealing since we can both perform inference and estimate the parameters efficiently. While it seems unlikely that the true dependency among the y_t^i will be tree-structured, we could select the best, single tree-structured model for their dependence [67]. However, we would likely observe artifacts similar to those described in Section 4.1, where the correlation between stations which are close spatially but not directly connected in the tree would be modeled poorly. This can be avoided by making the structure of y_t dependent on the value of the state x_t. Specifically, conditioned on a state x_t, the stations y_t^i are tree-structured, but the structure may be quite different for different values of x_t [85]. This type of structure is shown in Figure 9(c).

4.2.4 Examples of Models Learned Using the Australia Data

As briefly discussed in Section 3.4 we can use the EM algorithm to find maximum likelihood estimates of the parameters of a graphical model with known structure involving a hidden variable x_t. Complete details on using EM to learn parameters for each of the conditional–independence and tree–structured HMMs (the models described above) can be found in [86].

The cardinality of the hidden state x_t can be selected using different data-driven approaches—for the Southwest Australian data we maximized the out–of–sample log–likelihood under cross–validation [87]. More specifically, for each possible cardinality and each year of data, we withhold that year and use EM to train a model using the remaining years, and then evaluate the performance of each model by computing the log-likelihood of the withheld data. Finally, we select the cardinality which results in the best average cross–validated log–likelihood. In our example, using a HMM with a tree–structured model for $p(y_t|x_t)$, this results in five hidden states (i.e., x_t takes on one of five discrete values).

The tree distributions corresponding to $p(y_t|x_t)$, for two particular values of x_t, are shown in Figure 10. The first ($x_t = 2$) is a relatively wet state, characterized by rainfall probabilities exceeding 0.7 along the west coast, with probabilities below 0.5 inland. The tree structure indicates north–south dependencies between stations, rather than east–west, with the strong dependencies between stations in the southwest. This is consistent with the meteorology and topography of the region [81]. Winter rainfall over southwestern Australia is large-scale and frontal, impacting the southwest corner of the domain first and foremost.



(a) $p(y_t|x_t = 2)$ (b) $p(y_t|x_t = 5)$

Fig. 10. Visual interpretation of two of the hidden states for a 5-state HMM trained on Southwestern Australia data. Circle radii indicate the precipitation probability for each station given the state. Lines between the stations indicate the edges in the dependence tree. Patterns of lines correspond to different strengths of mutual information, a measure of pairwise dependence: solid (strong), dashed (moderate), dash-dotted (fair), and dotted (weak).

Hence, the tendency for correlations between stations along the coast during moderately wet weather states. The second state ($x_t = 5$) characterizes dry conditions throughout the domain, and spatial dependencies between stations are generally weaker. This is consistent with the lack of organized large scale rain- bearing systems; any rainfall is local and very sporadic. Notice also that in both cases, the tree structures are consistent with the topography of the region, i.e., the vast majority of edges connect neighboring stations, even though the model uses no geographical information about the stations.

4.2.5 Extensions

An advantage of the type of graphical model described above is that it can also be used to perform inference about missing observations. Since the state–dependent conditional distributions $p(y_t|x_t)$ are tree–structured, they can be estimated even in the presence of missing data with very little additional effort by marginalizing out any missing elements of y_t. This feature can be quite useful, since although the example shown used a complete data set, historical precipitation data sets for many geographic regions can have substantial numbers of missing observations [88].

