Composable, distributed-state models for high-dimensional time series

by

Graham William Taylor

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Composable, distributed-state models
for high-dimensional time series

Graham William Taylor
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto, 2009

In this thesis we develop a class of nonlinear generative models for high-dimensional time series. The first key property of these models is their distributed, or "componential" latent state, which is characterized by binary stochastic variables which interact to explain the data. The second key property is the use of an undirected graphical model to represent the relationship between latent state (features) and observations. The final key property is composability: the proposed class of models can form the building blocks of deep networks by successively training each model on the features extracted by the previous one.

We first propose a model based on the Restricted Boltzmann Machine (RBM) that uses an undirected model with binary latent variables and real-valued "visible" variables. The latent and visible variables at each time step receive directed connections from the visible variables at the last few time-steps. This "conditional" RBM (CRBM) makes on-line inference efficient and allows us to use a simple approximate learning procedure. We demonstrate the power of our approach by synthesizing various motion sequences and by performing on-line filling in of data lost during motion capture. We also explore CRBMs as priors in the context of Bayesian filtering applied to multi-view and monocular 3D person tracking.

We extend the CRBM in a way that preserves its most important computational properties and introduces multiplicative three-way interactions that allow the effective interaction weight between two variables to be modulated by the dynamic state of a third variable. We introduce a factoring of the

implied three-way weight tensor to permit a more compact parameterization. The resulting model can capture diverse styles of motion with a single set of parameters, and the three-way interactions greatly improve its ability to blend motion styles or to transition smoothly among them.

In separate but related work, we revisit Products of Hidden Markov Models (PoHMMs). We show how the partition function can be estimated reliably via Annealed Importance Sampling. This enables us to demonstrate that PoHMMs outperform various flavours of HMMs on a variety of tasks and metrics, including log likelihood.

Acknowledgements

Most importantly, I would like to thank my advisor, Geoff Hinton, whose special blend of wisdom and humor have made the last five years enlightening and enjoyable. I would also like to thank my co-advisor, Sam Roweis, who provided energy and creativity throughout my time in Toronto but especially during the first two years where I needed the most guidance.

I was fortunate to have a talented and open-minded supervisory committee which included Rich Zemel and Aaron Hertzmann. Chris Bregler most graciously agreed to not only act as an external appraiser but also to fly into Toronto for the defense when it looked like we might not have enough committee members physically present. This thesis has certainly benefited from their encouragement and constructive comments.

Leon Sigal was instrumental in developing the work on tracking (Chapter 6). He drafted much of Sec. 6.1-6.3. David Fleet also offered many helpful suggestions for this chapter. Some of the ideas presented in Chapter 5 were sparked while collaborating with Stefano Corazza and others at Animotion, Inc. My work also benefited from collaboration with Niko Troje, Matt Zeiler and the pigeon colony at Queen's University. Mike Seltzer and the speech group at Microsoft Research provided me with a stimulating internship opportunity.

Vinod Nair and I started our PhD programs at the same time and have been great friends throughout. He carefully proofread this thesis and provided many valuable suggestions. I am thankful for his support over the years. I also acknowledge the other students, post-docs and visitors with whom I

have interacted at the University of Toronto, including Ryan Adams, James Bergstra, Volha Bryl, George Dahl, Fernando Flores-Mangas, Inmar Govini, Xuming He, Navdeep Jaitly, Nikola Karamanov, Alex Krizhevsky, Dustin Lang, Hugo Larochelle, Scott Leishman, Jenn Listgarten, Ben Marlin, James Martens, Ted Meeds, Roland Memisevic, The Mnih Brothers, Iain Murray, Rama Natarajan, Simon Osindero, Michael Reimer, David Ross, Tanya Schmah, Russ Salakhutdinov, Josh Susskind, Ilya Sutskever, Danny Tarlow, Tijmen Tieleman, Laurens van der Maaten, and Max Volkovs. I apologize to anyone who I have forgotten.

There are many others not connected to U of T who have made a positive and significant impact on my life in the (much larger) world outside the Pratt Building. These include, but are not limited to, the students and residence life staff at 89 Chestnut Residence, the students and staff of Pathways to Education Regent Park, the Sky's the Limit community, and our neighbours on Seaton Street. I will not attempt to mention by name the friends from London and beyond who have been there for me during my tenure in Toronto as I will undoubtedly forget to include many important people.

I would like to thank my parents and sister as well as my "second family", Barb and Renato Natale, for all their support and love.

Finally, I would like to thank Andria Natale, my wife and best friend for not only the good times, but also the sacrifices she made (To quote her: "And I read your thesis; that was a sacrifice.").

This work was supported financially by the Canadian and Ontario Governments, the University of Toronto, the Walter C. Sumner Foundation and the Monica Ryckman Foundation.

Contents

Supporting Videos

While we have made every effort to ensure that this thesis is a self-contained work, it is difficult to emphasize the strong generative ability of the models presented herein without demonstrating visually some examples of synthesis. In Chapters 4-6 we make reference to several videos. They are available at `http://www.cs.toronto.edu/~gwtaylor/thesis/`, organized by chapter.

Notation

Scalar quantities are denoted by lower-case letters, such as x. Vector quantities are denoted by bold lower-case letters, such as $\boldsymbol{x}$. Matrices are denoted by upper-case letters, such as W.

Individual elements of vectors and matrices use subscripts, such as x_i and W_{ij}.

Time is assumed to be discrete and indexed by t. We often use the shorthand notation $\boldsymbol{x}_{1:T}$ to denote sequences of vectors, such as $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_t, \ldots, \boldsymbol{x}_T$.

When we need to index both time and an element of a vector, we separate the implied double-subscript with a comma. For example, $x_{i,t}$ refers to element i of vector $\boldsymbol{x}$ at time t; or equivalently, the ith element of $\boldsymbol{x}_t$.

Occasionally we use the same letter to denote different, but related, vectors or matrices. We use superscripts to distinguish them, as in $W^{\boldsymbol{v}}$ and $W^{\boldsymbol{h}}$.

$\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ is a Gaussian distribution with mean $\boldsymbol{\mu}$ and covariance Σ and $\mathbb{E}_{p^*}$ is an expectation with respect to distribution p^*. We also use the shorthand notation $\langle \cdot \rangle_{p^*}$ to denote an expectation.

1 Introduction

Recent advances in available computing power and statistical models have motivated attempts to capture the regularities of increasingly complex and high-dimensional data. These advances have also fueled the development of nonlinear methods. Much effort has been devoted to modeling static data (i.e. observations that are treated as independently and identically distributed). However, much of the data in the real world has temporal dependencies. Modeling these "time series" involves additional challenges, such as how to capture features of the long-term behaviour of the system, and how to perform inference when these features are complex and connected across time. In this thesis, we explore a family of models designed specifically to address these challenges.

1.1 Distributed, undirected, composable

The simplest time series models, and the earliest studied, contain no hidden variables. Two members of this class of "fully-observed" models are the vector autoregressive model and the N^{th} order Markov model. Though elegant in their construction, these models are limited by their lack of memory. To capture long-range structure they must maintain explicit links to observations in the distant past, which results in an often exponential blow-up in the number of parameters. The strong regularities present in many time series suggest that a more efficient parameterization is possible.

More powerful models, such as the popular Hidden Markov Model (HMM),

introduce a hidden (or latent) state variable that controls the dependence of the current observation on the history of observations. HMMs, however, cannot efficiently model data that is a result of multiple underlying influences since they rely on a single, discrete K-state multinomial to represent the entire history of observations. To model N bits of information about the past, they require 2^N hidden states.

In this thesis, we propose an alternative class of time series models that have three key properties which distinguish them from the prior art. The first property is distributed (i.e. componential) hidden state. Mixture models such as HMMs generate each observation from a single category. Distributed state models (e.g. products) generate each object from a set of features that each contain some aspect of that object's description. Linear Dynamical Systems (LDS) have a continuous, and therefore componential hidden state, but in order to make inference in these models tractable, the relationship between latent and visible variables is constrained to be linear. We show that by carefully choosing the right form of nonlinear observation model it is possible to attain tractable, exact inference, yet retain a rich representational capacity that is linear in the number of components.

Directed acyclic graphical models (or Bayes nets) are a dominant paradigm in models of static data. Their temporal counterparts, Dynamic Bayes nets (Ghahramani, 1998), generalize many existing models such as the HMM and its various extensions. In all but the simplest directed models, inference is made difficult due to a phenomenon known as "explaining away" where observing a child node renders its parents dependent (Pearl, 1988). To perform inference in these networks, typically one resorts to approximate techniques such as variational inference or Monte Carlo methods which have a significant number of disadvantages (Ghahramani, 1998; Murphy, 2002).

An alternative to directed models is to reconsider the causal relationship between variables, and instead focus on *undirected* models. One such model, the Restricted Boltzmann Machine (RBM) (Smolensky, 1986), has garnered recent interest due to its desirable property of permitting efficient exact inference. Unfortunately this comes at a cost: Exact maximum likelihood learning is no longer possible due to the existence of an intractable normalizing constant called the partition function. However, the RBM has an efficient, approximate learning algorithm, contrastive divergence (CD) (Hinton, 2002),

that has been shown to scale well to large problems. RBMs have been used in a variety of applications (Welling et al., 2005; Gehler et al., 2006; Hinton and Salakhutdinov, 2006; Larochelle et al., 2007; Salakhutdinov et al., 2007) and over the last few years their properties have become better understood (Bengio and Delalleau, 2008; Salakhutdinov and Murray, 2008; Sutskever and Hinton, 2008). The CD learning procedure has also been improved (Carreira-Perpinan and Hinton, 2005; Tieleman, 2008; Tieleman and Hinton, 2009). With a few exceptions (Hinton and Brown, 2000; Sutskever and Hinton, 2007) the literature on RBMs is confined to modeling static data. In this thesis, we leverage the desirable properties of an undirected architecture, the RBM, and extend it to model time series. This brings us to the second key property of the models we propose: their observation or emission distribution is an undirected, bipartite graph. This makes inference in our models simple and efficient.

The final key property of our proposed models is that they can form the building blocks of deep networks by incrementally learning one layer of feature extractors at a time. One motivation for promoting deep architectures is biological plausibility. Experimental evidence supports the belief that the brain uses multiple layers of feature-detecting neurons to process rich sensory input such as speech or visual signals (Hinton, 2007a). There is also a practical argument for deep learning. In capturing more abstract, high-level structure from the data, the higher layers provide a more statistically salient representation for tasks such as discrimination. These ideas are not new, but until recently, the problem of how to efficiently train deep networks remained open. The backpropagation algorithm has difficulties with poor local minima and vanishing gradients. A resurgence in the study of deep architectures has been sparked by the discovery that deep networks can be initialized by greedy unsupervised learning of each layer (Hinton et al., 2006). RBMs were originally proposed for this task, but autoencoders (Bengio et al., 2007) and sparse encoder-decoder networks (Ranzato et al., 2006) have also been shown to work. After a pre-training stage, the entire network can be fine-tuned with either a generative or discriminative objective. In this thesis, we show that the two-stage approach to learning deep networks is equally applicable to time series.

1.2 Applications

With the exception of some experiments carried out on binary rainfall occurrence data (Chapter 7) and video textures (Chapter 4), this thesis focuses exclusively on data captured from human motion (mocap). Given its high-dimensional nature, nonlinearities, and long-range dependencies, mocap data is ideal for both studying the limitations of time series models and demonstrating their effectiveness. Our work has benefited from the creation of several large repositories of mocap data, notably the CMU Graphics Lab motion capture database and a more recent collection developed at the Queen's BioMotion Lab. Modeling human motion has also enabled us to extend our work to develop dynamical priors in the context of Bayesian filtering for tracking (Chapter 6). While focusing on a particular domain has facilitated model development and comparison, there is nothing motion-specific to any of the models discussed herein. Therefore, there is no reason to believe that they cannot be applied to other time series.

1.3 Organization

The remainder of this thesis is structured as follows:

Chapter 2 provides a succinct review of time series models. While an exhaustive review of the literature is not possible in this context, we focus on models that relate to our motivations for proposing deep, distributed-state models.

Chapter 3 provides an introduction to motion capture (mocap) data and the change of representation that is a prerequisite for learning good generative models. Readers not interested in modeling human motion can safely skip this chapter.

Chapter 4 introduces the Conditional Restricted Boltzmann Machine (CRBM), a model for time series that has a powerful, distributed hidden state and permits simple, exact inference.

Chapter 5 introduces a model based on the CRBM in which context-related features modulate the connections between two units. The

result is an efficient, compact model which permits control over stylistic properties during synthesis.

Chapter 6 describes how to apply the CRBM as a prior over pose in the context of Bayesian filtering for 3D people tracking. The conditional structure of these models makes them a natural fit for this application.

Chapter 7 reviews Products of Hidden Markov Models (Brown and Hinton, 2001) and demonstrates how the partition function associated with these models can be estimated reliably via Annealed Importance Sampling (Neal, 2001). For the first time, PoHMMs are compared to HMMs and other probabilistic sequence models based on log likelihoods of held-out data. While the PoHMM is related to the RBM-based models because it 1) has distributed hidden state, 2) is based on an undirected observation model, and 3) is trained with contrastive divergence, it is not a conditional model, nor does it have stochastic binary hidden units. This chapter can be skipped without neglecting the flow of the thesis.

Chapter 8 summarizes our contributions and discusses future directions.

2 Background

Time is an integral part of many human behaviours, such as language and movement as well as many natural and man-made phenomena, such as weather patterns and financial returns. Simply ignoring time and applying models of static data to time series disregards much of the rich structure present in the data. In this chapter, we review a variety of statistical models that have been developed to explicitly capture temporal relationships. In modeling time series we face many of the same challenges as modeling static data, such as coping with high-dimensional observations and nonlinear relationships between variates. There also exist many challenges specific to dynamical models, such as permitting invariance under translation in time and perhaps most importantly, capturing long-range dependencies.

Traditionally there are three goals associated with the study of time series (Weigend and Gershenfeld, 1994). The one that has historically received the most attention, is forecasting (or prediction), which aims to accurately predict the short-term evolution of a system. The second is characterization, which asks: how can one, with little or no a priori knowledge of the system, determine its fundamental properties such as number of degrees of freedom, or amount of randomness? The final is modeling: finding a description that accurately captures features of the long-term behaviour of the system. This thesis is focused on modeling.

If the form of the underlying dynamical equations of the system is known, one can attempt to solve for the parameters of these equations. More often than not, systems are complex and their underlying dynamics are not known.

Arriving at an explicit set of dynamical equations to represent or approximate the system is also troublesome. Therefore, one must attempt to determine a set of rules governing the evolution of a system from regularities in past observations. This chapter presents several statistical models that implicitly capture these rules in their parameters. Specifically, we discuss models that aim to capture the regularities of the data in a compact form, and do not need to maintain a complete history of observations. In restricting ourselves to these types of models, we will neglect some interesting and rich non-parametric models (e.g. Gaussian Process Dynamical Models (Wang et al., 2006)) that require us to keep training data at hand to make predictions or synthesize new sequences. We will also not cover recent models such as the Video Epitome (Cheung et al., 2005) that fall somewhere between maintaining a complete history of observations and capturing regularities in a compact form by building and making use of a compressed representation of the training data.

