Dynamic Factor Graphs for Time Series Modeling

Piotr Mirowski and Yann LeCun

Courant Institute of Mathematical Sciences, New York University,
719 Broadway, New York, NY 10003 USA
{mirowski,yann}@cs.nyu.edu
http://cs.nyu.edu/~mirowski/

Abstract. This article presents a method for training Dynamic Factor Graphs (DFG) with continuous latent state variables. A DFG includes factors modeling joint probabilities between hidden and observed variables, and factors modeling dynamical constraints on hidden variables. The DFG assigns a scalar energy to each configuration of hidden and observed variables. A gradient-based inference procedure finds the minimum-energy state sequence for a given observation sequence. Because the factors are designed to ensure a constant partition function, they can be trained by minimizing the expected energy over training sequences with respect to the factors' parameters. These alternated inference and parameter updates can be seen as a deterministic EM-like procedure. Using smoothing regularizers, DFGs are shown to reconstruct chaotic attractors and to separate a mixture of independent oscillatory sources perfectly. DFGs outperform the best known algorithm on the CATS competition benchmark for time series prediction. DFGs also successfully reconstruct missing motion capture data.

Key words: factor graphs, time series, dynamic Bayesian networks, recurrent networks, expectation-maximization

1 Introduction

1.1 Background

Time series collected from real-world phenomena are often an incomplete picture of a complex underlying dynamical process with a high-dimensional state that cannot be directly observed. For example, human motion capture data gives the positions of a few markers that are the reflection of a large number of joint angles with complex kinematic and dynamical constraints. The aim of this article is to deal with situations in which the hidden state is continuous and high-dimensional, and the underlying dynamical process is highly non-linear, but essentially deterministic. It also deals with situations in which the observations have lower dimension than the state, and the relationship between states and observations may be non-linear. The situation occurs in numerous problems in speech and audio processing, financial data, and instrumentation data, for such

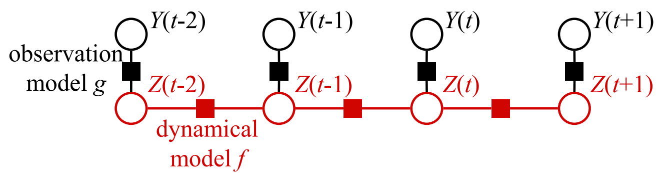

Fig. 1. A simple Dynamical Factor Graph with a 1^{st} order Markovian property, as used in HMMs and state-space models such as Kalman Filters.

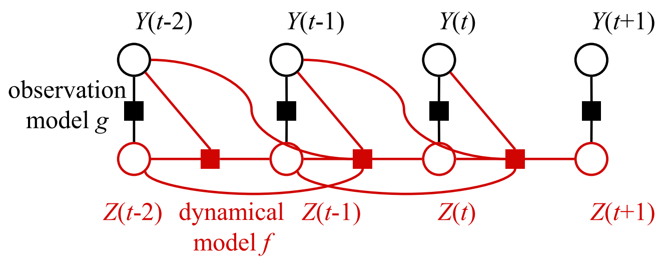

Fig. 2. A Dynamic Factor Graph where dynamics depend on the past two values of both latent state Z and observed variables Y.

tasks as prediction and source separation. It applies in particular to univariate chaotic time series which are often the projection of a multidimensional attractor generated by a multivariate system of nonlinear equations.

The simplest approach to modeling time series relies on time-delay embedding: the model learns to predict one sample from a number of past samples with a limited temporal span. This method can use linear auto-regressive models, as well as non-linear ones based on kernel methods (e.g. support-vector regression [12, 13]), neural networks (including convolutional networks such as time delay neural networks [6, 17]), and other non-linear regression models. Unfortunately, these approaches have a hard time capturing hidden dynamics with long-term dependency because the state information is only accessible indirectly (if at all) through a (possibly very long) sequence of observations [2].

To capture long-term dynamical dependencies, the model must have an internal state with dynamical constraints that predict the state at a given time from the states and observations at previous times (e.g. a state-space model). In general, the dependencies between state and observation variables can be expressed in the form of a *Factor Graph* [5] for sequential data, in which a graph motif is replicated at every time step. An example of such a representation of a state-space model is shown in Figure 1. Groups of variables (circles) are connected to a factor (square) if a dependency exists between them. The factor can be expressed in the negative log domain: each factor computes an energy value that can be interpreted as the negative log likelihood of the configuration of the variables it connects with. The total energy of the system is the sum of the factors' energies, so that the maximum likelihood configuration of variables can be obtained by minimizing the total energy.

Figure 1 shows the structure used in Hidden Markov Models (HMM) and Kalman Filters, including Extended Kalman Filters (EKF) which can model non-linear dynamics. HMMs can capture long-term dependencies, but they are limited to discrete state spaces. Discretizing the state space of a high-dimensional continuous dynamical process to make it fit into the HMM framework is often impractical. Conversely, EKFs deal with continuous state spaces with non-linear dynamics, but much of the machinery for inference and for training the parameters is linked to the problem of marginalizing over hidden state distributions and to propagating and estimating the covariances of the state distributions. This has lead several authors to limit the discussion to dynamics and observation functions that are linear or radial-basis functions networks [18, 4].

1.2 Dynamical Factor Graphs

By contrast with current state-space methods, our primary interest is to model processes whose underlying dynamics are essentially deterministic, but can be highly complex and non-linear. Hence our model will allow the use of complex functions to predict the state and observations, and will sacrifice the probabilistic nature of the inference. Instead, our inference process (including during learning) will produce the most likely (minimum energy) sequence of states given the observations. We call this method *Dynamic Factor Graph* (DFG), a natural extension of Factor Graphs specifically tuned for sequential data.