In addition to decoupling spatial and temporal dependencies, hidden states can also be used to efficiently incorporate other variables into the model. Since precipitation is a result of an atmospheric process, other atmospheric measurements can be a useful addition to the model. Assuming that other

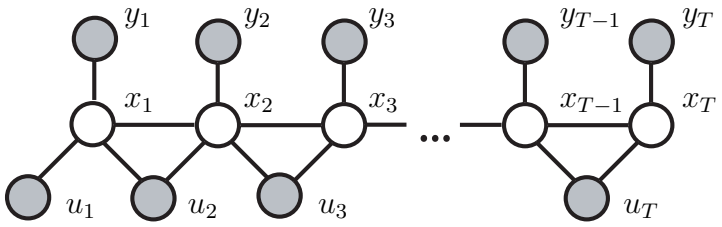


Fig. 11. Graphical model for a nonhomogeneous hidden Markov model (NHMM)—the additional atmospheric variables u_t lend information about the state by influencing the dynamics $p(x_t|x_{t-1}, u_t)$.

atmospheric variables are also measured (or simulated) at a daily scale for the same days as y, we will denote by u_t a vector of atmospheric variables for day t, and use a graphical model to represent $p(y_1, \ldots, y_T|u_1, \ldots, u_T)$. One way to incorporate the information in u_t is to make the transition distribution between the hidden states $p(x_t|x_{t-1})$ to be dependent on the values of u_t: $p(x_t|x_{t-1}, u_t)$ [82]. The resulting transition probabilities are no longer stationary, and thus the resulting HMM is sometimes referred to as a *nonhomogeneous* HMM, or NHMM. Its graphical model is shown in Figure 11. When provided relevant atmospheric variables, an NHMM model was found to produce more realistic simulated sequences than a corresponding HMM [82], and can also be applied to assimilate the information in atmospheric variables useful in modeling inter-annual variability [71,72].

5 Summary and Discussion

Graphical models provide a systematic framework for inference on sets of random variables. The key idea in graphical modeling is the equivalence between conditional independence among random variables and separation of nodes in a corresponding graph. In particular, highly structured, sparse graphs often lead to efficient computational inference methods. Well known computational schemes such as the Kalman filter recursions can be viewed as special cases of a more general framework for inference in graphical models.

In a data assimilation context the random variables in the graph typically include data nodes (whether observed or missing), unobserved state nodes, and possibly even nodes corresponding to parameters (whose values may be known or may need to be estimated). Questions about nodes in the graph whose values are not known with certainty (unobserved variables) can be answered in a systematic way by applying the message-passing algorithms discussed earlier. The generality of this approach lies in the fact that the inference computations are applicable equally well to nodes involving data (e.g., missing observations), state variables (unobserved states), parameters (unknown parameter values), or any combination of these.

As pointed out earlier, the framework of graphical models is not a cure–all for problems in data assimilation. For example, in a Gauss–Markov state–space model in which the state is a d-dimensional variable, the local message passing computations are $\mathcal{O}(d^3)$ due to the matrix computations required in updating the error covariance at each step. Tree–structured graphs are efficient in the sense that only a linear number of such computations (in the number of nodes of the graph) are required. However, for the large d values that may be of interest in atmospheric and oceanic modeling problems (e.g., $d \approx 10^6$ or 10^7), even a single such computation is not computationally feasible. In such systems, additional factorization (graph) structure may be imposed on the model (as in Sections 4.1–4.2), and other methods of approximation, such as low–rank, Monte Carlo, or ensemble approximations may be applied to reduce computation.

Acknowledgements

This paper is dedicated to the intellectual contributions to the field of Judea Pearl and to the memory of his son, Daniel Pearl, who died for the freedom of information and for better understanding between people and religions. It is based on an invitation to lecture on the topic at UCLA, invitation issued by the editors of this Special Issue to PS. Given the lifelong affiliation of Judea Pearl with UCLA, the authors feel it to be particularly appropriate to pay this tribute to his ideas.

The material in this paper is based on work supported in part by the National Science Foundation under grants SCI-0225642, IIS-0431085, and ATM-0530926 (AI, SK, and PS) and ATM-0082131 (MG), and by Department of Energy Grant DEFG02-02ER63413 (AR, MG, and PS).

References

[1] E. S. Epstein, Stochastic dynamic prediction, Tellus 21 (6) (1969) 739–759.

[2] T. Bengtsson, C. Snyder, D. Nychka, A nonlinear filter that extends to high dimensional systems, J. of Geophys. Res.–Atmos. 108 (D24).