2.1 Models without hidden state

In the probabilistic modeling of sequences, we frequently make use of the fact that the joint probability of a sequence of observations $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$ can always be factored as:

$$p(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T) = p(\boldsymbol{y}_1) \prod_{t=2}^{T} p(\boldsymbol{y}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1}). \tag{2.1}$$

In general, the models discussed in this chapter will seek a more concise form for $p(\boldsymbol{y}_t | \boldsymbol{y}_1, \ldots, \boldsymbol{y}_{t-1})$ such that $\boldsymbol{y}_t$ does not depend on the *entire history* of the time series. This is typically achieved by either restricting the dependencies between variables to a local window in time, or by the introduction of latent variables. We first discuss the former, reviewing models that aim to build a compact representation of the time series by explicitly capturing the dependencies within a recent history of observations; so-called "fully observable" models.

2.1.1 Vector autoregressive models

Prior to the 1920's, the field of time series analysis did not extend beyond extrapolating series by simple curve-fitting methods (Weigend and Gershen-

feld, 1994). The field changed in 1927, when Yule proposed a model for the prediction of sunspots: the sunspot number for a given year is a function of those for the two preceding years and a presumed random disturbance. The introduction of the autoregressive model started a half-century of time series analysis dominated by linear models driven by noise.

Moving average models (also known as Finite Impulse Response filters in the engineering literature) assume that we are given an external input series, $e_{1:T}$, and wish to modify it so as to produce another series, $y_{1:T}$. Assuming linearity and causality of the system, the relationship between the input and output is:

$$y_t = \sum_{n=0}^{N} b_n e_{t-n} = b_0 e_t + b_1 e_{t-1} + \ldots + b_N e_{t-N} \tag{2.2}$$

where N is the order of the model. In short form, this is known as a MA(N) model. MA filters operate in what is called an open loop, that is, they contain no feedback. The name "Finite Impulse Response" follows from the property that N steps after the input becomes zero, the output will also become zero. It may, however, be desirable for the output to continue even after the input ceases. Autoregressive (AR) models (also known as Infinite Impulse Response or IIR filters) operate with feedback and are defined as:

$$y_t = \sum_{n=1}^{N} a_n y_{t-n} + e_t = a_1 y_{t-1} + a_2 y_{t-2} + \ldots + a_N y_{t-N} + e_t \tag{2.3}$$

where, depending on the application, e_t can be a controlled input (as in the MA case) or noise. It is typically assumed to be white, Gaussian noise. This term is crucial to the "life" of an AR model. If it is absent, the series will simply decay to zero, diverge, or oscillate (depending on the coefficients a_n). The multivariate extension to Eq. 2.3 is called the vector autoregressive model (VAR):

$$\boldsymbol{y}_t = \boldsymbol{w} + \sum_{n=1}^{N} \boldsymbol{A}_n \boldsymbol{y}_{t-n} + \boldsymbol{e}_t \tag{2.4}$$

where $\boldsymbol{y}_t$ is a D-dimensional vector. We have introduced $\boldsymbol{w} \in \Re^D$ as a vector of intercept terms that is included to allow for a nonzero mean of the time series. Where $a_1, \ldots, a_N$ in the univariate case were scalar coordinates, $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_N \in \Re^{D \times D}$ are now coefficient matrices of the AR model. As in the univariate case, $\boldsymbol{e}_t$ are typically treated as uncorrelated random vectors with mean zero and covariance matrix $\boldsymbol{R} \in \Re^{D \times D}$.

The coefficients $\boldsymbol{A}_1, \ldots, \boldsymbol{A}_N$ and the intercept, $\boldsymbol{w}$, of a VAR model can be fit simply by treating the estimation as a linear regression from the past N observations to the current observation. The parameters are fit such that the squared difference between the model output and the observed vectors, summed over all training cases, is as small as possible. The covariance matrix of the residual error (under the estimated model) is a reasonable estimator of the noise covariance matrix (Neumaier and Schneider, 2001). As the model order is increased, fitting error over the training set will decrease, but the test error will usually start to increase as the model fits extraneous noise in the system. Model selection is often performed to determine an optimal value of N. This is typically done by optimizing some order selection criterion that penalizes overparameterization.

A major advantage of VAR models is that they are very well understood. If a multivariate time series can be modeled adequately with an AR model, dynamical characteristics of the time series can be examined by structural analyses of the fitted AR model (for example, an eigendecomposition) (Neumaier and Schneider, 2001). On the other hand, they can fail even for simple nonlinearities present in a system. Nonlinearities are responsible for interesting behaviour in many natural dynamical systems. Thus to model these more complicated dynamics we must turn to models whose properties are not as well understood. VAR models also lack internal state, or memory. More powerful models use latent variables to capture relevant features, including long-term dependencies. Though advances in computing since the 1980's have permitted more sophisticated, nonlinear models, vector autoregressive models and their variants remain popular. In addition to their convenient analytical properties, certain time series may in fact be modeled well by a linear model. AR models also fit well into other frameworks, such as a mixtures (Wong and Li, 2000), or as an emission distribution in Hidden Markov Models (see Sec. 2.2.1). Nonlinear models such as those discussed in Chapters 4 and 5 may employ linear autoregressive connections as part of a larger network of connections to capture linear relationships at the input level and allow hidden variables to model nonlinearities.

2.1.2 Markov models

Another simple model that is fully observable is the N^{th} order Markov model. Like the VAR(N) model, each observation depends only on the past N observations, but in contrast to the VAR(N) model, sequential observations may have a nonlinear dependence. The Markov model is derived probabilistically, by assuming that sequences have the Markov property. Formally, this states that, conditioned on the past N observations an observation is independent of all preceding observations:

$$p(\boldsymbol{y}_t|\boldsymbol{y}_1,\ldots,\boldsymbol{y}_{t-1}) = p(\boldsymbol{y}_t|\boldsymbol{y}_{t-N},\ldots,\boldsymbol{y}_{t-1}). \tag{2.5}$$

This leads to the probability of a joint sequence of observations:

$$p(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_T) = p(\boldsymbol{y}_1,\ldots,\boldsymbol{y}_N)\prod_{t=N+1}^{T} p(\boldsymbol{y}_t|\boldsymbol{y}_{t-N},\ldots,\boldsymbol{y}_{t-1}). \tag{2.6}$$

We typically assume that the conditional term in Eq. 2.6 is independent of time, and thus for discrete data, the model is completely parameterized by two probability tables: a table specifying the probabilities of the first N observations, and a table specifying the conditional probabilities $p(\boldsymbol{y}_t|\boldsymbol{y}_{t-N:t-1})$. Maximum likelihood learning is easy in this model because all variables are observed. For the discrete case, the probability tables are formed simply by counting and normalizing.

Markov models are popular in statistical language modeling, where they are known as N-gram models. Special care must be taken when fitting these models by counts, as many of the counts may in fact be zero on the training data set. Many methods have been proposed for smoothing, or regularizing Markov models.

A concern for memoryless models such as the VAR(N) or N^{th} order Markov model is that in order to capture long-range structure, they must maintain explicit links to observations in the distant past (i.e. N must be large). This is especially problematic for the Markov model: the number of parameters of a VAR(N) model is linear in N, but for a discrete N^{th} order Markov model, the number of parameters is exponential in N. One solution is to avoid the use of full conditional probability tables and use a more efficient parameterization. Another solution is to adopt a hidden state, which leads to the Hidden Markov Model.

2.2 Hidden Markov models

Hidden Markov Models (HMMs) are statistical models of sequential data that have proven successful in speech and language modeling (Rabiner, 1989), and more recently, biological sequence analysis (Durbin et al., 1998). The advantage of an HMM over an N^{th} order Markov model is that it introduces a hidden state, s_t, that controls the dependence of the current observation on the history of observations. The hidden variable is a K-state multinomial that has the Markov property. However, we do not assume that the observed data has the Markov property. Typically s_t is first-order Markov since an HMM of first order can emulate any higher order HMM by introducing a higher number of states (Bengio, 1999). It is not necessary for the hidden state to store everything about the past history of the time series - just the information that is relevant to modeling the current state and observation. In this sense, the HMM is compact.

The HMM defines a joint probability distribution over states $s_1, \ldots, s_T$ and observations $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_T$. Due to the independence assumptions defined by the graphical model (Figure 2.1a), the distribution factors as:

$$p(\boldsymbol{y}_{1:T}, s_{1:T}) = p(s_1)p(\boldsymbol{y}_1|s_1)\prod_{t=2}^{T} p(s_t|s_{t-1})p(\boldsymbol{y}_t|s_t). \tag{2.7}$$

The HMM can also be viewed as a mixture model, with temporal coupling between the assignment variables (Roweis and Ghahramani, 1999). The HMM has three sets of parameters:

- The initial state parameters, π, which control the prior distribution of the hidden state at time step $t = 1$.
- The transition matrix, A, which is a stochastic matrix whose elements A_{ij} determine the probability of transitioning to state $s_t = j$ given $s_{t-1} = i$.
- The emission distribution, $p(\boldsymbol{y}_t|s_t)$. The form of the emission distribution depends on the observed data as well as the modeling assumptions. If the observed data is discrete, this typically is another stochastic matrix, B. If the observed data is continuous, this is often taken to be a mixture of Gaussians whose parameters are conditional on the hidden state.

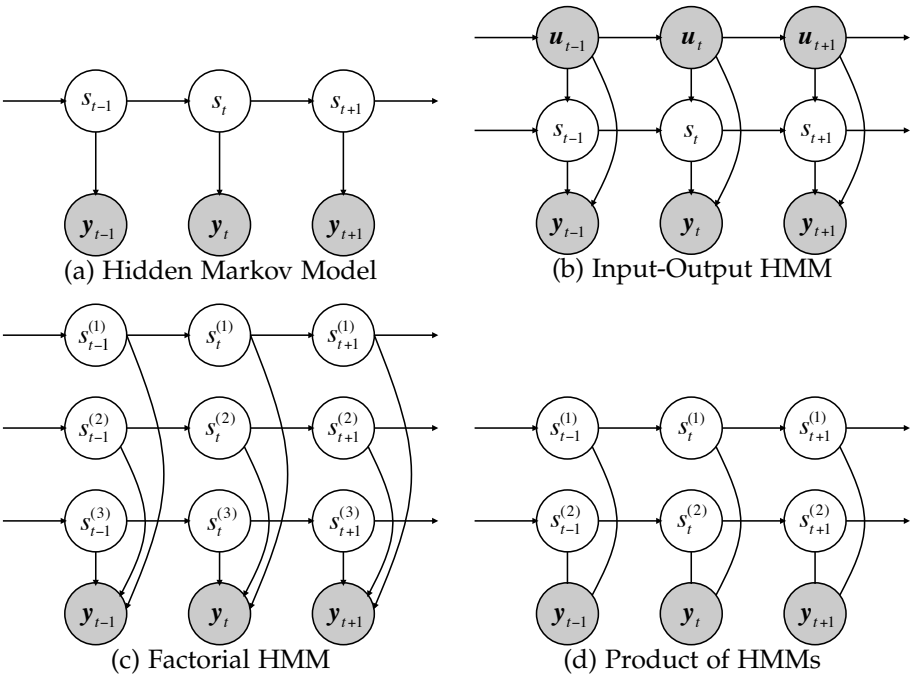


Figure 2.1. Graphical models specifying conditional independence relations for members of the HMM-based family.

Given an observation, or set of observations, there are three basic tasks which one is interested in performing:

1. **Inference:** What is $p(s_t = i|\boldsymbol{y}_{1:T})$, the probability that a model was in state i at time t?

2. **Classification:** What is $p(\boldsymbol{y}_{1:T})$, the probability that a sequence $\boldsymbol{y}_{1:T}$ was generated by this model?

3. **Learning:** Determine the settings of the parameters that maximize the probability of the set of training sequences under the model.

In exchange for making the restrictive assumption of a Markov chain on the discrete hidden state we gain an exact, efficient inference algorithm. This is based on a forward pass through each sequence (filtering) and backward pass (taken together with the forward pass, called smoothing). The forward pass is

defined by the following recursion:

$$\begin{aligned}\alpha(s_t) = p(\boldsymbol{y}_{1:t}, s_t) &= p(\boldsymbol{y}_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}) p(\boldsymbol{y}_{1:t-1}, s_{t-1}) \\ &= p(\boldsymbol{y}_t|s_t) \sum_{s_{t-1}} p(s_t|s_{t-1}) \alpha(s_{t-1}). \end{aligned} \tag{2.8}$$

The forward pass is initialized by setting $\alpha(s_1) = p(s_1, \boldsymbol{y}_1) = p(s_1)p(\boldsymbol{y}_1|s_1)$. Note that the filtering distribution, $p(s_t|\boldsymbol{y}_{1:t})$ can be computed by simply normalizing $\alpha(s_t)$. A similar recursion defines the backward pass:

$$\begin{aligned}\beta(s_t) = p(\boldsymbol{y}_{t+1:T}|s_t) &= \sum_{s_{t+1}} p(\boldsymbol{y}_{t+1}|s_{t+1}) p(s_{t+1}|s_t) p(\boldsymbol{y}_{t+2:T}|s_{t+1}) \\ &= \sum_{s_{t+1}} p(\boldsymbol{y}_{t+1}|s_{t+1}) p(s_{t+1}|s_t) \beta(s_{t+1}). \end{aligned} \tag{2.9}$$

It is initialized by setting $\beta(s_{T+1}) = 1$. The forward and backward passes taken together are known as the forward-backward algorithm for performing inference in HMMs. In practice, the algorithm cannot be implemented exactly as above, because the recursions involve a series of multiplications of small values that can quickly reach numerical underflow. Therefore a scaling step is generally introduced into the algorithm. Details are found in (Rabiner, 1989).

Using Bayes' rule, we obtain the state posterior marginals (i.e. the smoothed estimates of the state distribution):

$$p(s_t|\boldsymbol{y}_{1:T}) = \frac{p(\boldsymbol{y}_{1:t}, s_t) p(\boldsymbol{y}_{t+1:T}|s_t)}{\sum_{s'_t} p(\boldsymbol{y}^t_{1:t'}, s'_t) p(\boldsymbol{y}_{t+1:T}|s'_t)} = \frac{\alpha(s_t)\beta(s_t)}{\sum_{s'_t} \alpha(s'_t)\beta(s'_t)}. \tag{2.10}$$

Similarly, the transition posterior marginals can be obtained from the forward-backward algorithm and the model parameters:

$$p(s_t, s_{t+1}|\boldsymbol{y}_{1:T}) \propto \alpha(s_t) p(s_{t+1}|s_t) p(\boldsymbol{y}_t|s_t) \beta(s_{t+1}). \tag{2.11}$$

The probability of a sequence under the model can be computed by marginalizing out s_T from the final step of the forward pass:

$$p(\boldsymbol{y}_{1:T}) = \sum_{s_T} \alpha(s_T) = p(\boldsymbol{y}_{1:T}, s_T). \tag{2.12}$$

In many applications, the HMM state variable is associated with some meaning. Therefore it is often useful to infer the most likely state sequence, $s_{1:T}$, given the observations, $\boldsymbol{y}_{1:T}$. This is achieved via the Viterbi algorithm, which

essentially substitutes the max operation for the sum operation in the forward-backward algorithm. Just as the forward-backward algorithm is a special case of belief propagation in singly connected networks, the Viterbi algorithm generalizes to the max-product algorithm.

The parameters of an HMM are typically learned by a special case of the EM algorithm called the Baum-Welch algorithm. The "E-step" consists of performing the forward-backward algorithm to update the state and transition posteriors. The exact form of the parameter updates done in the "M-step" depend on the particular form of emission distributions. If these distributions are discrete, or members of the exponential family, exact "M-step" updates can be derived by taking the gradient of the expected complete log likelihood with respect to the parameters and setting the result to zero. If the distributions are more complicated, then one may have to resort to numerical methods such as gradient descent.