To model complex dynamics, the proposed model allows the state at a given time to depend on the states and observations over several past time steps. The corresponding DFG is depicted in Figure 2. The graph structure is somewhat similar to that of Taylor and Hinton's Conditional Restricted Boltzmann Machine [16]. Ideally, training a CRBM would consist in minimizing the negative log-likelihood of the data under the model. But computing the gradient of the log partition function with respect to the parameters is intractable, hence Taylor and Hinton propose to use a form of the contrastive divergence procedure, which relies on Monte-Carlo sampling. To avoid costly sampling procedures, we design the factors in such a way that the partition function is constant, hence the likelihood of the data under the model can be maximized by simply minimizing the average energy with respect to the parameters for the optimal state sequences. To achieve this, the factors are designed so that the conditional distributions of state $Z(t)$ given previous states and observation[1], and the conditional distribution of the observation $Y(t)$ given the state $Z(t)$ are both Gaussians with a fixed covariance.

In a nutshell, the proposed training method is as follows. Given a training observation sequence, the optimal state sequence is found by minimizing the

[1] Throughout the article, $Y(t)$ denotes the value at time t of multivariate time series Y, and $\mathbf{Y}_{t-p}^{t-1} \equiv \{Y(t-t), Y(t-2), \ldots, Y(t-p)\}$ a time window of p samples preceding the current sample. $Z(t)$ denotes the hidden state at time t.

energy using a gradient-based minimization method. Second, the parameters of the model are updated using a gradient-based procedure so as to decrease the energy. These two steps are repeated over all training sequences. The procedure can be seen as a sort of deterministic generalized EM procedure in which the latent variable distribution is reduced to its mode, and the model parameters are optimized with a stochastic gradient method. The procedure assumes that the factors are differentiable with respect to their input variables and their parameters. This simple procedure will allow us to use sophisticated non-linear models for the dynamical and observation factors, such as stacks of non-linear filter banks (temporal convolutional networks). It is important to note that *the inference procedure operates at the sequence level*, and produces the most likely state sequence that best explains the entire observation. In other words, future observations may influence previous states.

In the DFG shown in Figure 1, the dynamical factors compute an energy term of the form $E_d(t) = ||Z(t) - f(X(t), Z(t-1))||^2$, which can seen as modeling the state $Z(t)$ as $f(X(t), Z(t-1))$ plus some Gaussian noise variable with a fixed variance $\epsilon(t)$ (inputs $X(t)$ are not used in experiments in this article). Similarly, the observation factors compute the energy $E_o(t) = ||Y(t) - g(Z(t))||^2$, which can be interpreted as $Y(t) = g(Z(t)) + \omega(t)$, where $\omega(t)$ is a Gaussian random variable with fixed variance.

Our article is organized in three additional sections. First, we explain the gradient-based approximate algorithm for parameter learning and deterministic latent state inference in the DFG model (2). We then evaluate DFGs on toy, benchmark and real-world datasets (3). Finally, we compare DFGs to previous methods for deterministic nonlinear dynamical systems and to training algorithms for Recurrent Neural Networks (4).

2 Methods

The following subsections detail the deterministic nonlinear (neural networks-based) or linear architectures of the proposed Dynamic Factor Graph (2.1) and define the EM-like, gradient-based inference (2.2) and learning (2.4) algorithms, as well as how DFGs are used for time-series prediction (2.3).

2.1 A Dynamic Factor Graph

Similarly to Hidden Markov Models, our proposed Dynamic Factor Graph contains an observation and a dynamical factors/models (see Figure 1), with corresponding observed outputs and latent variables.

The *observation model* g links latent variable $Z(t)$ (an m-dimensional vector) to the observed variable $Y(t)$ (an n-dimensional vector) at time t under Gaussian noise model $\omega(t)$ (because the quadratic observation error is minimized). g can be nonlinear, but we considered in this article linear observation models, i.e. an $n \times m$ matrix parameterized by a weight vector $\mathbf{W}_o$. This model can be simplified even further by imposing each observed variable $y_i(t)$ of the multivariate time

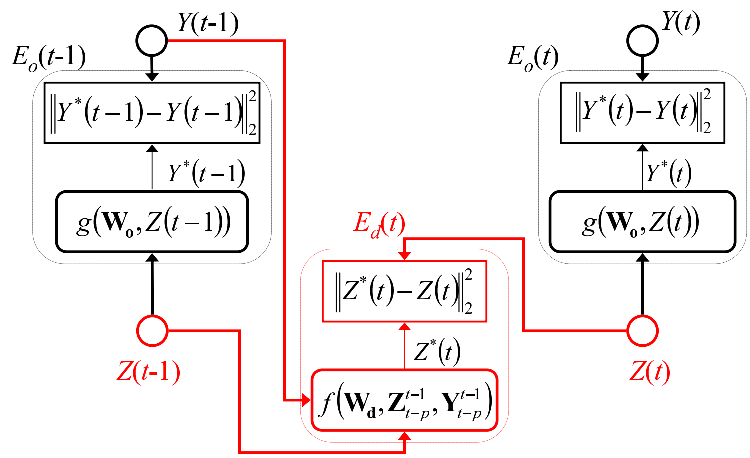

Fig. 3. Energy-based graph of a DFG with a 1^{st} order Markovian architecture and additional dynamical dependencies on past observations. Observations $Y(t)$ are inferred as $Y^*(t)$ from latent variables $Z(t)$ using the observation model parameterized by $\mathbf{W}_o$. The (non)linear dynamical model parameterized by $\mathbf{W}_d$ produces transitions from a sequence of latent variables $\mathbf{Z}_{t-p}^{t-1}$ and observed output variables $\mathbf{Y}_{t-p}^{t-1}$ to $Z(t)$ (here $p = 1$). The total energy of the configuration of parameters and latent variables is the sum of the observation $E_o(.)$ and dynamic $E_d(.)$ errors.