[3] L. M. Berliner, Physical–statistical modeling in geophysics, J. of Geophys. Res.–Atmos. 108 (D24).

[4] A. C. Lorenc, The potential of the ensemble Kalman filter for NWP—a comparison with 4d-var, Q. J. Royal Meteor. Soc. 129 (2003) 3183–3203.

[5] R. E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME, Ser. D: J. Basis Engineering 82 (1960) 35–45.

[6] A. Gelb (Ed.), Applied Optimal Estimation, MIT Press, Cambridge, 1974.

[7] R. S. Bucy, P. D. Joseph, Filtering for Stochastic Processes with Applications to Guidance, 2nd Edition, Chelsea, 1987.

[8] E. N. Lorenz, The mechanics of vacillation, J. Atmos. Sci. 20 (1963) 448–464.

[9] G. Veronis, An analysis of wind-driven ocean circulation with a limited number of fourier components, J. Atmos. Sci. 20 (1963) 577–593.

[10] M. Ghil, S. Childress, Topics in Geophysical Fluid Dynamics: Atmospheric Dynamics, Dynamo Theory and Climate Dynamics, Springer–Verlag, New York/Berlin/London/Paris/Tokyo, 1987.

[11] H. W. Sorenson (Ed.), Kalman Filtering: Theory and Application, IEEE Press, 1985.

[12] A. H. Jazwinski (Ed.), Stochastic Processes and Filtering Theory, Academic Press, New York, 1970.

[13] R. N. Miller, M. Ghil, F. Gauthiez, Advanced data assimilation in strongly nonlinear dynamical systems, Journal of the Atmospheric Sciences 51 (8) (1994) 1037–1056.

[14] K. Ide, M. Ghil, Extended Kalman filtering for vortex systems. Part I: Methodology and point vortices, Dyn. of Atmos. and Oceans 27 (1-4) (1998) 301–332.

[15] K. Ide, M. Ghil, Extended Kalman filtering for vortex systems. Part II: Rankine vortices and observing-system design, Dyn. of Atmos. and Oceans 27 (1-4) (1998) 333–350.

[16] G. Evensen, Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics, J. of Geophys. Res 99 (C5) (1994) 10143–10162.

[17] H. Moradkhani, K.-L. Hsu, H. Gupta, S. Sorooshian, Uncertainty assessment of hydrologic model states and parameters: sequential data assimilation using the particle filter, Water Res. Research 41 (5).

[18] R. N. Miller, E. F. Carter, S. T. Blue, Data assimilation into nonlinear stochastic models, Tellus Series A – Dynamic Meteorology and Oceanography 51 (2) (1999) 167–194.

[19] I. Fukumori, Assimilation of TOPEX sea level measurements with a reduced-gravity, shallow water model of the tropical Pacific Ocean, J. of Geophys. Res.–Oceans 100 (C12) (1995) 25027–25039.

[20] I. Fukumori, A partitioned Kalman filter and smoother, Monthly Weather Review 130 (5) (2002) 1370–1383.

[21] P. M. Lyster, S. E. Cohn, R. Menard, L.-P. Chang, S.-J. Lin, R. Olsen, Parallel implementation of a Kalman filter for constituent data assimilation, Mon. Weather Rev. 125 (2) (1997) 1674–1686.

[22] C. L. Keppenne, Data assimilation into a primitive-equation model with a parallel ensemble Kalman filter, Monthly Weather Review 128 (6) (2000) 1971–1981.

[23] M. Ghil, P. Malanotte-Rizzoli, Data assimilation in meteorology and oceanography, in: B. Saltzmann (Ed.), Adv. in Geophys., Vol. 33, Academic Press, 1991, pp. 141–266.

[24] M. Ghil, K. Ide, A. F. Bennett, P. Courtier, M. Kimoto, N. Sato (Eds.), Data Assimilation in Meteorology and Oceanography: Theory and Practice, Meteorological Society of Japan and Universal Academy Press, Tokyo, 1997.