2.2.1 Input-Output HMMs

Input-Output Hidden Markov Models (IOHMMs) (Bengio and Frasconi, 1995) have transition and emission distributions which are conditional on another sequence, called the input sequence: $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_L$. For this reason, they are also known as Conditional HMMs. In IOHMMs, the observations modeled with the emission distributions are called the outputs. The model defines a probability distribution over outputs conditional on the inputs, $p(\boldsymbol{y}_{1:T}|\boldsymbol{u}_{1:L})$. Figure 2.1b shows the graphical model for an IOHMM.

IOHMMs where the length of the input sequence matches the length of the output sequence ($L = T$) are called *synchronous* IOHMMs, and for these models, learning is similar to learning in HMMs. Details are given in (Bengio and Frasconi, 1995). In applications such as speech recognition, the input and output sequences may not be the same length. The IOHMM model has been generalized to this *asynchronous* case (Bengio and Bengio, 1996).

The standard HMM is said to be *homogeneous*, since its transition and emission distributions do not change over time. The IOHMM, on the other hand, is *heterogeneous*, due to the fact that these distributions are conditional on a sequence that changes over time. IOHMMs typically use complex nonlinear emission and transition distributions based on neural networks and rely on gradient-based parameter updates. While in many situations it is conceivable

to use either a HMM or IOHMM to model a particular time series, there are several potential advantages to employing an IOHMM. Bengio (1999) gives a complete list, which we summarize here:

- The emission and transition densities of an IOHMM are typically represented by complex nonlinear models (like neural networks) which may be more powerful than the mixture of Gaussians typically employed by HMMs. These local models can also take into a wider context (such as neighbouring observations in the sequence) without violating the Markov assumption.
- We can define IOHMMs such that the output variable is discrete, taking a small number of values, where the input variable is multivariate and real. Having a simpler emission distribution reduces the problem of imbalance between the importance of the emission and transition distributions in an HMM (which can occur when using complicated emission distributions). This approach is typically used in speech applications, where the input represents an acoustic feature vector and the output represents phonemes or sub-phonemes.
- Long-term dependencies are more easily learned in an IOHMM, as the state variable is less likely to "mix" and forget past contexts.
- Even in situations where there is no obvious input/output relationship (for example, modeling a simple time series, $\boldsymbol{y}_{1:T}$) one may include as input variables transformations of the observation, $f(\boldsymbol{y}_{1:T})$. The emission distribution would have the conditional form $p(\boldsymbol{y}_t|s_t, f(\boldsymbol{y}_{1:t-1}))$. In theory, the model would require less hidden states, as the input $f(\boldsymbol{y}_{1:T})$ already provides useful summary information about the past sequence.

The models presented in Chapters 4 and 5 are similar to IOHMMs in that they use conditional distributions in modeling both dynamics and observations.

2.2.2 Factorial HMMs

In Chapter 1, we claimed that in order to efficiently model data that has componential structure we need distributed (i.e. componential) hidden state. Factorial Hidden Markov Models (FHMMs) were introduced by Ghahramani and Jordan (1997) to address the need for distributed hidden state in HMMs.

The FHMM generalizes the HMM by using a collection of M state variables, rather than a single discrete state: $s_t = s_t^{(1)}, \ldots, s_t^{(m)}, \ldots, s_t^{(M)}$ (Figure 2.1c). Each of the states can take on $K^{(m)}$ values, but for simplicity, we will assume that $K^{(m)} = K, \forall m$. The extension to differing $K^{(m)}$ is trivial. The state space of this model is the cross-product of all of these state variables, thus making no constraints on the state transition structure would result in a $K^M \times K^M$ transition matrix. Not only is this simply equivalent to a giant HMM with K^M states, the time and space complexity of the inference algorithm would be exponential in M.

Therefore, the authors considered a constrained FHMM in which each state variable evolves according to its own dynamics:

$$p(s_t|s_{t-1}) = \prod_{m=1}^{M} p(s_t^{(m)}|s_{t-1}^{(m)}), \tag{2.13}$$

and so the transition structure of this model can be parameterized by M $K \times K$ transition matrices. Unfortunately, even in this version of the model where the transition structure corresponding to each of the HMM chains is decoupled from one another, inference is still intractable since observing a sequence introduces dependencies between the chains. This is a case of "explaining away". Ghahramani and Jordan proposed that inference be performed either by Gibbs sampling or by one of two variational approximations to the posterior distribution.

Variational methods approximate the true posterior with a simpler, tractable distribution, $Q(s_{1:T})$. The simplest approximation is a completely factorized distribution: all state variables are assumed independent given the observations. This type of approximation is often used in physics, where it is known as the *mean field* approximation to statistical mechanical systems. Another variational technique applied by Ghahramani and Jordan is a *structured variational approximation* (Saul and Jordan, 1996). In this case, the approximate posterior is not only tractable but preserves many of the original dependencies in the true posterior. The structured approximation for a FHMM decouples the M HMMs, so that within each HMM the forward-backward algorithm can be applied independently and exactly. In applying either of the two simpler distributions, *variational parameters* are introduced. These parameters are updated by following the negative gradient with respect to the Kullback-Leibler (KL)

divergence between Q and the true posterior, P:

$$\mathrm{KL}(Q||P) = \sum_{s_t} Q(s_{t:T}) \log \left[\frac{Q(s_{1:T})}{P(s_{1:T}|\boldsymbol{y}_t)} \right]. \tag{2.14}$$

Learning is done via a generalized EM algorithm which optimizes a lower bound on the data log likelihood. The "E-step" consists of iteratively updating the variational parameters until convergence has been reached (by monitoring KL divergence), then updating the usual expectations required that will now depend on the variational parameters. As in standard HMMs, the "M-step" is simple and tractable, consisting of closed-form updates to the model parameters.

The choice between the approximate inference schemes must take into account both theoretical and practical issues (Ghahramani and Jordan, 1997). Markov chain Monte Carlo methods offer a theoretical guarantee that the sampling procedure will converge to the correct posterior in the limit, but tend to be slow. Variational approximations offer a theoretical assurance that a lower bound on the likelihood is being maximized but optimization of the KL-divergence function is subject to being caught in local minima. It is also difficult to assess how well the variational distribution approximates the true posterior. The structured variational approximation seems to be preferable to the completely factorized approximation in that much of the structure of the true posterior is preserved, with no loss of tractability. A major issue with factorial HMMs is that computing the probability of a sequence under the model is intractable. This makes model comparison difficult. One can, however, compute a lower bound on the log-likelihood using either Gibbs sampling or variational inference.

2.2.3 Products of HMMs

Products of Hidden Markov Models (PoHMM) (Brown and Hinton, 2001) are the undirected analogue to Factorial HMMs. They are a member of the Product of Experts family (Hinton, 2002) where each of the constituent experts is an HMM. Like the FHMM, the model provides a distributed state representation, but the relationship between hidden state and observations is no longer causal (Figure 2.1d). Conditioned on a sequence of observations, each HMM in the product is independent. Generating a sequence, however, is

not as easy as in a FHMM since the HMMs are unconditionally dependent. Chapter 7 discusses PoHMMs in depth.

2.3 Continuous state-space models

HMMs permit tractable inference and learning, as well as arbitrary non-linear relationships between the hidden state and observations. The primary limitation of these models is their simple discrete state representation. A related family of models, known as state-space models, make the same independence assumptions as HMMs and therefore specify an identical joint distribution over states and observations (Eq. 2.7). The only difference is that an HMM uses a discrete state variable, s_t, and state-space models use a real-valued state vector, $\boldsymbol{x}_t$:

$$p(\boldsymbol{y}_{1:T}, \boldsymbol{x}_{1:T}) = p(\boldsymbol{x}_1)p(\boldsymbol{y}_1|\boldsymbol{x}_1)\prod_{t=2}^{T} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{y}_t|\boldsymbol{x}_t). \tag{2.15}$$

The graphical model for a state-space model is identical to Figure 2.1a (replacing s_t by $\boldsymbol{x}_t$). Typically state-space models place some type of simplifying restriction on the emission and transition distributions (as opposed to a HMM which allows arbitrary distributions). We first discuss the most commonly treated member of this family, the Linear Dynamical system (LDS), which employs linear-Gaussian dynamics and a linear-Gaussian emission distribution.

2.3.1 Linear Dynamical Systems

Linear Dynamical Systems are described by the following two equations:

$$\boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + \boldsymbol{w}_t, \tag{2.16}$$

$$\boldsymbol{y}_t = C\boldsymbol{x}_t + \boldsymbol{v}_t \tag{2.17}$$

where the state and output noise, $\boldsymbol{w}_t$ and $\boldsymbol{v}_t$, are zero-mean Gaussian distributed random variables with covariance matrices Q and R, respectively. Note that the output, $\boldsymbol{y}_t$ is a linear function of the state, $\boldsymbol{x}_t$ and this current state depends linearly on the state at the last time step, $\boldsymbol{x}_{t-1}$. The linear-Gaussian assumptions permit tractable inference.

Note that both Eq. 2.16 and Eq. 2.17 contain both a deterministic transition function and a noise vector. These can be combined into conditional densities

for both the state and output, similar to how we have defined the HMM:

$$p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(A\boldsymbol{x}_{t-1}, Q), \tag{2.18}$$

$$p(\boldsymbol{y}_t|\boldsymbol{x}_t) = \mathcal{N}(C\boldsymbol{x}_t, R). \tag{2.19}$$

The initial state density is also Gaussian:

$$p(\boldsymbol{x}_1) = \mathcal{N}(\boldsymbol{\pi}_1, V_1). \tag{2.20}$$

Substituting Eq. 2.18, Eq. 2.19, and Eq. 2.20 into Eq. 2.15 and taking the logarithm, we have a joint log probability that is the sum of quadratic terms:

$$\begin{aligned}\log p(\boldsymbol{y}_{1:T}, \boldsymbol{x}_{1:T}) = &- \sum_{t=1}^{T} \left(\frac{1}{2} \left[\boldsymbol{y}_t - C\boldsymbol{x}_t\right]' R^{-1} \left[\boldsymbol{y}_t - C\boldsymbol{x}_t\right] \right) - \frac{T}{2} \log |R| \\ &- \sum_{t=2}^{T} \left(\frac{1}{2} \left[\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}\right]' Q^{-1} \left[\boldsymbol{x}_t - A\boldsymbol{x}_{t-1}\right] \right) - \frac{T-1}{2} \log |Q| \\ &- \frac{1}{2} \left[\boldsymbol{x}_1 - \boldsymbol{\pi_1}\right]' V_1^{-1} \left[\boldsymbol{x}_1 - \boldsymbol{\pi_1}\right] - \frac{1}{2} \log |V_1| + \text{const.} \end{aligned} \tag{2.21}$$

Similar to the IOHMM, there may be another real-valued input vector, $\boldsymbol{u}_t$, whose value is observed. In this case, the goal is to model the output of the system conditional on the input. The state dynamics would then become:

$$\boldsymbol{x}_t = A\boldsymbol{x}_{t-1} + B\boldsymbol{u}_t + \boldsymbol{w}_t \tag{2.22}$$

where B is an input matrix that relates $\boldsymbol{u}_t$ linearly to $\boldsymbol{x}_t$. For the remainder of our discussion, we will consider the case of the output-only LDS, but the extension to include inputs is straightforward (Ghahramani and Hinton, 1996).

In an LDS, only the $\boldsymbol{y}_{1:T}$ are observed. The $\boldsymbol{x}_{1:T}$ and noise are unobserved. Therefore, as in other latent variable models, we cannot perform maximum likelihood. Shumway and Stoffer (1982) showed that the expectation-maximization (EM) algorithm could be applied to estimate the parameters of an LDS. EM alternates between updating the model parameters given the expected sufficient statistics under the posterior distribution (the "M-step") and computing these statistics given the current parameter estimates (the "E-step"). The "M-step" is a series of closed-form updates, obtained by taking the gradient of the complete data log-likelihood Eq. 2.21 with respect to the parameters, and using expectations under the posterior in place of the actual hidden state. The update rules are summarized in (Ghahramani and Hinton, 1996).

In order to compute the parameters in the "M-step" we need to somehow estimate statistics of the hidden state of a sequence at time t, given the sequence of observations. For the case of the HMM, we discussed a forward and backward recursion to compute the necessary statistics. The analogous method for LDS is called Kalman smoothing (or Rauch-Tung-Streibel smoothing). Filtering alone (or simply, the forward pass) is known as Kalman filtering. It is the fact that both the transition and emission distribution are in the exponential family (and specifically the property of closure under multiplication) that allows for a tractable inference algorithm. At each step in the forward and backward recursion, multiplication by the transition and emission densities does not make the recursive functions more complex, but only changes their parameters. Like the forward-backward algorithm for HMMs, Kalman smoothing is a special case of the belief propagation algorithm for directed graphical models. The equations for the Kalman smoother can be derived by analytically evaluating the Gaussian integrals that result from applying belief propagation to the state-space graphical model. More details are found in (Minka, 1999).

2.3.2 Nonlinear estimation

Because of the linear-Gaussian assumption, probabilistic inference in an LDS can be carried out tractably and exactly by the Kalman recursions. However, for many real-world problems that require nonlinear distributions, Kalman filtering (or belief propagation in general) is intractable and we must apply approximations.

For many years, the Extended Kalman Filter (EKF) has been the dominant technique for approximate inference in state-space models that do not hold to the linear-Gaussian assumption. This algorithm approximates the state distribution by a Gaussian random variable. It then linearizes the system equations by a Taylor series expansion about the the mean of this random variable. The statistics (mean and covariance) needed by the EKF are taken from the first-order truncation of this series. The EKF essentially makes a first-order approximation to the optimal terms which are analytically intractable to evaluate. Despite being a widely used technique, the EKF suffers from two major flaws. Expanding about a single point in the Taylor series disregards the uncertainty around the random variables representing system state, process

and system noise. Furthermore, the accuracy of the solution is affected by neglecting the higher order terms. A family of algorithms called Sigma-Point Kalman filters (van der Merwe, 2004) generalizes two popular extensions of the EKF, the Unscented Kalman Filter (UKF) and the Central Difference Kalman Filter (CDKF). These methods are based on deterministic sampling to linearize the underlying system and, in practice, calculate state estimates that are more accurate than the EKF.

2.3.3 Particle filtering

We now introduce sequential Monte Carlo methods (more commonly known as particle filtering) as another variant for approximate inference in continuous state-space systems. In general, they can be applied to the problem of approximating optimal Bayesian inference in tree-structured graphical models when the model distributions are non-Gaussian, nonlinear and non-stationary. While at the heart of the standard EKF and SPKF are Gaussian assumptions on the relevant distributions, particle filters make no such assumptions and therefore can support multi-modal posterior distributions.