series Y to be the sum of k latent variables, with $m = k \times n$, and each latent variable contributing to only one observed variable. In the general case, the generative output is defined as:

$$Y^*(t) \equiv g\left(\mathbf{W}_o, Z(t)\right) \tag{1}$$
$$Y(t) = Y^*(t) + \omega(t) \tag{2}$$

In its simplest form, the linear or nonlinear *dynamical model* f establishes a causal relationship between a sequence of p latent variables $\mathbf{Z}_{t-p}^{t-1}$ and latent variable $Z(t)$, under Gaussian noise model $\epsilon(t)$ (because the quadratic dynamic error is minimized). (3) thus defines p^{th} order Markovian dynamics (see Figure 1 where $p = 1$). The dynamical model is parameterized by vector $\mathbf{W_d}$.

$$Z^*(t) \equiv f\left(\mathbf{W}_d, \mathbf{Z}_{t-p}^{t-1}\right) \tag{3}$$
$$Z(t) = Z^*(t) + \epsilon(t) \tag{4}$$

Typically, one can use simple multivariate autoregressive linear functions to map the state variables, or can also resort to nonlinear dynamics modeled by a Convolutional Network [7] with convolutions (FIR filters) across time, as in Time-Delay Neural Networks [6, 17].

Other dynamical models, different from the Hidden Markov Model, are also possible. For instance, latent variables $Z(t)$ can depend on a sequence of p past latent variables $\mathbf{Z}_{t-p}^{t-1}$ and p past observations $\mathbf{Y}_{t-p}^{t-1}$, using the same error term $\epsilon(t)$, as explained in (5) and illustrated on Figure 2.

$$Z^*(t) \equiv f\left(\mathbf{W}_d, \mathbf{Z}_{t-p}^{t-1}, \mathbf{Y}_{t-p}^{t-1}\right) \tag{5}$$

$$Z(t) = Z^*(t) + \epsilon(t) \tag{6}$$

Figure 3 displays the interaction between the observation (1) and dynamical (5) models, the observed Y and latent Z variables, and the quadratic error terms.

2.2 Inference in Dynamic Factor Graphs

Let us define the following *total* (7), *dynamical* (8) and *observation* (9) energies (quadratic errors) on a given time interval $[t_a, \ldots, t_b]$, where respective weight coefficients α, β are positive constants (in this article, $\alpha = \beta = 0.5$):

$$E\left(\mathbf{W}_d, \mathbf{W}_o, \mathbf{Y}_{t_a}^{t_b}\right) = \sum_{t=t_a}^{t_b} \left[\alpha E_d(t) + \beta E_o(t)\right] \tag{7}$$

$$E_d(t) \equiv \min_Z E_d\left(\mathbf{W}_d, \mathbf{Z}_{t-p}^{t-1}, Z(t)\right) \tag{8}$$

$$E_o(t) \equiv \min_Z E_o\left(\mathbf{W}_o, Z(t), Y(t)\right) \tag{9}$$

Inferring the sequence of latent variables $\{Z(t)\}_t$ in (7) and (8) is equivalent to simultaneous minimization of the sum of dynamical and observation energies at all times t:

$$E_d\left(\mathbf{W}_d, \mathbf{Z}_{t-p}^{t-1}, Z(t)\right) = ||Z^*(t) - Z(t)||_2^2 \tag{10}$$

$$E_o\left(\mathbf{W}_o, Z(t), Y(t)\right) = ||Y^*(t) - Y(t)||_2^2 \tag{11}$$

Observation and dynamical errors are expressed separately, either as Normalized Mean Square Errors (NMSE) or Signal-to-Noise Ratio (SNR).

2.3 Prediction in Dynamic Factor Graphs

Assuming fixed parameters $\mathbf{W}$ of the DFG, two modalities are possible for the prediction of unknown observed variables Y.

- *Closed-loop (iterated) prediction*: when the continuation of the time series is unknown, the only relevant information comes from the past. One uses the dynamical model to predict $Z^*(t)$ from $\mathbf{Y}_{t-p}^{t-1}$ and inferred $\mathbf{Z}_{t-p}^{t-1}$, set $Z(t) = Z^*(t)$, use the observation model to compute prediction $Y^*(t)$ from $Z(t)$, and iterate as long as necessary. If the dynamics depend on past observations, one also needs to rely on predictions $Y^*(t)$ in (5).

– *Prediction as inference*: this is the case when only some elements of Y are unknown (e.g. estimation of missing motion-capture data). First, one infers latent variables through gradient descent, and simply does not backpropagate errors from unknown observations. Then, missing values $y_i^*(t)$ are predicted from corresponding latent variables $Z(t)$.