[25] Panel on Model–Assimilated Data Sets (D.R. Johnson, J.R. Bates, G.P. Brasseur, M. Ghil, A. Hollingsworth, R.L. Jenne, K. Miyakoda, E. Rasmusson, E.S. Sarachik, and T.T. Warner), National Research Council, Four-Dimensional Model Assimilation of Data: A Strategy for the Earth System Sciences, National Academy Press, Washington, D.C., 1991.

[26] J. Kao, D. Flicker, R. Henninger, S. Frey, M. Ghil, K. Ide, Data assimilation with an extended Kalman filter for impact-produced shock-wave dynamics, J. Comp. Phys. 196 (2) (2004) 705–723.

[27] D. B. Stephenson, C. A. S. Coelho, F. J. Doblas-Reyes, M. Balmaseda, Forecast assimilation: a unifed framework for the combination of multi–model weather and climate predictions, Tellus A 57 (3) (2005) 253–264.

[28] M. C. Thomson, F. J. Doblas-Reyes, S. J. Mason, R. Hagedorn, S. J. Connor, T. Phindela, A. P. Morse, T. N. Palmer, Multi-model ensemble seasonal climate forecasts for malaria early warning, Nature, submitted.

[29] S. Lauritzen, D. Spiegelhalter, Local computations with probabilities on graphical structures and their applications to expert systems, J. R. Stat. Soc. B 50 (1988) 157–224.

[30] J. Pearl, Probabilistic Reasoning in Intelligent Systems, Morgan Kaufman, San Mateo, 1988.

[31] J. Whittaker, Graphical Models in Applied Multivariate Statistics, John Wiley & Sons, New York, 1990.

[32] M. I. Jordan (Ed.), Learning in Graphical Models, MIT Press, Cambridge, 1998.

[33] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, D. J. Spiegelhalter, Probabilistic Networks and Expert Systems, Springer-Verlag, New York, 1999.

[34] M. I. Jordan, Graphical models, Statistical Science 19 (2004) 140–155.

[35] S. L. Lauritzen, N. A. Sheehan, Graphical models for genetic analyses, Statistical Science 18 (2003) 489–514.

[36] M. Fishelson, D. Geiger, Exact genetic linkage computations for general pedigrees, Bioinformatics 18 (2002) 189–198.

[37] R. McEliece, D. Mackay, J. Cheng, Turbo decoding as an instance of Pearl's "belief propagation" algorithm, IEEE J. Sel. Areas Comm. 16 (2) (1998) 140–152.

[38] L. R. Rabiner, A tutorial in hidden Markov mdoels and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286.

[39] W. T. Freeman, E. C. Pasztor, O. T. Carmichael, Learning low-level vision, IJCV 40 (1) (2000) 25–47.

[40] M. Beal, N. Jojic, H. Attias, A graphical model for audiovisual object tracking, IEEE Trans. PAMI 25 (7) (2003) 828–836.

[41] Z. W. Tu, X. R. Chen, A. L. Yuille, S. C. Zhu, Image parsing: unifying segmentation, detection and recognition, IJCV 63 (2) (2005) 113–140.

[42] E. Segal, M. Shapira, A. Regev, D. Peér, D. Botstein, D. Koller, N. Friedman, Module networks: identifying regulatory modules and their condition–specific regulators from gene expression data, Nature Genetics 34 (2) (2003) 166–176.

[43] A. Raval, Z. Ghahramani, D. L. Wild, A bayesian network model for protein fold and remote homologue recognition, Bioinformatics 18 (2002) 788–801.

[44] T. Chin, A. Mariano, E. Chassignet, Spatial regression and multiscale approximations for sequential data assimilation in ocean models, Journal of Geophysical Research 104 (C4) (1999) 7991–8014.

[45] J. S. Yedidia, W. T. Freeman, Y. Weiss, Understanding belief propagation and its generalizations, in: IJCAI, 2001.