Particle filtering is based on a Monte Carlo method called sequential importance sampling (SIS). This is the recursive form of a well-known technique called importance sampling. The idea of importance sampling is to represent a distribution $p(\boldsymbol{x})$ by an empirical approximation based on a set of weighted samples (called particles):

$$p(\boldsymbol{x}) \approx \sum_{p=1}^{P} w^{(p)} \delta\left(\boldsymbol{x} - \boldsymbol{x}^{(p)}\right) \tag{2.23}$$

where $\delta(\cdot)$ is the Dirac delta function. Members of the weighted sample set, $\{\boldsymbol{x}^{(p)}, w^{(p)}\}_{p=1}^{P}$, are drawn from a proposal distribution $\pi(\boldsymbol{x})$ that is both related to the distribution of interest and easy to sample. The importance weights are given by:

$$w^{(p)} = \frac{p\left(\boldsymbol{x}^{(p)}\right) / \pi\left(\boldsymbol{x}^{(p)}\right)}{\sum_{p'=1}^{N} p\left(\boldsymbol{x}^{(p')}\right) / \pi\left(\boldsymbol{x}^{(p')}\right)}. \tag{2.24}$$

Statistics of interest, such as $E[\boldsymbol{x}]$, follow easily from Eq. 2.23. The basic idea behind SIS is to use the first-order Markov nature of the state transitions and the conditional independence of the observations given the state to

derive a recursive update formula for the weights at each time-step t. More details are given in Chapter 6. The point-mass estimate of the posterior filtering distribution can approximate any nonlinear distribution arbitrarily well, provided that the number of particles is sufficient, and certain sampling conditions are met (Doucet et al., 2000). Perhaps the greatest challenge to particle filtering is that that the variance of the importance weights is guaranteed to increase over time. As a result, typically the weight of one particle will eventually reach unity while the others tend towards zero. A variety of resampling methods have been proposed to address this issue.

2.3.4 Switching mixtures of Linear Dynamical Systems

So far we have discussed models with discrete dynamics (Sec. 2.2) and models with continuous dynamics (Sec. 2.3). Many models have been proposed in disciplines from control systems to econometrics that allow for a combination of the two. They have gone by several names, including hybrid models, state-space models with switching, and jump-linear systems. We review a particular model, the Switching Linear Dynamical System (SLDS), which combines and generalizes both the HMM and the LDS. While several variants of SLDS have been proposed (Pavlovic et al., 2001; Li et al., 2002), we describe the formulation in Ghahramani and Hinton (2000).

State-space models are typically based on a single, real-valued state vector. The SLDS generalizes these methods to multiple real-valued state vectors, $\boldsymbol{x}_t^{(m)}, m = 1, \ldots, M$. The SLDS also has a multinomial discrete state, s_t, (called the switch variable) that takes on one of M values. Conditioned on the setting of the switch state, $s_t = m$, the observed variable is multivariate Gaussian with the parameters determined by the setting of m. The joint probability of observations and latent variables factorizes into:

$$p(\boldsymbol{y}_{1:T}, s_{1:T}, \boldsymbol{x}_{1:T}^{(1)}, \ldots, \boldsymbol{x}_{1:T}^{(M)}) = p(s_1) \prod_{t=2}^{T} p(s_t|s_{t-1}) \prod_{m=1}^{M} p(\boldsymbol{x}_1^{(m)}) \prod_{t=2}^{T} p(\boldsymbol{x}_t^{(m)}|\boldsymbol{x}_{t-1}^{(m)}) \times \prod_{t=1}^{T} p(\boldsymbol{y}_t|\boldsymbol{x}_t^{(1)}, \ldots, \boldsymbol{x}_t^{(M)}, s_t). \quad (2.25)$$

The probability of an observation, $\boldsymbol{y}_t$ is:

$$p(\boldsymbol{y}_t|\boldsymbol{x}_t^{(1)},\ldots,\boldsymbol{x}_t^{(M)},s_t=m) = (2\pi)^{-\frac{D}{2}}|R|^{-\frac{1}{2}} \times \exp\left\{-\frac{1}{2}\left(\boldsymbol{y}_t - C^{(m)}\boldsymbol{x}_t^{(m)}\right)^T R^{-1}\left(\boldsymbol{y}_t - C^{(m)}\boldsymbol{x}_t^{(m)}\right)\right\} \quad (2.26)$$

where D is the dimension of the observation, R is the (tied) covariance matrix, and $C^{(m)}$ is the output matrix for state-space model m. Each state-space model follows its own dynamics, as specified by Eq. 2.18, and the discrete state, s_t evolves according to the simple dynamics of an HMM.

The SLDS can also be viewed as a mixture of experts (Jacobs et al., 1991), where the individual experts are the M state-space models, which are gated by a switch, s_t, that has first-order Markov dynamics. Unlike simpler models like the LDS and HMM, the SLDS cannot be trained with the exact EM algorithm. EM requires that inference be tractable (for the E-step) but observing a sequence couples the real-valued state variables and the discrete switch variables together such that the posterior distribution over latent variables is a Gaussian mixture with M^T terms.

Similar to the inference method employed in the Factorial HMM, Ghahramani and Hinton (2000) propose to relax the EM algorithm via a structured variational approximation. They approximate the true posterior with a simple, tractable distribution, $Q(s_{1:T}, \boldsymbol{x}_{1:T})$ which eliminates some of the dependencies. As in the case of the FHMM, the tractable substructures of the original network are preserved, and so exact inference algorithms (in this case, the forward-backward and Kalman smoothing algorithms) can be performed within the relevant substructures.

The "E-step" proceeds by iteratively updating the variational parameters and computing the sufficient statistics of the latent variables under the approximate posterior. As in standard EM, the "M-step" recomputes the model parameters by optimizing the expected complete log likelihood with respect to those parameters. These updates end up being responsibility-weighted versions of the original updates (for the LDS parameters) and Baum-Welch updates (for the discrete Markov chain involving the switch variable). This generalized EM algorithm is guaranteed to optimize a lower bound on the log probability of the observed data.

Experimentally, the SLDS performs equivalently (in terms of the likelihood of test data under the model) to HMMs with an equivalent number of param-

eters. However, this means that the HMM has more discrete states, and the ability of the SLDS to "segment" sequences into those for which each setting of the switch state is responsible may mean that the SLDS leads to a more intuitive explanation of the data. However, a major downside of switching models is that it becomes difficult to maintain continuity of the real-valued states at switch times, and this can pose a problem in data generation or the modeling of transitions between modes (for example, modeling walking to running transitions in human motion).

2.3.5 Capacity of continuous state-space models

In Chapter 1 we pointed out that a Hidden Markov Model with K states could only communicate $\log_2 K$ bits of information about the past to the future. How does this compare to models with real-valued state, such as Linear Dynamical Systems? At first glance, it seems that models with real-valued state could communicate infinite information into the future. In practice, however, the precision of the real-values communicated is limited by hardware and furthermore, the LDS contains two noise processes which are essential components of the model. Because the hidden state of an LDS is essentially an autoregressive model, we can make the same argument as we did in 2.1.1: In the absence of process noise, the hidden state would either decay exponentially towards zero, or explode exponentially in the direction of the leading eigenvector of the transition matrix. In the absence of observation noise, the state would no longer be hidden and the model would not be an LDS.

These essential noise processes have an effect on the information that can be communicated into the future, and the nature of this effect can be explained by coding theory. As the temporal links in an LDS are limited to just the hidden states, we will consider the dynamics model alone. Additionally, for ease of presentation, we will consider the case of first-order single-state dynamics. In coding theory, the dynamics correspond to a Gaussian channel. It has a real-valued input, x (the signal), and a real-valued output, x' (the signal plus noise). The noise model is zero-mean Gaussian with variance σ^2. Relating this to the LDS, x and x' correspond to the state at $t-1$ and t, respectively. The *capacity* of the channel measures the maximum amount of error-free information that can be transmitted over the channel per unit time.

The optimal capacity is achieved when the signal, x, is Gaussian-distributed with variance ν, where ν is the average power of the signal. Note that this is consistent with the LDS as its hidden state is always Gaussian[†]. The optimal capacity is given by

$$C = \frac{1}{2}\log_2\left(1 + \frac{\nu}{\sigma^2}\right) \tag{2.27}$$

where $\frac{\nu}{\sigma^2}$ is the signal-to-noise ratio (SNR).

Thus the capacity, which is a function of the SNR, gives a sense of how many bits of information can be communicated into the future by the real-valued state. The capacity can similarly be defined for multiple "parallel" channels corresponding to multiple state variables, but the derivation is considerably more involved. Its form is similar to Eq. 2.27 but it is a function of the noise power spectrum. We refer the reader to (Cover and Thomas, 2006) for more details.

2.4 Neural networks

The literature on neural networks has shown that simple architectures are capable of discovering useful and interesting internal representations of static data. Although the parallel nature of computation in neural networks may seem at odds with the serial nature of temporal events, there have been many attempts to extend such models to temporal data. One of the simplest ways of temporally extending neural networks is to "parallelize time" by giving it a spatial representation. However, there are several downsides to this approach. First, we assume that some buffer exists, collecting the input so that it can be presented to the network all at once. Such a buffer is not evidenced in biological systems, nor does it permit online processing. The second downside to spatial representations of time is the inability for the network to process input vectors of varying length. This is especially troublesome for domains like speech or language. Finally, parallelizing time does not permit the network to distinguish between absolute and relative temporal position. This is a serious limitation to a model's capability to generalize. These issues motivate the implicit representation of time in a neural network, where time is represented by an effect on processing and not as an extra dimension of the input. In

[†]When we apply a linear transformation and apply Gaussian noise to a Gaussian-distributed quantity it is still Gaussian.

effect, this gives the network a memory. In this section, we review several interesting ways of implicitly incorporating time into neural networks.

2.4.1 Elman networks

Elman (1990) presented a simple temporal extension to a feed-forward network with a single hidden layer that is closely related to an earlier model by Jordan (1986). A standard feed-forward network is augmented at the input layer with a series of "context units". The hidden layer is connected to these context units such that they represent the hidden layer activations at the previous time step. Both the input and context units are fully connected to the hidden layer. The Elman network is shown in Figure 2.2.

In learning the weights of such a network, the input-to-hidden and context-to-hidden weights as well as the hidden-to-output weights are updated. The hidden-to-context weights are never updated, such that the context units always represent a time-delayed copy of the hidden layer activations. Given a time series, the network is trained by back-propagation to predict the next step given the current step. In contrast to a traditional feed-forward network, where hidden units develop representations of the input patterns to aid in predicting the correct output, the internal representations of an Elman network are sensitive to temporal context. Experimentally, the trained networks were

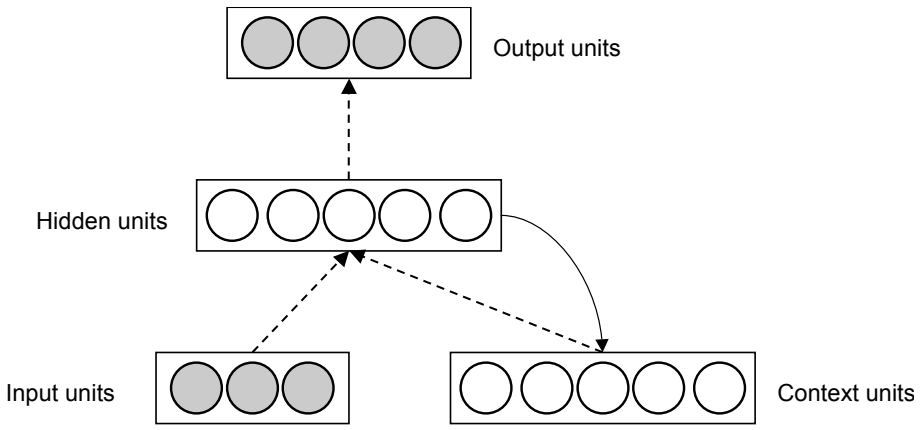


Figure 2.2. Elman network. The dashed connections are learned. The solid connection indicates copying the hidden units to the context units at the next time step.

shown to learn interesting structure in simple language sequences. Hierarchical clustering of internal representations showed that the network structured the available high-dimensional latent space in such a way that important relations between lexical items were translated into spatial relationships. The hierarchies that develop are implicit in the structure of the hidden unit activations, and do not need to be specified *a priori* in terms of network structure or depth. This is a consequence of using distributed representations, which make available this space for structuring.

The disadvantage of Elman networks is their limited feed-forward structure. If the task in which we are interested is truly predicting the next step, then such an architecture may be acceptable. However, for tasks such as synthesizing novel sequences or scoring time series, it is preferable to build a generative model of time series (like the models presented in Chapters 4 and 5). Generative models can also be used for prediction.

2.4.2 Time-delay neural networks

Another serious limitation of Elman networks is that they do not contain any long-term connections to the past. A connection is made to the most recent hidden state, with the hope that the network will serendipitously utilize these connections to capture the entire history of the time series. Furthermore, Elman uses a truncated form of the backpropagation through time algorithm (BPTT), such that only standard backpropagation is applied at each time step. The Time-Delay Neural Network (TDNN) (Waibel et al., 1989; Wan, 1994) addresses these issues: it permits arbitrarily-long connections backwards in time, and it is trained by the full BPTT algorithm.

The TDNN is able to represent temporal relationships between inputs while allowing for invariance under translation in time. This is achieved by incorporating time into the architecture of the network in the form of tapped delay lines. The basic unit used in many neural networks computes the weighted sum of its inputs and typically passes this result through a nonlinear activation function, such as a sigmoid. In a TDNN, this basic unit is modified by introducing N delays for each of its J inputs, and maintaining a series of weights to each of the J inputs, at each of N delays (the weights to the current input are also maintained). As in a typical feed-forward network, multiple layers of hidden units can be used, and the number of delay taps

can be different at each layer. For example, higher level units can have more delay taps so as to learn longer-range dependencies, whereas lower levels can act more locally. The TDNN can be unfolded in time by removing all time delays to expand the network into a larger equivalent static network that is constrained by imposed symmetries. The TDNN can thus be seen as a compact representation of a larger static network.

Unfortunately a major challenge of training TDNNs is that the gradients eventually disappear as they are propagated back in time. This is simply a consequence of multiplying partial derivatives. Training is also computationally expensive, though the recent popularity and availability of parallel computing environments may rekindle interest in such models.

2.4.3 Temporal Boltzmann Machines

Networks using time-delay architectures are less successful at handling longer duration events that require something equivalent to "time warping" in order to match learned representations to the data (Williams and Hinton, 1990). One way to achieve such behaviour is to temporally extend a powerful static model by replicating it through time and then sharing weights. Several authors have proposed to temporally extend some form of a Boltzmann Machine (Hinton and Sejnowski, 1987). This is of particular relevance to this thesis, as the models presented in Chapters 4 and 5 are extensions of Boltzmann machines. We briefly depart from the discussion of time series models to review two models of static data.

Boltzmann Machines

The Boltzmann machine is a collection of symmetrically connected, neuron-like stochastic binary units (Figure 2.3a). Each unit decides whether it is on or off by considering the total input coming to it from all of the other units. The probability that unit i turns on is given by the logistic function

$$p(s_i = 1 | \{s_j\}, j \neq i) = \frac{1}{1 + \exp(-b_i - \sum_j W_{ij} s_j)} \tag{2.28}$$

where W_{ij} is the weight on the connection between unit i and unit j, b_i is the bias of unit i and $s_j = 1$ if unit j is on and $s_j = 0$ otherwise.

Given a training set of state vectors, the weights and biases in a Boltzmann machine can be adjusted, or learned, to assign high probability to vectors in

the training set. The probability of any given state vector, $\boldsymbol{v} = \{s_i\}$, is given by

$$p(\boldsymbol{v}) = \frac{\exp\left(-E(\boldsymbol{v})\right)}{\sum_{u} \exp\left(-E(\boldsymbol{u})\right)} \tag{2.29}$$

where $E(\boldsymbol{v})$ is the "energy" of state vector v, defined as

$$E(\boldsymbol{v}) = -\sum_{i} b_i s_i - \sum_{i<j} W_{ij} s_i s_j. \tag{2.30}$$

Eq. 2.29 is the well-known Boltzmann distribution and it leads to a very simple maximum likelihood learning rule (Hinton and Sejnowski, 1987):

$$\Delta W_{ij} \propto \langle s_i s_j \rangle_{\text{data}} - \langle s_i s_j \rangle_{\text{model}} \tag{2.31}$$

where $\langle \cdot \rangle_{\text{data}}$ is an expectation with respect to the data distribution and $\langle \cdot \rangle_{\text{model}}$ is an expectation with respect to the model's equilibrium distribution[†]. While we do not know the true distribution of the data, we can estimate $\langle \cdot \rangle_{\text{data}}$ empirically from our training set. The update rule for the biases is simply

$$\Delta b_i \propto \langle s_i \rangle_{\text{data}} - \langle s_i \rangle_{\text{model}}. \tag{2.32}$$

These learning rules describe how to perform gradient ascent in the log probability that the Boltzmann machine would generate the observed data when sampling from its equilibrium distribution. In practice, we increment the current W_{ij} or b_i by a small learning rate times the right-hand side of Eq. 2.31 or 2.32, respectively.