2.4 Training of Dynamic Factor Graphs

Learning in an DFG consists in adjusting the parameters $\mathbf{W} = \left[\mathbf{W}_d^T, \mathbf{W}_o^T\right]$ in order to minimize the loss $L(\mathbf{W}, \mathbf{Y}, \tilde{Z})$:

$$L(\mathbf{W}, Y, Z) = E\left(\mathbf{W}, Y\right) + R_z(Z) + R\left(\mathbf{W}\right) \tag{12}$$

$$\tilde{Z} = argmin_Z L(\tilde{\mathbf{W}}, \mathbf{Y}, Z) \tag{13}$$

$$\tilde{\mathbf{W}} = argmin_{\mathbf{W}} L(\mathbf{W}, \mathbf{Y}, \tilde{Z}) \tag{14}$$

where $R(\mathbf{W})$ is a regularization term on the weights $\mathbf{W}_d$ and $\mathbf{W}_o$, and $R_z(\mathbf{Z})$ represents additional constraints on the latent variables further detailed. Minimization of this loss is done iteratively in an Expectation-Maximization-like fashion in which the states Z play the role of auxiliary variables. During inference, values of the model parameters are clamped and the hidden variables are relaxed to minimize the energy. The inference described in part (2.2) and equation (13) can be considered as the *E-step* (state update) of a gradient-based version of the EM algorithm. During learning, model parameters $\mathbf{W}$ are optimized to give lower energy to the current configuration of hidden and observed variables. The parameter-adjusting *M-step* (weight update) described by (14) is also gradient-based.

In its current implementation, the E-step inference is done by gradient descent on Z, with learning rate η_z typically equal to 0.5. The convergence criterion is when energy (7) stops decreasing. The M-step parameter learning is implemented as a stochastic gradient descent (diagonal Levenberg-Marquard) [8] with individual learning rates per weight (re-evaluated every 10000 weight updates) and global learning rate η_w typically equal to 0.01. These parameters were found by trial and error on a grid of possible values.

The state inference is not done on the full sequence at once, but on *mini-batches* (typically 20 to 100 samples), and the weights get updated once after each mini-batch inference, similarly to the Generalized EM algorithm. During one epoch of training, the batches are selected randomly and overlap in such a way that each state variable $Z(t)$ is re-inferred at least a dozen times in different mini-batches. This learning approximation echoes the one in regular stochastic gradient with no latent variables and enables to speed up the learning of the weight parameters.

The learning algorithm turns out to be particularly simple and flexible. The hidden state inference is however under-constrained, because of the higher dimensionality of the latent states and despite the dynamical model. For this reason, this article proposes to (in)directly regularize the hidden states in several ways.

First, one can add to the loss function an L_1 regularization term $R(\mathbf{W})$ on the weight parameters. This way, the dynamical model becomes "sparse" in terms of its inputs, e.g. the latent states. Regarding the term $R_z(\mathbf{Z})$, an L_2 norm on the hidden states $Z(t)$ limits their overall magnitude, and an L_1 norm enforces their sparsity both in time and across dimensions. Respective regularization coefficients λ_w and λ_z typically range from 0 to 0.1.

2.5 Smoothness Penalty on Latent Variables

The second type of constraints on the latent variables is the *smoothness penalty*. In an apparent contradiction with the dynamical model (3), this penalty forces two consecutive variables $z_i(t)$ and $z_i(t+1)$ to be similar. One can view it as an attempt at inferring slowly varying hidden states and at reducing noise in the states (which is particularly relevant when observation Y is sampled at a high frequency). By consequence, the dynamics of the latent states are smoother and simpler to learn. Constraint (15) is easy to derivate w.r.t. a state $z_i(t)$ and to integrate into the gradient descent optimization (13):

$$R_z\left(\mathbf{Z}_t^{t+1}\right) = \sum_i (z_i(t) - z_i(t+1))^2 \tag{15}$$

In addition to the smoothness penalty, we have investigated the decorrelation of multivariate latent variables $Z(t) = (z_1(t), z_2(t), \dots, z_m(t))$. The justification was to impose to each component $\mathbf{z}_i$ to be independent, so that it followed its own dynamics, but we have not obtained satisfactory results yet. As reported in the next section, the interaction of the dynamical model, weight sparsification and smoothness penalty already enables the separation of latent variables.

3 Experimental Evaluation

First, working on toy problems, we investigate the latent variables that are inferred from an observed time series. We show that using smoothing regularizers, DFGs are able to perfectly separate a mixture of independent oscillatory sources (3.1), as well as to reconstruct the Lorenz chaotic attractor in the inferred state space (3.2). Secondly, we apply DFGs to two time series prediction and modeling problems. Subsection (3.3) details how DFGs outperform the best known algorithm on the CATS competition benchmark for time series prediction. In (3.4) we reconstruct realistic missing human motion capture marker data in a walk sequence.

3.1 Asynchronous Superimposed Sine Waves

The goal is to model a time series constituted by a sum of 5 asynchronous sinusoids: $y(t) = \sum_{j=1}^{5} \sin(\lambda_j t)$ (see Fig. 4a). Each component $x_j(t)$ can be

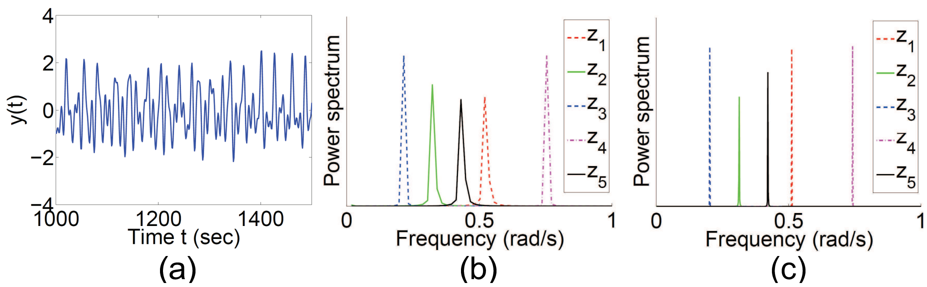

Fig. 4. (a) Superposition of five asynchronous sinusoids: $y(t) = \sum_{j=1}^{5} \sin(\lambda_j t)$ where $\lambda_1 = 0.2$, $\lambda_2 = 0.311$, $\lambda_3 = 0.42$, $\lambda_4 = 0.51$ and $\lambda_5 = 0.74$. Spectrum analysis shows that after learning and inference, each reconstructed state z_i isolates only one of the original sources x_j, both on the training (b) and testing (c) datasets.