[46] F. Kschischang, B. Frey, H.-A. Loeliger, Factor graphs and the sum-product algorithm, IEEE Trans. IT 47 (2) (2001) 498–519.

[47] P. Clifford, Markov random fields in statistics, in: G. R. Grimmett, D. J. A. Welsh (Eds.), Disorder in Physical Systems, Oxford University Press, Oxford, 1990, pp. 19–32.

[48] D. R. Cox, N. Wermuth, Multivariate Dependencies – Models, Analysis and Interpretation, Vol. 67 of Monographs on Statistics and Applied Probability, Chapman & Hall, 1996.

[49] R. M. Neal, Probabilistic inference using Markov Chain Monte Carlo methods, Tech. Rep. CRG-TR-93-1, Dept. of Comp. Sci., Univ. of Toronto (1993).

[50] D. MacKay, Introduction to monte carlo methods, in: M. I. Jordan (Ed.), Learning in Graphical Models, MIT Press, 1998.

[51] S. L. Lauritzen, Graphical Models, Oxford University Press, Oxford, 1996.

[52] L. E. Baum, T. Petrie, G. Soules, N. Weiss, A maximization technique occurring in statistical analysis of probabilistic functions of Markov chains, Ann. Math. Stat. 41 (1) (1970) 164–171.

[53] A. Doucet, N. de Freitas, N. Gordon (Eds.), Sequential Monte Carlo in Practice, Springer-Verlag, 2001.

[54] B. Ristic, S. Arulampalam, N. Gordon, Beyond the Kalman Filter: Particle Filters for Tracking Applications, Artech House, Boston/London, 2004.

[55] E. B. Sudderth, A. T. Ihler, W. T. Freeman, A. S. Willsky, Nonparametric belief propagation, in: CVPR, 2003.

[56] M. J. Wainwright, T. S. Jaakkola, A. S. Willsky, Tree–based reparameterization analysis of sum–product and its generalizations, IEEE Trans. IT 49 (5) (2003) 1120–1146.

[57] R. McEliece, S. M. Aji, The generalized distributive law, IEEE Trans. Info. Theory 46 (2) (2000) 325–343.

[58] M. J. Wainwright, T. Jaakkola, A. S. Willsky, Tree consistency and bounds on the performance of the max-product algorithm and its generalizations, Statistics and Computing 14 (2004) 143–166.

[59] D. P. Bertsekas, Dynamic Programming and Optimal Control, Vol. 1, Athena Scientific, Belmont, MA, 1995.

[60] A. J. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Trans. IT 13 (2) (1967) 260–269.

[61] K. Murphy, Y. Weiss, M. Jordan, Loopy-belief propagation for approximate inference: An empirical study, in: UAI 15, 1999, pp. 467–475.

[62] J. S. Yedidia, W. T. Freeman, Y. Weiss, Constructing free energy approximations and generalized belief propagation algorithms, Tech. Rep. 2004-040, MERL (May 2004).

[63] R. Dechter, K. Kask, R. Mateescu, Iterative join-graph propagation, in: UAI, 2002, pp. 128–136.

[64] M. J. Wainwright, M. I. Jordan, Graphical models, exponential families, and variational inference, Tech. Rep. 629, UC Berkeley Dept. of Statistics (Sep. 2003).

[65] N. Srebro, Maximum likelihood bounded tree-width Markov networks, in: UAI 17, 2001.

[66] D. Brown, A note on approximations to discrete probability distributions, Information and Control 2 (1959) 386–392.

[67] C. K. Chow, C. N. Liu, Approximating discrete probability distributions with dependence trees, IEEE Trans. IT 14 (3) (1968) 462–467.

[68] A. P. Dempster, N. M. Laird, D. B. Rubin, Maximum likelihood from incomplete data via EM algorithm, J. R. Stat. Soc. B 39 (1) (1977) 1–38.