The units in a Boltzmann machine can be partitioned into two subsets: "visible" units, whose states can be observed and "hidden" units which correspond to latent, unobserved features. The partitioned Boltzmann Machine is shown in Figure 2.3b. The hidden units can capture higher-order structure not modeled by the lateral connections and the partitioning does not change the learning rule. However, given a setting of the visible units, the units must be updated repeatedly until the model reaches equilibrium. The expectations in Eq. 2.31 depend on this inference step, and so learning is very slow.

[†]If the units are updated sequentially in an order that does not depend on their total inputs, the network will eventually reach the distribution defined by Eq. 2.29. This is the network's equilibrium distribution.

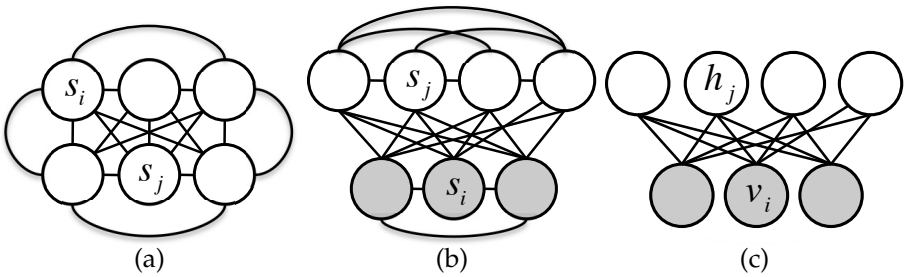


Figure 2.3. a) A Boltzmann Machine. b) A Boltzmann Machine partitioned into visible (shaded) and hidden units. c) A Restricted Boltzmann Machine.

Restricted Boltzmann Machines

The Restricted Boltzmann Machine (Smolensky, 1986) is a Boltzmann Machine with a special structure (Figure 2.3c). It has a layer of visible units fully connected to a layer of hidden units but no connections within a layer. This bipartite structure ensures that the hidden units are conditionally independent given a setting of the visible units and vice-versa. Simplicity and exactness of inference are the main advantages to using an RBM compared to a fully connected Boltzmann Machine.

To make the distinction between visible and hidden units clear, we use v_i to denote the state of visible unit i and h_j to denote the state of hidden unit j. We also distinguish biases on the visible units, a_i from biases on the hidden units, b_j. The RBM assigns a probability to any joint setting of the visible units, $\boldsymbol{v}$ and hidden units, $\boldsymbol{h}$:

$$p(\boldsymbol{v}, \boldsymbol{h}) = \frac{\exp\left(-E(\boldsymbol{v}, \boldsymbol{h})\right)}{Z} \tag{2.33}$$

where $E(\boldsymbol{v}, \boldsymbol{h})$ is an energy function,

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{ij} W_{ij} v_i h_j - \sum_i a_i v_i - \sum_j b_j h_j \tag{2.34}$$

and Z is a normalization constant called the partition function, whose name comes from statistical physics. The partition function is intractable to compute exactly as it involves a sum over the (exponential) number of possible joint configurations:

$$Z = \sum_{v', h'} E(\boldsymbol{v}', \boldsymbol{h}'). \tag{2.35}$$

Maximum likelihood learning in an RBM does not change from Eq. 2.31 which we rewrite, noting that connections only exist *between* the visible and hidden units, and not within a layer:

$$\Delta W_{ij} \propto \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}. \tag{2.36}$$

Because of the conditional independence properties of the RBM, we can easily obtain an unbiased sample of $\langle v_i h_j \rangle_{\text{data}}$ by clamping the visible units to a vector in the training data set, and sampling the hidden units in parallel according to

$$p(h_j = 1|\boldsymbol{v}) = \frac{1}{1 + \exp(-b_j - \sum_i W_{ij} v_i)}. \tag{2.37}$$

This is repeated for each vector in the training set to obtain the empirical estimate of $\langle v_i h_j \rangle_{\text{data}}$[†]. To compute $\langle v_i h_j \rangle_{\text{model}}$ requires us to obtain unbiased samples from the joint distribution $p(\boldsymbol{v}, \boldsymbol{h})$. However, there is no known algorithm to draw samples from this distribution in a practical amount of time. We can perform alternating Gibbs sampling by iterating between sampling from $p(\boldsymbol{h}|\boldsymbol{v})$ using Eq. 2.37 and sampling from $p(\boldsymbol{v}|\boldsymbol{h})$ using

$$p(v_i = 1|\boldsymbol{h}) = \frac{1}{1 + \exp(-a_i - \sum_j W_{ij} h_j)}. \tag{2.38}$$

However, Gibbs sampling in high-dimensional spaces typically takes too long to converge. Empirical evidence suggests that rather than running the Gibbs sampler to convergence, learning works well if we replace Eq. 2.36 with

$$\Delta W_{ij} \propto \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{recon}}, \tag{2.39}$$

where the second expectation is with respect to the distribution of "reconstructed" data. The reconstruction is obtained by starting with a data vector on the visible units and alternating between sampling all of the hidden units using Eq. 2.37 and all of the visible units using Eq. 2.38 K times. The learning rule for the hidden biases is just a simplified version of Eq. 2.39:

$$\Delta b_j \propto \langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{recon}}. \tag{2.40}$$

The above procedure is not maximum likelihood learning but it corresponds to approximately following the gradient of another function called the

[†]In practice, it is common to present the data to the network in small "mini-batches" and update the weights after each mini-batch.

contrastive divergence (Hinton, 2002). We use the notation CD-K to denote contrastive divergence using K steps of alternating Gibbs sampling. Typically K is set to 1, but recent results show that gradually increasing K with learning can significantly improve performance at a modest additional computational cost (Carreira-Perpinan and Hinton, 2005).

Mean field Boltzmann Machines through time

Williams and Hinton (1990) proposed an early method to extend Boltzmann Machines to model sequences. Their networks were based on a deterministic approximation to a Boltzmann Machine called a mean field network (MFN). Replacing the binary stochastic units with real-valued deterministic units permits faster learning at the cost of not being able to model multi-modal distributions. The temporally extended MFN parallelized time by laying out the data in an input buffer, but shared weights across time-steps, so that it did not suffer the usual problems associated with spatial representations. The architecture of this model is shown in Figure 2.4.

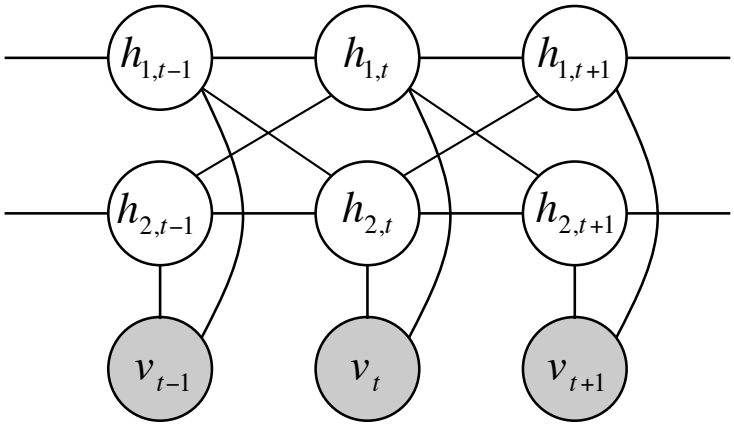


Figure 2.4. Williams and Hinton's Boltzmann Machine through time.

Williams and Hinton used a specialized discriminative learning rule for the task of classifying temporally distorted strings. The results showed an improvement over similarly-sized HMMs and indicated that componential structure was discovered by the networks. The temporal MFN was only applied to a single small task where it is difficult to appreciate the benefits of distributed hidden state (in terms of reducing parameters). However,

the model was an early glimpse at the potential afforded by distributed representations.

Spiking Boltzmann Machines

The Spiking Boltzmann Machine (SBM) (Hinton and Brown, 2000) is another temporal extension of Boltzmann Machines that unrolls the visible and hidden units through time and employs weight-sharing to achieve invariance to time shifts. The SBM also incorporates temporal smoothness by means of a low-pass filter that smooths out the binary activations of the hidden units[†]. This is useful for modeling data such as video or human motion, where changes in pixel intensity or joint angles from one frame to the next tend to be smooth.

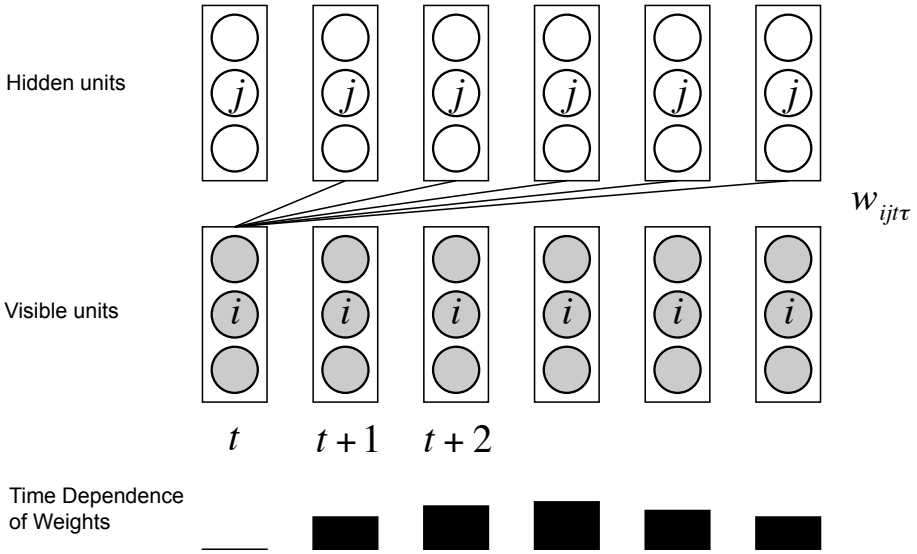


Figure 2.5. A Spiking Boltzmann Machine unrolled through time. The connections represent the weight between a single visible unit and a single hidden unit. The bar chart shows the slowly changing influence of visible units on the hidden units via the fixed temporal kernel.

The SBM is shown in Figure 2.5. Each visible unit is connected to each hidden unit at successive time steps by a time-dependent weight, $W_{ijt\tau} = W_{ij}r(t-\tau)$. The temporal kernel, $r(t-\tau)$, is fixed to have a sharp rise time

[†]Much like a convolutional neural network which achieves spatial smoothness by spatial filters and weight sharing.

and a slow decay and so only the scaling, W_{ij}, is learned. The SBM also gains hidden unit dynamics by borrowing an idea from Elman networks: the hidden units are allowed to see their past states, but the past states are treated as fixed observations. This approximation has been used in more recent temporal extensions of RBMs (Sutskever and Hinton, 2007).

The SBM is still a bi-partite graph. Exact inference is easy and the parameters can be learned by contrastive divergence. The reconstruction step, however, is non-causal. To reconstruct a visible unit requires knowledge of all future hidden units up to the width of the temporal kernel. The SBM, therefore, cannot be used in an on-line setting.

3 Modeling Human Motion

Motion capture (mocap) is the process of recording the movement of a subject as a time series of 3D Cartesian coordinates corresponding to real or virtual points on the body. Most modern systems use a series of synchronized high-speed cameras to capture the location of strategically-placed physical markers attached to the subject (so called "marker-based" systems) or use image features to infer points of interest (so-called "markerless" systems). Marker-based systems are much more common but necessitate the use of a laboratory setting. Markerless systems permit motion capture in more natural environments (e.g. outdoors) but in general require more time to post-process the data. Recent advances in motion capture technology have fueled interest in the analysis and synthesis of motion data for computer animation and tracking. In Chapter 6 we discuss the use of mocap for building dynamical priors for tracking people in monocular and multi-view video sequences. In this chapter, we briefly review mocap-driven motion synthesis.

3.1 Motion synthesis for computer animation

A dominant approach in computer animation is "keyframing" whereby an animator employs software to manually configure the "key" body poses over time, and these frames are interpolated to form smooth trajectories. This process, however, is time and labor intensive. It is therefore common to use mocap data to supplement or replace keyframing. A variety of methods have been developed to utilize the plethora of high-quality motion sequences

available for animation. These approaches can be loosely divided into a handful of categories which we describe below.

Concatenation methods. Perhaps the simplest way to generate new motion sequences based on data is to sensibly concatenate short examples from a motion database to meet sparse user-specified constraints (Tanco and Hilton, 2000; Arikan and Forsyth, 2002; Kovar et al., 2002; Lee et al., 2002; Arikan et al., 2003). Pullen and Bregler (2002) propose a hybrid approach where low-frequency components are retained from user input and high-frequency components, called "texture", are added from the database. The obvious benefit of concatenation approaches is the high-quality motion that is produced. However, the "synthesized" motions are restricted to content already in the database and therefore many resources must be devoted to capture all desired content.

Blending and interpolation methods. Many methods produce new motions by interpolating or blending existing content from a database (Rose et al., 1998; Park et al., 2002; Kovar and Gleicher, 2004; Mukai and Kuriyama, 2005). Unfortunately, these methods typically require extensive pre-processing which generally involves some type of time-warping to align the original sequences. Furthermore, the resulting motions often grossly violate dynamics, resulting in artifacts such as "footskate" and thereby requiring extensive clean-up using inverse kinematics.

Transforming existing motion. Another method is to transform motion in the training data to new sequences by learning to adjust its style or other characteristics (Urtasun et al., 2004; Hsu et al., 2005; Torresani et al., 2007). Such approaches have produced impressive results given user-supplied motion content but we seek more powerful methods that can synthesize both style and content.

Physics-based methods. Models based on the physics of masses and springs have produced some impressive results by using sophisticated "energy-based" learning methods (LeCun et al., 1998) to estimate physical parameters from motion capture data (Liu et al., 2005). However, if we want to generate realistic human motion, we need to model all the complexities of the real dynamics which is extremely difficult to do analytically. In this thesis we focus on model

driven analysis and synthesis but avoid the complexities involved in imposing physics-based constraints, relying instead on a "pure" learning approach in which all the knowledge in the model comes from the data.

Generative models. Data from modern motion capture systems is high-dimensional and contains complex nonlinear relationships among the components of each observation, which is typically a series of joint angles with respect to some skeletal structure. This is a challenge for existing approaches to sequence modeling. However, there are examples of successes in the literature. Brand and Hertzmann (2000) model style and content of human motion with Hidden Markov Models (HMMs) whose emission distributions depend on stylistic parameters learned directly from the data. Their approach permits sampling of novel sequences from the model and applying new styles to existing content. HMMs, however, cannot efficiently model mocap data due to their simple, discrete state. Linear Dynamical Systems, on the other hand, have a more powerful hidden state but they cannot model the complex nonlinear dynamics created by the nonlinear properties of muscles, contact forces of the foot on the ground and myriad other factors. This problem has been addressed by applying piecewise-linear models to synthesize motion (Pavlovic et al., 2001; Li et al., 2002; Bissacco, 2005). In general, exact inference and learning is intractable in such models and approximations are costly and difficult to evaluate.