considered as a "source", and $y(t)$ is a mixture. This problem has previously been tackled by employing Long-Short Term Memory (LSTM), a special architecture of Recurrent Neural Networks that needs to be trained by genetic optimization [19].

After EM training and inference of hidden variables $Z(t)$ of dimension $m = 5$, frequency analysis of the inferred states on the training (Fig. 4b) and testing (Fig. 4c) datasets showed that each latent state $z_i(t)$ reconstructed one individual sinusoid. In other words, the 5 original sources from the observation mixture $y(t)$ were inferred on the 5 latent states. The observation SNR of 64dB, and the dynamical SNR of 54dB, on both the training and testing datasets, proved both that the dynamics of the original time series $y(t)$ were almost perfectly reconstructed. DFGs outperformed LSTMs on that task since the multi-step iterated (closed-loop) prediction of DFG did not decrease in SNR even after thousands of iterations, contrary to [19] where a reduction in SNR was already observed after around 700 iterations.

As architecture for the dynamical model, 5 independent Finite Impulse Response (FIR) filters of order 25 were chosen to model the state transitions: each of them acts as a band-pass filter and models an oscillator at a given frequency. One can hypothesize that the smoothness penalty (15), weighted by a small coefficient of 0.01 in the state regularization term $R_z(\mathbf{Z})$ helped shape the hidden states into perfect sinusoids. Note that the states or sources were made independent by employing five independent dynamical models for each state. This specific usage of DFG can be likened to Blind Source Separation from an unique source, and the use of independent filters for the latent states (or sources) echoes the approach of BSS using linear predictability and adaptive band-pass filters.

3.2 Lorenz Chaotic Data

As a second application, we considered the 3-variable (x_1, x_2, x_3) Lorenz dynamical system [11] generated by parameters $\rho = 16, b = 4, r = 45.92$ as in

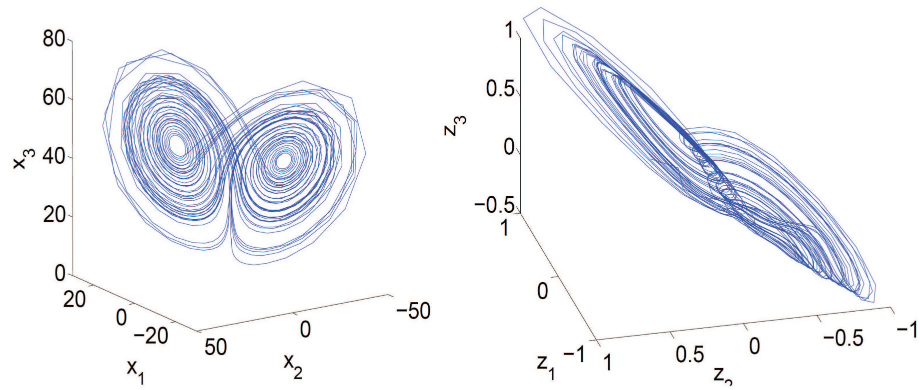

Fig. 5. Lorenz chaotic attractor (left) and the reconstructed chaotic attractor from the latent variables $Z(t) = \{z_1(t), z_2(t), z_3(t)\}$ after inference on the testing dataset (right).

Table 1. Comparison of 1-step prediction error using Support Vector Regression, with the errors of the dynamical and observation models of DFGs, measured on the Lorenz test dataset and expressed as signal-to-noise ratios.

Architecture	SVR	DFG
Dynamic SNR	41.6 dB	46.2 dB
Observation SNR	-	31.6 dB

[12] (see Fig. 5a). Observations consisted in one-dimensional time series $y(t) = \sum_{j=1}^{3} x_j(t)$.

The DFG was trained on 50s (2000 samples) and evaluated on the following 40s (1600 samples) of y. Latent variables $Z(t) = (z_1(t), z_2(t), z_3(t))$ had dimension $m = 3$, as it was greater than the attractor correlation dimension of 2.06 and equal to the number of explicit variables (sources). The dynamical model was implemented as a 3-layered convolutional network. The first layer contained 12 convolutional filters covering 3 time steps and one latent component, replicated on all latent components and every 2 time samples. The second layer contained 12 filters covering 3 time steps and all previous hidden units, and the last layer was fully connected to the previous 12 hidden units and 3 time steps. The dynamical model was autoregressive on $p = 11$ past values of Z, with a total of 571 unique parameters. "Smooth" consecutive states were enforced (15), thanks to the state regularization term $R_z(Z)$ weighted by a small coefficient of 0.01. After training the parameters of DFG, latent variables Z were inferred on the full length of the training and testing dataset, and plotted in 3D values of triplets $(z_1(t), z_2(t), z_3(t))$ (see Fig. 5b).