[69] D. P. Dee, S. E. Cohn, A. Dalcher, M. Ghil, An efficient algorithm for estimating noise covariances in distributed systems, IEEE Trans. AC 30 (11) (1985) 1057–1065.

[70] A. Willsky, Multiresolution Markov models for signal and image processing, Proc. IEEE 90 (8) (2002) 1396–1458.

[71] A. W. Robertson, S. Kirshner, P. Smyth, Downscaling of daily rainfall occurrence over Northeast Brazil using a hidden Markov model, Journal of Climate 17 (22) (2004) 4407–4424.

[72] A. W. Robertson, S. Kirshner, P. Smyth, S. P. Charles, B. B. Bates, Subseasonal-to-interdecadal variability of the Australian monsoon over North Queensland, Quarterly Journal of the Royal Meteorological Society 132 (2006) 519–542.

[73] P. W. Fieguth, W. C. Karl, A. S. Willsky, C. Wunsch, Multiresolution optimal interpolation and statistical analysis of TOPEX/POSEIDON satellite altimetry, IEEE Trans. Geosci. Remote Sensing 33 (2) (1995) 280–292.

[74] M. Bhatia, W. C. Karl, A. S. Willsky, Tomographic reconstruction and estimation based on multi-scale natural pixel bases, IEEE Trans. IP 6 (1997) 463–478.

[75] M. Schneider, P. W. Fieguth, W. C. Karl, A. S. Willsky, Multiscale methods for the segmentation of images, in: ICASSP, Vol. 4, 1996, pp. 2247–2250.

[76] Jet Propulsion Laboratory, The PO.DAAC Ocean ESIP Tool, http://poet.jpl.nasa.gov/.

[77] R. W. Reynolds, T. M. Smith, Improved global sea surface temperature analyses using optimal interpolation, Journal of Climate 7 (1994) 929–948.

[78] W. W. Irving, P. W. Fieguth, A. S. Willsky, An overlapping tree approach to multiscale stochastic modeling and estimation, IEEE Trans. IP 6 (11) (1997) 1517–1529.

[79] M. J. Wainwright, E. B. Sudderth, A. S. Willsky, Tree-based modeling and estimation of gaussian processes on graphs with cycles, in: NIPS 13, MIT Press, 2000, pp. 661–667.

[80] M. M. Daniel, A. S. Willsky, The modeling and estimation of statistically self-similar processes in a multiresolution framework, IEEE Trans. IT 45 (3) (199) 955–970.

[81] J. P. Hughes, P. Guttorp, S. P. Charles, A non-homogeneous hidden Markov model for precipitation occurrence, J. R. Stat. Soc. C 48 (1) (1999) 15–30.

[82] J. P. Hughes, P. Guttorp, A class of stochastic models for relating synoptic atmospheric patterns to regional hydrologic phenomena, Water Res. Research 30 (1994) 1535–1546.

[83] W. Zucchini, P. Guttorp, A hidden Markov model for space-time precipitation, Water Res. Research 27 (8) (1991) 1917–1923.

[84] J. P. Hughes, P. Guttorp, Incorporating spatial dependence and atmospheric data in a model of precipitation, J. of App. Meteor. 33 (12) (1994) 1503–1515.

[85] M. Meilă, M. I. Jordan, Learning with mixtures of trees, J. Mach. Learn. Research 1 (1) (2000) 1–48.

[86] S. Kirshner, P. Smyth, A. W. Robertson, Conditional Chow-Liu tree structures for modeling discrete-valued vector time series, in: M. Chickering, J. Halpern (Eds.), Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI-04), AUAI Press, 2004, pp. 317–324.

[87] T. M. Cover, Learning in pattern recognition, in: S. Watanabe (Ed.), Methodologies of Pattern Recognition, Academic Press, NY, 1969, pp. 111–132.

[88] M. C. Acock, Y. A. Pachepsky, Estimating missing weather data for agricultural simulations using group method of data handling, Journal of Applied Meteorology 39 (2000) 1176–1184.