Gaussian process models. Models based on Gaussian processes (GPs) have received a great deal of recent attention, especially in the tracking literature (see Chapter 6). The Gaussian Process Dynamical Model (Wang et al., 2006) extends the Gaussian Process Latent Variable Model (GP-LVM) (Lawrence, 2004) with a GP-based dynamical model over the latent representations. This model has been shown to discover interesting structure in motion data and permit synthesis of simple actions. However, the main concern with GP-based approaches is their computational expense (cubic in the number of training examples for learning, quadratic in the number of training examples for prediction or generation). This problem may be alleviated by sparse methods (see Chapter 6) but this remains to be seen. Another downside of the GPDM is that a single model cannot synthesize multiple types of motion, a limitation of the simple manifold structure and unimodal dynamics learned by these

models. Recently proposed models such as the multi-factor GP (Wang et al., 2007) and hierarchical GP-LVMs (Lawrence and Moore, 2007) address this limitation.

3.2 Data representation

The most statistically salient patterns of variation in the data may differ considerably from the patterns that humans find perceptually and expressively salient (Brand and Hertzmann, 2000). Therefore our learning algorithms can benefit from a carefully chosen representation that highlights important sources of variation and suppresses irrelevant sources of variation. Specifically, we aim to make our representation of motion invariant to rotation about the gravitational vertical (which we will simply call the vertical) and translation in the ground-plane. In the following discussion, we describe the steps taken to achieve a representation amenable to learning.

3.2.1 Original representation

Data from a motion capture system typically consists of the 3D Cartesian coordinates of 15-30 virtual markers (usually representing joint centres) for a series of discrete time-steps, which we call frames. The data is processed to remove missing and noisy markers and then converted to a joint angle hierarchy through an optimization that assumes constant limb lengths. For each frame, we obtain a vector of relative joint angle orientations, each 1-3 degrees of freedom (dof) plus a root orientation and translation in global coordinates (6 dof). The definition of the root depends on the data source, but typically it is the coccyx, near the base of the back. In our experiments, we used a variety of mocap sources, each of which provided the data already in a hierarchical "joint-angle" format.

3.2.2 Conversion to exponential maps

The most common representation for orientations in mocap data are Euler angles. Euler angles describe a one, two or three dof orientation by a sequence of rotations about axes in the global or local coordinate system. The order of rotations is user-defined and is a common source of confusion, often differing between data sources. Euler angles do not permit distances between rotations

to be directly computed nor do they support interpolation or optimization since the orientation space is highly nonlinear. It is also not trivial to ensure that similar poses are expressed by similar Euler angles. Euler angles also suffer from "gimbal lock", the loss of rotational degrees of freedom due to singularities in the parameter space. Equivalent representations such as the 3×3 rotation matrix or 4D quaternion are not well suited to optimization and synthesis as they require additional constraints to ensure that they remain valid. Therefore we convert joint angles to an exponential map parameterization (Grassia, 1998) before learning.

The exponential map parameterization is also known as "axis-angle" representation since it consists of a three-element vector, whose direction specifies an axis of rotation and whose magnitude specifies the angle by which to rotate about this axis. Exponential maps are well suited to interpolation, optimization and unconstrained synthesis since they are locally linear and every three-element vector maps to a valid rotation. The parameterization still contains singularities and therefore is subject to gimbal lock, but the singularities in the exponential map are often avoidable (Grassia, 1998).

For joints with a single degree of freedom, the exponential map reduces to an Euler angle and so we do not convert. The orientation of the root does not need to be converted to exponential maps since we build an alternative representation in the following section which requires the orientation to be expressed as a 3×3 rotation matrix.

3.2.3 Conversion to body-centred orientations

We treat the root specially because it encodes a transformation with respect to a fixed global coordinate system. At each frame, t, this transformation can be described by a 3×3 rotation matrix, R_t, and a translation vector, $\begin{bmatrix} x_t & y_t & z_t \end{bmatrix}^T$[†]. We will assume, for our discussion, that z corresponds to the vertical. When R_t is the identity matrix, this defines the "rest position" which is typically defined by skeleton meta-data that accompanies the joint angles. Without loss of generality, let us assume that in the rest position the subject is axis-aligned such that the dorsoventral axis (from spinal column to belly) aligns with the x axis:

$$\boldsymbol{u}_t^0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T, \tag{3.1}$$

the lateral axis (from left to right side of body) aligns with the y axis:

$$\boldsymbol{v}_t^0 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T, \tag{3.2}$$

[†]It is also common to represent the transformation by a 4×4 matrix.

and the anteroposterior axis (from head to feet) aligns with the negative z axis:

$$\boldsymbol{w}_t^0 = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T. \tag{3.3}$$

When the root is rotated (i.e. R_t is not the identity) the body-centred coordinate system is no longer axis aligned. It becomes:

$$\boldsymbol{u}_t = R_t^T \boldsymbol{u}_t^0, \tag{3.4}$$

$$\boldsymbol{v}_t = R_t^T \boldsymbol{v}_t^0, \tag{3.5}$$

$$\boldsymbol{w}_t = R_t^T \boldsymbol{w}_t^0 \tag{3.6}$$

where we have assumed a particular convention for the rotation matrix. Note that these axes are simply the rows of the rotation matrix under our chosen convention.

Measuring the angle that the dorsoventral axis makes with the vertical gives us a measure of pitch:

$$\phi_t = \cos^{-1}\left(\frac{\boldsymbol{u}_t \cdot \boldsymbol{w}_t^0}{||\boldsymbol{u}_t||\,||\boldsymbol{w}_t^0||}\right) = \cos^{-1}\left(\frac{\boldsymbol{u}_t \cdot \boldsymbol{w}_t^0}{||\boldsymbol{u}_t||}\right). \tag{3.7}$$

Similarly, measuring the angle that the lateral axis makes with the gravitational vertical gives us a measure of roll:

$$\psi_t = \cos^{-1}\left(\frac{\boldsymbol{v}_t \cdot \boldsymbol{w}_t^0}{||\boldsymbol{v}_t||\,||\boldsymbol{w}_t^0||}\right) = \cos^{-1}\left(\frac{\boldsymbol{v}_t \cdot \boldsymbol{w}_t^0}{||\boldsymbol{v}_t||}\right). \tag{3.8}$$

Both pitch and roll are invariant to rotation about the vertical and therefore can be thought of as "body-centred" rotations. By projecting $\boldsymbol{u}_t$ into the ground-plane, this provides a measure of yaw, or rotation about the vertical:

$$\theta_t = \tan^{-1}\left(\frac{u_t^y}{u_t^x}\right) \tag{3.9}$$

where u_t^x and u_t^y are the first two components of vector $\boldsymbol{u}_t$. Care should be taken to use the four-quadrant version of $\tan^{-1}$ (often called the `atan2` function). We unwrap θ_t to eliminate discontinuities.

3.2.4 Conversion to incremental changes

We represent the rotation about the vertical, as well as translations in the ground plane by their incremental changes (forward differences) and not their

absolute values:

$$\dot{\theta}_t = \theta_{t+1} - \theta_t, \tag{3.10}$$

$$\dot{x}_t = x_{t+1} - x_t, \tag{3.11}$$

$$\dot{y}_t = y_{t+1} - y_t. \tag{3.12}$$

For the last frame, we can use the two preceding frames to make a constant-velocity prediction. To achieve translational invariance, we need to express velocity in the ground-plane with respect to body-centred and not global coordinates. We can represent velocity in the ground-plane by its magnitude:

$$\alpha_t = \sqrt{\dot{x}_t^2 + \dot{y}_t^2} \tag{3.13}$$

and its angle with respect to the x-axis:

$$\beta_t = \tan^{-1}\left(\frac{\dot{y}_t}{\dot{x}_t}\right). \tag{3.14}$$

Again we make use of the four-quadrant version of $\tan^{-1}$. The velocity is then expressed with respect to the orientation about the vertical, θ_t, in both a forward and lateral component:

$$\dot{\gamma}_t = \alpha_t \cos\left(\theta_t - \beta_t\right), \tag{3.15}$$

$$\dot{\xi}_t = \alpha_t \sin\left(\theta_t - \beta_t\right) \tag{3.16}$$

where we have used "dot" notation to imply that these quantities are incremental values. Taken collectively, $\begin{bmatrix} \dot{\gamma}_t & \dot{\xi}_t & z_t & \phi_t & \psi_t & \dot{\theta}_t \end{bmatrix}^T$ form our invariant representation of the root. Note that the height, z_t, is untouched.

3.2.5 Data normalization

Any joint angle dimensions that have constant value are not modeled and removed from the training data (they are re-inserted before playback or export). Each component of the data is normalized to have zero mean and unit variance.

4 Conditional Restricted Boltzmann Machines

We have emphasized that models with distributed hidden state are necessary for efficiently modeling complex time series. But using distributed representations for hidden state in directed models of time series (Bayes nets) makes inference difficult in all but the simplest models. If, however, we use a Restricted Boltzmann Machine to model the probability distribution of the observation vector at each time frame, the posterior over latent variables factorizes completely, making inference easy. In this chapter, we introduce the Conditional Restricted Boltzmann Machine (CRBM). The CRBM extends the RBM to capture temporal dependencies yet maintains its most important computational properties: simple, exact inference and efficient approximate learning using the contrastive divergence algorithm.

4.1 RBMs with real-valued observations

Typically, RBMs use stochastic binary units for both the visible data and hidden variables, but for many applications the observed data is non-binary. For some domains (e.g. modeling handwritten digits) we can normalize the data and use the real-valued probabilities of the binary visible units in place of their activations. When we use mean-field logistic units to model data that is very non-binary (e.g. modeling patches of natural images), it is difficult to obtain sharp predictions for intermediate values and so it is more desirable to use units that match the distribution of the data.

Fortunately, the stochastic binary units of RBMs can be generalized to any

distribution that falls in the exponential family (Welling et al., 2005). This includes multinomial units, Poisson units and linear, real-valued units that have Gaussian noise (Freund and Haussler, 1992). To model real-valued data (e.g. mocap), we use a modified RBM with binary logistic hidden units and real-valued Gaussian visible units. Any setting of the hidden units makes a linear contribution to the mean of each visible unit:

$$p(v_i|\boldsymbol{h}) = \mathcal{N}\left(a_i + \sigma_i \sum_j W_{ij} h_j, \sigma_i^2\right) \tag{4.1}$$

where a_i is the bias and σ_i is the standard deviation of the Gaussian noise for visible unit i. The symmetric weight, W_{ij}, connects visible unit i to hidden unit j. Inference simply uses a scaled form of Eq. 2.37:

$$p(h_j = 1|\boldsymbol{v}) = \frac{1}{1 + \exp(-b_j - \sum_i W_{ij} \frac{v_i}{\sigma_i})} \tag{4.2}$$

where b_j is the bias of hidden unit j. The joint probability of $\boldsymbol{v}$ and $\boldsymbol{h}$ follows the form of Eq. 2.33 where the energy function is now

$$E(\boldsymbol{v}, \boldsymbol{h}) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{ij} W_{ij} \frac{v_i}{\sigma_i} h_j - \sum_i b_j h_j. \tag{4.3}$$

Given the hidden units, the distribution of each visible unit is defined by a parabolic log likelihood function that makes extreme values very improbable. For any setting of the parameters, the gradient of the quadratic log likelihood will always overwhelm the gradient due to the weighted input from the binary hidden units provided the value v_i of a visible unit is far enough from its bias, a_i. Conveniently, the contrastive divergence learning rules remain the same as in an RBM with binary visible units.

Finally, a brief note about σ_i: while it can be learned, in practice, we rescale our data to have zero mean and unit variance. We have found that fixing σ_i at unity makes the learning work well even though we would expect a good model to predict the data with much higher precision. For the remainder of the discussion, we will assume $\sigma_i = 1$.

4.2 The Conditional RBM

The RBM models static frames of data, but does not incorporate any temporal information. We can model temporal dependencies by treating

the visible variables in the previous time slice(s) as additional fixed inputs. We add two types of directed connections: autoregressive connections from the past N configurations (time steps) of the visible units to the current visible configuration, and connections from the past M configurations of the visible units to the current hidden configuration. The addition of these directed connections turns the RBM into a Conditional RBM (Figure 4.1). The autoregressive weights can model linear, temporally local structure very well, leaving the hidden units to model nonlinear, higher-level structure.

N and M are tunable parameters and need not be the same for both types of directed connections. To simplify discussion, we will assume $N = M$ and refer to N as the order of the model. Typically, in our experiments, we use a small number such as $N = 3$. In modeling motion capture with higher frame rates, we have found that a good rule of thumb is to set $N = F/10$ where F is the frame rate of the data (in frames per second).

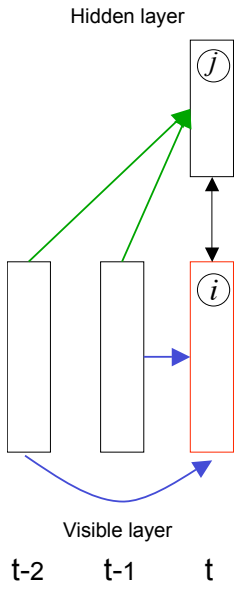


Figure 4.1. Architecture of our model. In this figure we show $N = 2$ but in our experiments, we typically use a slightly higher order.

To simplify the presentation, we will assume the data at $t-1, \ldots, t-N$ is concatenated into a "history" vector which we call $v_{<t}$. So if v_t is of dimension D, then $v_{<t}$ is of dimension $N \cdot D$. We will use k to index the elements of $v_{<t}$.

The autoregressive parameters are summarized by an $N \cdot D \times D$ weight matrix called A and the directed "past to hidden" parameters are summarized by an $N \cdot D \times H$ matrix B where H is the number of binary hidden units. This does not change the computation, but allows us to simplify the presentation of the following equations as we can avoid explicitly summing over past frames.

4.2.1 Inference and learning

Fortunately, inference in the CRBM is no more difficult than in the standard RBM. The states of the hidden units are determined by both the input they receive from the current observation and the input they receive from the recent past. Given $\boldsymbol{v}_t$ and $\boldsymbol{v}_{<t}$, the hidden units at time t are conditionally independent. The effect of the past on each hidden unit can be viewed as a dynamic bias:

$$\hat{b}_{j,t} = b_j + \sum_k B_{kj} v_{k,<t} \tag{4.4}$$

which includes the static bias, b_j, and the contribution from the past. This only slightly modifies the factorial distribution over hidden units: b_j in Eq. 2.37 is replaced with $\hat{b}_{j,t}$ to obtain

$$p(h_{j,t} = 1 | \boldsymbol{v}_t, \boldsymbol{v}_{<t}) = \frac{1}{1 + \exp(-\hat{b}_{j,t} - \sum_i W_{ij} v_{i,t})}. \tag{4.5}$$

Note that we are now conditioning on $\boldsymbol{v}_{<t}$. Figure 4.2 shows an example of frame-by-frame inference in a trained CRBM.