The 1-step dynamical SNR obtained with a training set of 2000 samples was higher than the 1-step prediction SNR reported for Support Vector Regression (SVR) [12] (see Table 1). According to the Takens theorem [15], it is possible to

Table 2. Prediction results on the CATS competition dataset comparing the best algorithm (Kalman Smoothers [14]) and Dynamic Factor Graphs. E_1 and E_2 are unnormalized MSE, measured respectively on all five missing segments or on the first four missing segments.

Architecture	Kalman smoother	DFG
E1 (5 segments)	408	390
E2 (4 segments)	346	288

reconstruct an unknown (hidden) chaotic attractor from an adequately long window of observed variables, using time-delay embedding on $y(t)$, but we managed to reconstruct this attractor on the latent states $(z_1(t), z_2(t), z_3(t))$ inferred both from the training or testing datasets (Fig. 5). Although one of the "wings" of the reconstructed butterfly-shaped attractor is slightly twisted, one can clearly distinguish two basins of attraction and a chaotic orbit switching between one and the other. The reconstructed latent attractor has correlation dimensions 1.89 (training dataset) and 1.88 (test dataset).

3.3 CATS Time Series Competition

Dynamic Factor Graphs were evaluated on time series prediction problems using the CATS benchmark dataset [9]. The goal of the competition was the prediction of 100 missing values divided into five groups of 20, the last group being at the end of the provided time series. The dataset presented a noisy and chaotic behaviour commonly observed in financial time series such as stock market prices.

In order to predict the missing values, the DFG was trained for 10 epochs on the known data (5 chunks of 980 points each). 5-dimensional latent states on the full 5000 point test time series were then inferred in one E-step, as described in section 2.3. The dynamical factor was the same as in section 3.2. As shown in Table 2, the DFG outperformed the best results obtained at the time of the competition, using a Kalman Smoother [14], and managed to approximate the behavior of the time series in the missing segments.

3.4 Estimation of Missing Motion Capture Data

Finally, DFGs were applied to the problem of estimating missing motion capture data. Such situations can arise when "the motion capture process [is] adversely affected by lighting and environmental effects, as well as noise during recording" [16]. Motion capture data[2] Y consisted of three 49-dimensional time series representing joint angles derived from 17 markers and coccyx, acquired on a subject walking and turning, and downsampled to 30Hz. Two sequences of 438 and 3128 samples were used for training, and one sequence of 260 samples for testing.

[2] We used motion capture data from the MIT database as well as sample Matlab code for motion playback and conversion, developped or adapted by Taylor, Hinton and Roweis, available at: `http://www.cs.toronto.edu/~gwtaylor/`.

Table 3. Reconstruction error (NMSE) for 4 sets of missing joint angles from motion capture data (two blocks of 65 consecutive frames, about 2s, on either the left leg or entire upper body). DFGs are compared to standard nearest neighbors matching.

Method	Nearest Neighb.	DFG
Missing leg 1	0.77	0.59
Missing leg 2	0.47	0.39
Missing upper body 1	1.24	0.9
Missing upper body 2	0.8	0.48

We reproduced the experiments from [16], where Conditional Restricted Boltzman Machines (CRBM) were utilized. On the test sequence, two different sets of joint angles were erased, either the left leg (1) or the entire upper body (2). After training the DFG on the training sequences, missing joint angles $y_i(t)$ were inferred through the E-step inference. The DFG was the same as in sections 3.2 and 3.3, but with 147 hidden variables (3 per observed variable) and no smoothing. Table 3 shows that DFGs significantly outperformed nearest neighbor interpolation (detailed in [16]), by taking advantage of the motion dynamics modeled through dynamics on latent variables. Contrary to nearest neighbors matching, DFGs managed to infer smooth and realistic leg or upper body motion. Videos comparing the original walking motion sequence, and the DFG- and nearest neighbor-based reconstructions are available at `http://cs.nyu.edu/~mirowski/pub/mocap/`. Figure 6 illustrates the DFG-based reconstruction (we did not include nearest neighbor interpolation resuts because the reconstructed motion was significantly more "hashed" and discontinuous).

4 Discussion

In this section, we establish a comparison with other nonlinear dynamical systems with latent variables (4.1) and suggest that DFGs could be seen as an alternative method for training Recurrent Neural Networks (4.2).

4.1 Comparison with Other Nonlinear Dynamical Systems with Latent States

An earlier model of nonlinear dynamical system with hidden states is the Hidden Control Neural Network [10], where latent variables $Z(t)$ are added as an additional input to the dynamical model on the observations. Although the dynamical model is stationary, the latent variable $Z(t)$ modulates its dynamics, enabling a behavior more complex than in pure autoregressive systems. The training algorithm iteratively optimizes the weights $\mathbf{W}$ of the Time-Delay Neural Network (TDNN) and latent variables Z, inferred as $\tilde{Z} \equiv argmin_Z \sum_t ||Y(t) - f_{\tilde{\mathbf{W}}} \left(Y(t-1), Z\right)||^2$.

The latter algorithm is likened to approximate maximum likelihood estimation, and iteratively finds a sequence of dynamic-modulating latent variables and learns dynamics on observed variables. DFGs are more general, as they allow the latent variables $Z(t)$ not only to modulate the dynamics of observed variables, but also to generate the observations $Y(t)$, as in DBNs. Moreover, [10] does not introduce dynamics between the latent variables themselves, whereas DFGs model complex nonlinear dynamics where hidden states $Z(t)$ depend on past states $\mathbf{Y}_{t-p}^{t-1}$ and observations $\mathbf{Z}_{t-p}^{t-1}$. Because our method benefits from highly complex non-linear dynamical factors, implemented as multi-stage temporal convolutional networks, it differs from other latent states and parameters estimation techniques, which generally rely on radial-basis functions [18, 4].