The past has a similar effect on the visible units. The reconstruction distribution becomes

$$p(v_{i,t} | \boldsymbol{h_t}, \boldsymbol{v}_{<t}) = \mathcal{N}\left(\hat{a}_{i,t} + \sum_j W_{ij} h_{j,t}, 1\right) \tag{4.6}$$

where $\hat{a}_{i,t}$ is also a dynamically changing bias that is an affine function of the past:

$$\hat{a}_{i,t} = a_i + \sum_k A_{ki} v_{k,<t}. \tag{4.7}$$

We can still use contrastive divergence for training the CRBM. The updates for the symmetric weights, W, as well as the static biases, $\boldsymbol{a}$ and $\boldsymbol{b}$, have the same form as Eq. 2.39 and Eq. 2.40 but have a different effect because the states of the hidden units are now influenced by the previous visible units. The

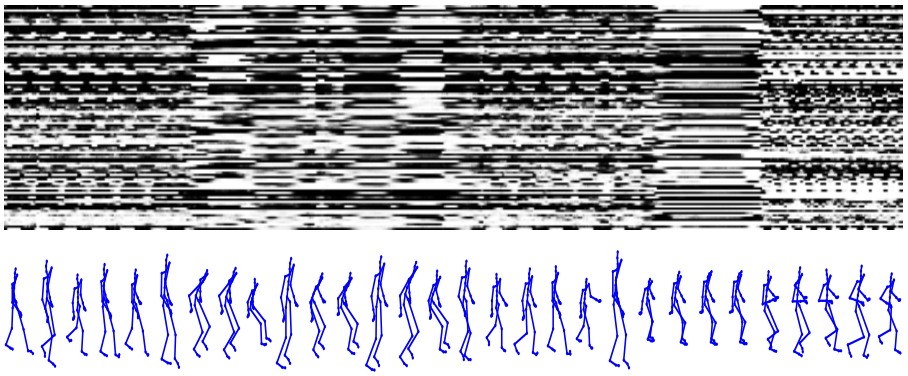


Figure 4.2. In a trained model, probabilities of each feature being "on", conditional on the data at the visible units. Shown is a 100-hidden unit model and a sequence which contains (in order) walking, sitting/standing (three times), walking, crouching, and running. Rows represent features, columns represent sequential frames.

updates for the directed weights are also based on simple pairwise products. The gradients are now summed over all time steps:

$$\Delta W_{ij} \propto \sum_t \left(\langle v_{i,t} h_{j,t} \rangle_{\text{data}} - \langle v_{i,t} h_{j,t} \rangle_{\text{recon}} \right) \tag{4.8}$$

$$\Delta A_{ki} \propto \sum_t \left(\langle v_{i,t} v_{k,<t} \rangle_{\text{data}} - \langle v_{i,t} v_{k,<t} \rangle_{\text{recon}} \right) \tag{4.9}$$

$$\Delta B_{kj} \propto \sum_t \left(\langle h_{j,t} v_{k,<t} \rangle_{\text{data}} - \langle h_{j,t} v_{k,<t} \rangle_{\text{recon}} \right) \tag{4.10}$$

$$\Delta a_i \propto \sum_t \left(\langle v_{i,t} \rangle_{\text{data}} - \langle v_{i,t} \rangle_{\text{recon}} \right) \tag{4.11}$$

$$\Delta b_j \propto \sum_t \left(\langle h_{j,t} \rangle_{\text{data}} - \langle h_{j,t} \rangle_{\text{recon}} \right) \tag{4.12}$$

where $\langle \cdot \rangle_{\text{data}}$ is an expectation with respect to the data distribution, and $\langle \cdot \rangle_{\text{recon}}$ is the K-step reconstruction distribution as obtained by alternating Gibbs sampling, starting with the visible units clamped to the training data.

While learning a CRBM, we do not need to proceed sequentially through the training data sequences. The updates are only conditional on the past N time steps, not the entire sequence. As long as we isolate "chunks" of $N+1$ frames (the size depending on the order of the directed connections), these small windows can be mixed and formed into mini-batches. To speed up the

learning, we assemble these chunks of frames into "balanced" mini-batches of size 100.

We randomly assign chunks to different mini-batches so that the chunks in each mini-batch are as uncorrelated as possible. To save computer memory, time frames are not actually replicated in mini-batches; we simply use indexing to simulate the "chunking" of frames.

4.2.2 Scoring observations

The CRBM defines a joint probability distribution over a data vector, $\boldsymbol{v}_t$, and a vector of hidden states, $\boldsymbol{h}_t$, conditional on the recent past, $\boldsymbol{v}_{<t}$:

$$p(\boldsymbol{v}_t, \boldsymbol{h}_t | \boldsymbol{v}_{<t}) = \frac{\exp\left(-E\left(\boldsymbol{v}_t, \boldsymbol{h}_t | \boldsymbol{v}_{<t}\right)\right)}{Z(\boldsymbol{v}_{<t})} \tag{4.13}$$

where the partition function, Z, is constant with respect to $\boldsymbol{v}_t$ and $\boldsymbol{h}_t$ but depends on $\boldsymbol{v}_{<t}$. As in the RBM, it is intractable to compute exactly because it involves an integration over all possible settings of the visible and hidden units:

$$Z(\boldsymbol{v}_{<t}) = \sum_{\boldsymbol{h}'_t} \int_{\boldsymbol{v}'_t} \exp\left(\left(-E(\boldsymbol{v}'_t, \boldsymbol{h}'_t | \boldsymbol{v}_{<t}\right)\right) d\boldsymbol{v}'_t. \tag{4.14}$$

The energy function is given by

$$E(\boldsymbol{v}_t, \boldsymbol{h}_t | \boldsymbol{v}_{<t}) = \frac{1}{2} \sum_i (v_{i,t} - \hat{a}_{i,t})^2 - \sum_{ij} W_{ij} v_{i,t} h_{j,t} - \sum_j \hat{b}_{j,t} h_{j,t} \tag{4.15}$$

where we have assumed $\sigma_i = 1$. The probability of observing $\boldsymbol{v}_t$ can be expressed by marginalizing out the binary hidden units:

$$p(\boldsymbol{v}_t | \boldsymbol{v}_{<t}) = \sum_{\boldsymbol{h}_t} p(\boldsymbol{v}_t, \boldsymbol{h}_t | \boldsymbol{v}_{<t}) = \frac{\sum_{\boldsymbol{h}_t} \exp\left(-E(\boldsymbol{v}_t, \boldsymbol{h}_t | \boldsymbol{v}_{<t})\right)}{Z(\boldsymbol{v}_{<t})}. \tag{4.16}$$

Under the CRBM, the probability of observing a *sequence*, $\boldsymbol{v}_{(N+1):T}$, given $\boldsymbol{v}_{1:N}$, is just the product of all the local conditional probabilities:

$$p(\boldsymbol{v}_{(N+1):T} | \boldsymbol{v}_{1:N}) = \prod_{t=N+1}^{T} p(\boldsymbol{v}_t | \boldsymbol{v}_{<t}). \tag{4.17}$$

We do not attempt to model the first N frames of each sequence, though a separate set of biases could be learned for this purpose.

Although the partition function makes Eq. 4.16 and Eq. 4.17 intractable to compute exactly, we can exploit the fact that the hidden units are binary and integrate them out to arrive at the "free energy":

$$F(\boldsymbol{v}_t|\boldsymbol{v}_{<t}) = \frac{1}{2}\sum_i (v_{i,t} - \hat{a}_{i,t})^2 - \sum_j \log\left(1 + \exp(\sum_i W_{ij} v_{i,t} + \hat{b}_{j,t})\right), \quad (4.18)$$

which is a function of the model parameters and recent past. It is the negative log probability of an observation plus $\log Z$ (see Eq. 4.13). The free energy allows us to score observations under a fixed setting of the parameters, but unlike a probability it does not let us compare between models[†]. It can still be useful, however, in making deterministic forward predictions (as described in the following section). The details on deriving the free energy for an RBM can be found in (Freund and Haussler, 1992).

4.2.3 Generation

Causal generation from a learned CRBM can be done on-line with no smoothing, just like the learning procedure. The visible states at the last few time steps determine the effective biases of the visible and hidden units at the current time step. We always keep the previous visible states fixed and perform alternating Gibbs sampling to obtain a joint sample from the CRBM. This picks new hidden and visible states that are compatible with each other and with the recent (visible) history (Figure 4.3). To start alternating Gibbs sampling, we need to initialize with either $\boldsymbol{v}_t$ or $\boldsymbol{h}_t$. For time-series data that is smooth (e.g. mocap), a good choice is to initially set $\boldsymbol{v}_t = \boldsymbol{v}_{t-1}$. In practice, we alternate 30 to 100 times, though the quality of generated data does not seem to be sensitive to this parameter.

Generation does not require us to retain the training dataset, but it does require initialization with N observations. Typically we use randomly drawn consecutive frames from the training data as an initial configuration.

A trained CRBM has the ability to easily fill in missing data (complete or partial observations), regardless of where the dropouts occur in a sequence. Due to its conditional independence assumptions it has the ability to fill in such missing data on-line. Filling in missing data with the CRBM is very

[†]Different models will have different partition functions.

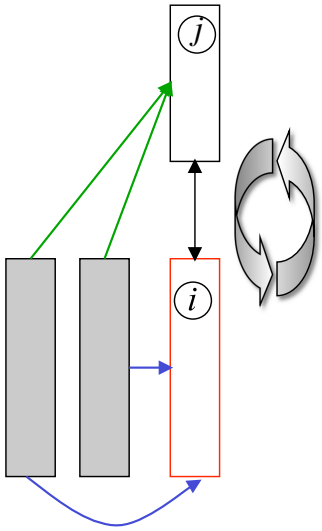


Figure 4.3. On-line generation from a CRBM. Shading implies that the past is held fixed while alternating Gibbs sampling between the hidden and visible units.

similar to generation. We simply clamp the known data to the visible units, initialize the missing data to something reasonable (for example, the value at the previous frame), and alternate between stochastically updating the hidden and visible units, *with the known visible states held fixed*.

The noise in sampling may be an asset when using the CRBM to generate sequences, but when using the CRBM to fill in missing data, or in a predictive setting it may be undesirable. Rather than obtaining a sample, we may want the model's "best guess". Given the model parameters, and past history, we can follow the negative gradient of the free energy (Eq. 4.18) with respect to either a complete or partial setting of the visible variables, $\boldsymbol{v}_t$:

$$-\frac{\partial F(\boldsymbol{v}_t|\boldsymbol{v}_{<t})}{\partial v_{i,t}} = v_{i,t} - \left(\hat{a}_{i,t} + \sum_j W_{ij} f\left(-\sum_i W_{ij} v_{i,t} - \hat{b}_{j,t}\right)\right) \tag{4.19}$$

where $f(\cdot)$ is the logistic function. The gradient at a unit has an intuitive form: it is the difference between its current value and the value that would be obtained by mean-field reconstruction. We use conjugate-gradient optimization, but any general purpose gradient-based optimizer is suitable.

4.2.4 Approximations

In practice, we make several small modifications to the algorithms for both learning and generation. These rely on several approximations, most of which are chosen based on collective experience of training similar networks. The approximations typically replace sampled values with expected values, to reduce unnecessary noise.

While training a CRBM, we replace $v_{i,t}$ in Eq. 4.8, 4.9 and 4.11 by its expected value and we also use the expected value of $v_{i,t}$ when computing the probability of activation of the hidden units (Eq. 4.5). However, to compute each of the K reconstructions of the data (Eq. 4.6), we use stochastically chosen binary values of the hidden units. This prevents the hidden activities from transmitting an unbounded amount of information from the data to the reconstruction (Teh and Hinton, 2001).

While updating the directed visible-to-hidden connections (Eq. 4.10), the symmetric undirected connections (Eq. 4.8), and the hidden biases (Eq. 4.12), we use the stochastically chosen binary values of the hidden units in the first term (under the data), but replace $h_{j,t}$ by its expected value in the second term (under the reconstruction). We take this approach because the reconstruction of the data depends on the binary choices made when selecting hidden state. Thus, when we infer the hiddens from the reconstructed data, the probabilities are highly correlated with the binary hidden states inferred from the data. On the other hand, we stop after K reconstructions, so the binary choice of hiddens from the Kth reconstruction does not correlate with any other terms, and there is no reason to include this extra noise.

The alternating Gibbs sampling used when generating data is similar to the procedure we use to learn a CRBM. So we make similar approximations during generation: using stochastically chosen binary values of the hidden units but the expected values of the reconstructed visible units. As a further step to reduce noise, on the *final iteration* of Gibbs sampling, we use the real-valued probabilities of the hidden units when updating the visible units.

4.3 Higher level models: the Conditional Deep Belief Network

Once we have trained the model, we can add layers in the same way as a Deep Belief Network (DBN) (Hinton et al., 2006). The previous layer CRBM

is kept, and the sequence of hidden state vectors, while driven by the data, is treated as a new kind of "fully observed" data. The next level CRBM has the same architecture as the first (though we can alter the number of its units) and is trained in the exact same way. Upper levels of the network can then model higher-order structure.

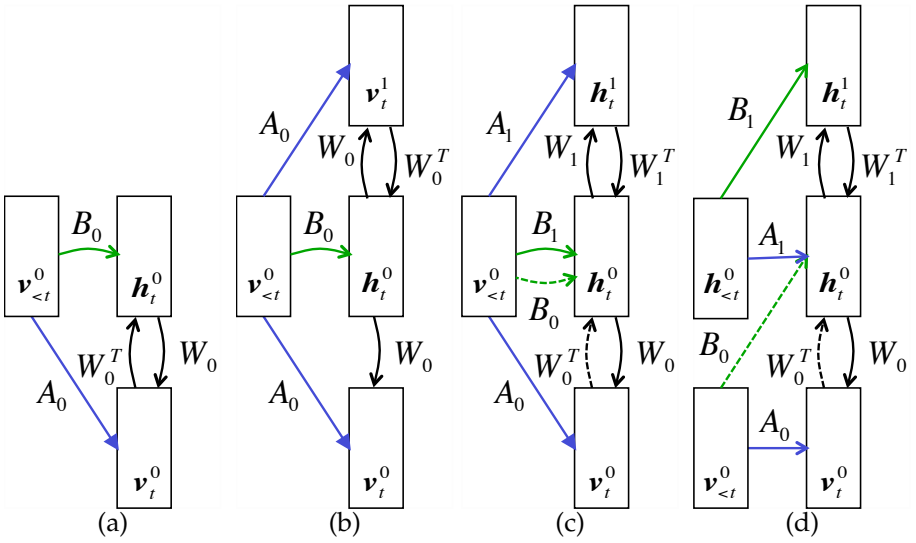


Figure 4.4. Building a Conditional Deep Belief Network. (a) The CRBM. (b) A generative model whose weights between layers are tied. It defines the same joint distribution over $\boldsymbol{v}_t^0$ and $\boldsymbol{h}_t^0$. The top two layers interact using symmetric connections while all other connections are directed. (c) Improving the model by untying the weights; holding W_0, A_0 and B_0 fixed and greedily training W_1, A_1 and B_1. Note that the "dashed" directed, bottom-up weights are not part of the generative model. They are used to infer factorial, approximate posterior distributions over $\boldsymbol{h}_t^0$ when $\boldsymbol{v}_t^0$ is clamped to the data. (d) The model we use in practice. We ignore uncertainty in the past hidden states.

Figure 4.4a shows a CRBM whose symmetric, undirected weights have been represented explicitly by two sets of directed weights: top-down "generative" weights W_0, and bottom-up "recognition" weights, W_0^T. This representation is purely illustrative: it does not at all change the model. The use of the zero subscripts and superscripts simply indicates that the CRBM is first in a series of layers which we will introduce shortly.