The DFG introduced in this article also differs from another, more recent, model of DBN with deterministic nonlinear dynamics and explicit inference of latent variables. In [1], the hidden state inference is done by message passing in the forward direction only, whereas our method suggests hidden state inference as an iterative relaxation, i.e. a forward-backward message passing until "equilibrium".

In a limit case, DFGs could be restricted to a deterministic latent variable generation process like in [1]. One can indeed interpret the dynamical factor as hard constraints, rather than as an energy function. This can be done by setting the dynamical weight α to be much larger than the observation weight β in (7).

4.2 An Alternative Inference and Learning for Recurrent Neural Networks

An alternative way to model long-term dependencies is to use recurrent neural networks (RNN). The main difference with the proposed DFG model is that RNN use fully deterministic noiseless mappings for the state dynamics and the observations. Hence, there is no other inference procedure than running the network forward in time. Unlike with DFG, the state at time t is fully determined by the previous observations and states, and does not depend on future observations.

Exact gradient descent learning algorithms for Recurrent Neural Networks (RNN), such as Backpropagation Through Time (BPTT) or Real-Time Recurrent Learning (RTRL) [20], have limitations. The well-known problem of vanishing gradients is responsible for RNN to forget, during training, outputs or activations that are more than a dozen time steps back in time [2]. This is not an issue for DFG because the inference algorithm effectively computes "virtual targets" for the function f at every time step.

The faster of the two algorithms, BPTT, requires $O(T|\mathbf{W}|)$ weight updates per training epoch, where $|\mathbf{W}|$ is the number of parameters and T the length of the training sequence. The proposed EM-like procedure, which is dominated by the E-step, requires $O(aT|\mathbf{W}|)$ operations per training epoch, where a is the average number of E-step gradient descent steps before convergence (a few to a few dozens if the state learning rate is set properly).

Moreover, because the E-step optimization of hidden variables is done on mini-batches, longer sequences T simply provide with more training examples and thus facilitate learning; the increase in computational complexity is linear with T.

5 Conclusion

This article introduces a new method for learning deterministic nonlinear dynamical systems with highly complex dynamics. Our approximate training method is gradient-based and can be likened to Generalized Expectation-Maximization.

We have shown that with proper smoothness constraints on the inferred latent variables, Dynamical Factor Graphs manage to perfectly reconstruct multiple oscillatory sources or a multivariate chaotic attractor from an observed one-dimensional time series. DFGs also outperform Kalman Smoothers and other neural network techniques on a chaotic time series prediction tasks, the CATS competition benchmark. Finally, DFGs can be used for the estimation of missing motion capture data. Proper regularization such as smoothness or a sparsity penalty on the parameters enable to avoid trivial solutions for high-dimensional latent variables. We are now investigating the applicability of DFG to learning genetic regulatory networks from protein expression levels with missing values.

Acknowledgments. The authors wish to thank Marc'Aurelio Ranzato for fruitful discussions and Graham Williams for his feedback and dataset.

References

1. Barber, D.: Dynamic bayesian networks with deterministic latent tables. In: Advances in Neural Information Processing Systems NIPS'03, pp. 729–736. MIT Press, Cambridge MA (2003)
2. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks 5, 157–166 (1994)
3. Dempster, A., Laird, N., Rubin, D.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B, 39, 1–38 (1977)
4. Ghahramani, Z., Roweis, S.: Learning nonlinear dynamical systems using an EM algorithm. In: Advances in Neural Information Processing Systems NIPS'99. Morgan Kaufmann, MIT Press, Cambridge MA (1999)
5. Kschischang, F., Frey, B., H.-A., L.: Factor graphs and the sum-product algorithm. IEEE Transactions on Information Theory 47, 498–519 (2001)
6. Lang, K., Hinton, G.: The development of the time-delay neural network architecture for speech recognition. Technical Report CMU-CS-88-152, Carnegie-Mellon University (1988)
7. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324 (1998a)
8. LeCun, Y., Bottou, L., Orr, G., Muller, K.: Efficient backprop. Lecture Notes in Computer Science. Berlin/Heidelberg: Springer. (1998b)
9. Lendasse, A., Oja, E., Simula, O.: Time series prediction competition: The CATS benchmark. In: Proceedings of IEEE International Joint Conference on Neural Networks IJCNN, pp. 1615–1620 (2004)

10. Levin, E.: Hidden control neural architecture modeling of nonlinear time-varying systems and its applications. IEEE Transactions on Neural Networks 4, 109–116 (1993)
11. Lorenz, E.: Deterministic nonperiodic flow. Journal of Atmospheric Sciences 20, 130–141 (1963)
12. Mattera, D., Haykin, S.: Support vector machines for dynamic reconstruction of a chaotic system. In: Scholkopf, B., Burges, C.J.C., Smola, A.J. (eds) Advances in Kernel Methods: Support Vector Learning, 212–239. MIT Press, Cambridge MA (1999)
13. Muller, K., Smola, A., Ratsch, G., Scholkopf, B., Kohlmorgen, J., Vapnik, V.: Using support vector machines for time-series prediction. In: Scholkopf, B., Burges, C.J.C., Smola, A.J. (eds) Advances in Kernel Methods: Support Vector Learning, 212–239. MIT Press, Cambridge MA (1999)
14. Sarkka, S., Vehtari, A., Lampinen, J.: Time series prediction by kalman smoother with crossvalidated noise density. In: Proceedings of IEEE International Joint Conference on Neural Networks IJCNN, pp. 1653–1657 (2004)
15. Takens, F.: Detecting strange attractors in turbulence. Lecture Notes in Mathematics 898, 336–381 (1981)
16. Taylor, G., Hinton, G., Roweis, S.: Modeling human motion using binary latent variables. In: Advances in Neural Information Processing Systems NIPS'06. Morgan Kaufmann, MIT Press, Cambridge MA (2006)
17. Wan, E.: Time series prediction by using a connectionist network with internal delay lines. In: Weigend, A.S., Gershenfeld, N.A. (eds) Time Series Prediction: Forecasting the Future and Understanding the Past, 195–217. Addison-Wesley, Reading MA (1993)
18. Wan, E., Nelson, A.: Dual kalman filtering methods for nonlinear prediction, estimation, and smoothing. In: Advances in Neural Information Processing Systems (1996)
19. Wierstra, D., Gomez, F., Schmidhuber, J.: Modeling systems with internal state using Evolino. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1795–2005 (2005)
20. Williams, R., Zipser, D.: Gradient-based learning algorithms for recurrent networks and their computational complexity. In: Backpropagation: Theory, Architectures and Applications, 433–486. Lawrence Erlbaum Associates (1995)

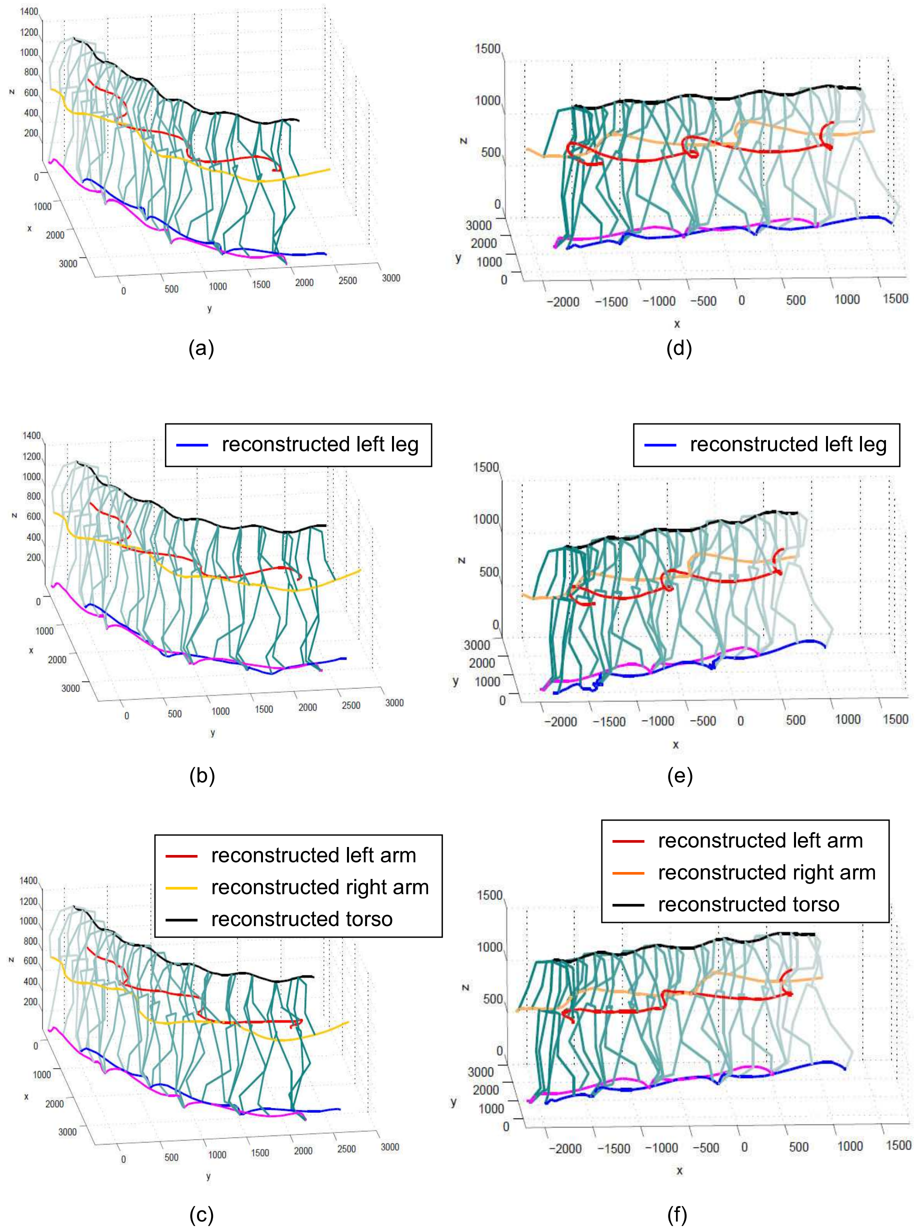

Fig. 6. Application of a DFG for the reconstruction of missing joint angles from motion capture marker data (1 test sequence of 260 frames at 30Hz). 4 sets of joint angles were alternatively "missing" (erased from the test data): 2 sequences of 65 frames, of either left leg or the entire upper body. (a) Subsequence of 65 frames at the beginning of the test data. (b) Reconstruction result after erasing the left leg markers from (a). (c) Reconstruction results after erasing the entire upper body markers from (a). (d) Subsequence of 65 frames towards the end of the test data. (e) Reconstruction result after erasing the left leg markers from (d). (f) Reconstruction results after erasing the entire upper body markers from (d).