Figure 4.4b shows a generative model that is equivalent to the original CRBM in the sense that their joint distributions over $\boldsymbol{v}_t^0$ and $\boldsymbol{h}_t^0$, conditional on $\boldsymbol{v}_{<t}^0$ are the same. We have added a second set of visible units, $\boldsymbol{v}_t^1$, identical to the first, and ensured that the undirected, symmetric weights between $\boldsymbol{v}_t^1$ and $\boldsymbol{h}_t^0$ are equal to the weights used in the original CRBM. The weights are therefore "tied" between the two layers. Additionally, the bottom-up weights between $\boldsymbol{v}_t^0$ and $\boldsymbol{h}_t^0$, W_0^T, are no longer part of the generative model in Figure 4.4b. Although it defines the same joint distribution, its semantics are very different than the CRBM. To generate an observation, $\boldsymbol{v}_t^0$, conditional on $\boldsymbol{v}_{<t}^0$, we must reach equilibrium in the conditional associative memory formed by the top two layers and then perform a single down-pass using directed weights W_0 and A_0. The CRBM generates observations as explained in 4.2.3.

Note that if we observe $\boldsymbol{v}_t^0$, the units $\boldsymbol{h}_t^0$ are no longer conditionally independent because the undirected connections between $\boldsymbol{v}_t^0$ and $\boldsymbol{h}_t^0$ have been replaced by directed connections. The new model is therefore subject to the effects of "explaining away". However, because of the tied weights, the CRBM at the top two layers becomes a "complementary prior": meaning that when we multiply the likelihood term by the prior, the posterior is factorial.

If we hold W_0, A_0 and B_0 fixed, but "untie" the weights between the top two layers (Figure 4.4c) we can improve the generative model by greedily learning W_1, A_1 and B_1, treating the activations of $\boldsymbol{h}_t^0$ while driven by the training data as a kind of "fully-observed" data. When the weights are untied, units in the topmost layer no longer represent the visible units, but another layer of latent features, $\boldsymbol{h}_t^1$. We can still use W_0^T and B_0 (which are not part of the generative model) to infer factorial *approximate* posterior distributions over the states of $\boldsymbol{h}_t^0$.

The joint distribution defined by the original CRBM, $p(\boldsymbol{v}_t^0, \boldsymbol{h}_t^0 | \boldsymbol{v}_{<t}^0)$, decomposes into a mapping from features to data, $p(\boldsymbol{v}_t^0 | \boldsymbol{h}_t^0, \boldsymbol{v}_{<t}^0)$, and an implicit prior over the features, $p(\boldsymbol{h}_t^0 | \boldsymbol{v}_{<t}^0)$. We can think of training the next layer as a means of improving the prior model. By fixing W_0, A_0, the distribution $p(\boldsymbol{v}_t^0 | \boldsymbol{h}_t^0, \boldsymbol{v}_{<t}^0)$ is unchanged. The gain from building a better model of $p(\boldsymbol{h}_t^0 | \boldsymbol{v}_{<t}^0)$ more than offsets the loss from having to perform approximate inference. This greedy learning algorithm can be applied recursively to any number of higher layers and is guaranteed to never decrease a variational lower bound on the log probability of the data under the full generative model. For details see (Hinton

et al., 2006).

In practice, greedily training multiple layers of representation works well. However, there are a number of small changes we make to gain flexibility and improve the computational cost of performing inference and learning. Bending the rules as follows breaks the above guarantee:

1. We replace maximum likelihood learning with contrastive divergence (for obvious computational reasons).

2. The guarantee relies on initializing the weights of each successive layer with the weights in the layer below. This assumes that all layers are of equal sizes. However, in practice, we use layers of different sizes.

3. Rather than train each layer conditional on $\boldsymbol{v}^0_{<t}$ (which we assume to be the fully-observed recent past of the visible units), we ignore uncertainty in the previous hidden states and train each layer on its own recent past, $\boldsymbol{v}^0_{<t}, \boldsymbol{h}^0_{<t}, \ldots, \boldsymbol{h}^{H-1}_{<t}$ (where H is the number of hidden layers), always treating the past as fully-observed.

The model that we use in practice is shown in Figure 4.4d. It is a Conditional Deep Belief Network (CDBN). The inference we perform in this model, conditional on past visible states, is approximate because it ignores the future (it does not do smoothing). Because of the directed connections, exact inference within the model should include both a forward and backward pass through each sequence. We perform only a forward pass because smoothing is intractable in the multi-layer model. Effectively, at each layer we replace the full posterior by an approximate filtering distribution. However, there is no guarantee that this is a good approximation. Compared with an HMM, the lack of smoothing is a loss. But the deep model is still exponentially more powerful at representing data.

4.3.1 On-line generation with higher-level models

The generative model for a Conditional DBN consists of a top-level conditional associative memory (with symmetric weights and dynamic biases) and any number of directed lower layers (with top-down generative weights and dynamic generative biases). We also maintain bottom-up connections that are used in approximate inference. Like in a DBN, to generate a sample, $\boldsymbol{v}_t$, the

associative memory must settle on a joint setting of the units in the top two hidden layers and then the top-down weights are used to generate the lower layers. Since the model is conditional, each layer must also consider the effect of the past via the dynamic biases. Note that in a deep network, all but the topmost hidden layer will have two sets of dynamic biases: recognition biases from when it was greedily trained as a hidden layer, and generative biases from when it was greedily trained as fully observed. We must be careful not to double-count the input to each layer; we use the dynamic recognition biases during inference and dynamic generative biases during generation.

As a concrete example, let us consider generating an observation from a Conditional DBN built by greedily training two CRBMs (the same network shown in Figure 4.4d).

1. If the first CRBM is order N and the second CRBM is order M then we must initialize with $N + M$ frames, $\boldsymbol{v}_{1:(N+M)}$ (Figure 4.5a).

2. Next, we initialize M frames of the first hidden layer using a mean-field up-pass through the first CRBM (Figure 4.5b).

3. Then we initialize the first layer hidden units at $t = N + M + 1$ to be a copy of the real-valued probabilities we have just inferred at $t = N + M$. We perform alternating Gibbs sampling in the 2nd layer CRBM. At each step, we stochastically activate the top-level hidden units, but on the final step, we use the real-valued probabilities of the top layer to obtain the real-valued probabilities of the first layer hidden units (Figure 4.5c).

4. We do a mean-field down-pass in the first layer CRBM to obtain the visible units at time $t = N + M + 1$ (Figure 4.5d).

Again, we copy of the real-valued probabilities of the first layer hidden units to initialize Gibbs sampling for the next frame, and repeat steps 3 and 4 above for as many frames as desired.

4.3.2 Fine-tuning

Following greedy learning, both the weights and the simple inference procedure are suboptimal in all but the top layer of the network, as the weights have not changed in the lower layers since their respective stage of greedy training.

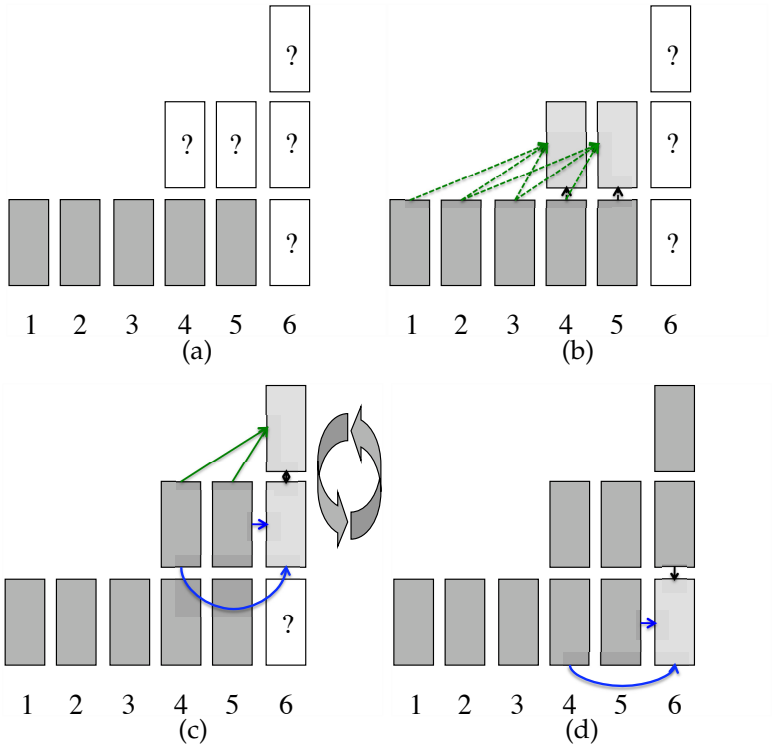


Figure 4.5. Generating from a Conditional Deep Belief Network with two hidden layers. For this example, we assume the first layer CRBM is third order and the second layer CRBM is second order. We provide five frames to initialize the model.

We, however, can use a contrastive form of the "wake-sleep" algorithm (Frey et al., 1997) called the "up-down" algorithm (Hinton et al., 2006) to fine-tune the generative model. In our experiments, we have observed that fine-tuning improves the visual quality of generated sequences at a modest additional computational cost.

4.4 Temporal links between hidden units

In a Conditional Restricted Boltzmann Machine the hidden state and visible state depend only on past instances of the visible variables. The CRBM is a special case of the Temporal Restricted Boltzmann Machine (TRBM) (Sutskever

and Hinton, 2007) in which there are no temporal connections between hidden units. This makes filtering in the CRBM exact, and "mini-batch" learning possible, as training does not have to be done sequentially. This latter property can greatly speed up learning as well as smooth the learning signal, as the order of data vectors presented to the network can be randomized. This ensures that the data in each mini-batch is as uncorrelated as possible.

As soon as we introduce connections between hidden units, we must resort to approximate filtering or deterministic methods (Sutskever et al., 2009) even in a single layer model. In training higher-level models using CRBMs, we gain hidden-to-hidden links via the autoregressive connections of the higher layers. At each stage of greedy learning, filtering is exact within each CRBM. However, filtering in the overall multi-layer model is approximate.

4.5 Experiments

We have carried out a series of experiments training CRBM models on motion capture data from publicly available repositories. After learning a model using the updates described in Sec. 4.2, we can demonstrate in several ways what it has learned about the structure of human motion. Perhaps the most direct demonstration, which exploits the fact that it is a probability density model of sequences, is to use the model to generate *de-novo* a number of synthetic motion sequences. Video files of these sequences are available on the website mentioned in the preamble; these motions have not been retouched by hand in any motion editing software. Note that we also do not have to keep a reservoir of training data sequences for generation - we only need the weights of the trained model and N valid frames for initialization. This implies that our model is thus suitable for low-memory devices[†]. More importantly, we believe that compact models are more likely to be better at generalization.

Data source and representation. The first dataset used in these experiments was obtained from `http://mocap.cs.cmu.edu`. It will be hereafter referred to as the CMU dataset. The second dataset used in these experiments was obtained from `http://people.csail.mit.edu/ehsu/work/sig05stf/` and was

[†]The level of compression obtained will of course vary with the number of free parameters and size of the data set.

released as supplementary material to (Hsu et al., 2005). It will be hereafter referred to as the MIT dataset. The data we gathered consisted of 3D joint angles derived from 30 (CMU) or 17 (MIT) markers plus a root orientation and displacement. Data was represented with the encoding described in Sec. 3.2. The final dimensionality of our data vectors was 62 (CMU) and 49 (MIT).

One advantage of the CRBM is the fact that the data does not need to be heavily preprocessed or dimensionality reduced before learning. Other generative approaches (Brand and Hertzmann, 2000; Li et al., 2002) apply PCA to reduce noise and dimensionality. However, dimensionality reduction becomes problematic when a wider range of motions is to be modeled. The autoregressive connections can be thought of as doing a kind of "whitening" of the data.

Details of learning. Except where noted, all CRBM models were trained as follows: Each training case was a window of $N + 1$ consecutive frames and the order of the training cases was randomly permuted. The training cases were presented to the model as "mini-batches" of size 100 and the weights were updated after each mini-batch. Models were trained using CD-1 (see Sec. 2.4.3) for a fixed number of epochs (complete passes through the data). All parameters used a learning rate of 10^{-3}, except for the autoregressive weights which used a learning rate of 10^{-5}. A momentum term was also used: 0.9 of the previous accumulated gradient was added to the current gradient.

4.5.1 Generation of walking and running sequences from a single model

In our first demonstration, we train a single CRBM on data containing both walking and running motions; we then use the learned model to generate both types of motion, depending on how it is initialized. We extracted 23 sequences of walking and 10 sequences of running from subject 35 in the CMU dataset. After downsampling to 30Hz, the training data consisted of 2813 frames. We trained a 200 hidden-unit CRBM for 4000 passes through the training data, using a third-order model (for directed connections). The order of the sequences was randomly permuted such that walking and running sequences were distributed throughout the training data.

Figure 4.6 shows a walking sequence and a running sequence generated

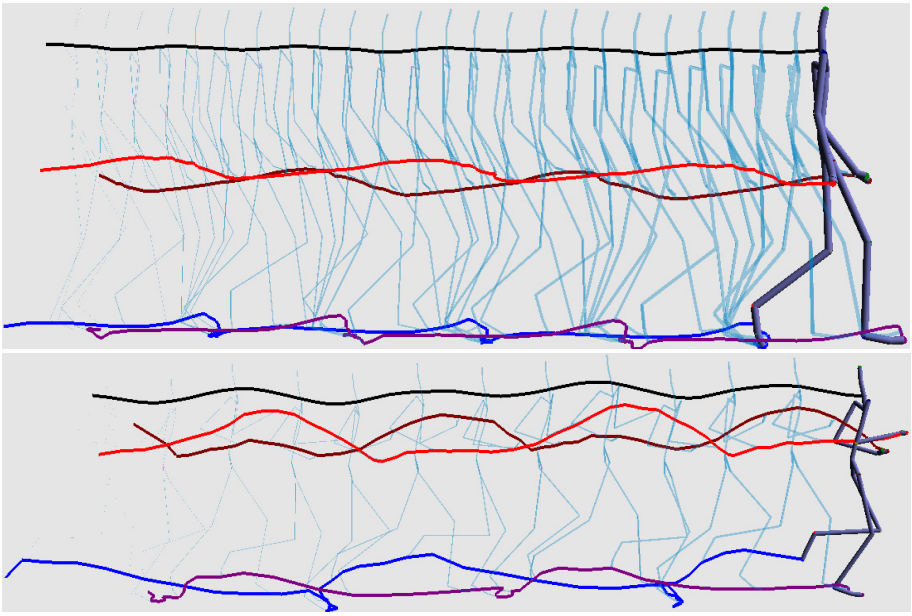


Figure 4.6. After training, the same model can generate walking (top) and running (bottom) motion (see videos on the website). Each skeleton is 4 frames apart.

by the same model, using alternating Gibbs sampling (with the probability of hidden units being "on" conditional on the current and previous three visible vectors). Since the training data does not contain any transitions between walking and running (and *vice-versa*), the model will continue to generate walking or running motions depending on where it is initialized.

4.5.2 Learning transitions between walking and running

In our second demonstration, we show that our model is capable of learning not only several types of homogeneous motion content but also the transitions between them, when the training data itself contains examples of such transitions. We trained on 9 sequences (from the MIT database, file `Jog1_M`) containing long examples of walking and running, as well as a few transitions between the two gaits. After downsampling to 30Hz, this provided us with 2515 frames. Training was done as before, but after the model was trained, an identical 200 hidden-unit model was trained on top of the first model

(see Sec. 4.3). The resulting two-level model was used to generate data. A video available on the website demonstrates our model's ability to stochastically transition between various types of motion during a single generated sequence.

4.5.3 Introducing transitions using noise

In our third demonstration, we show how transitions between different types of motion content can be generated even when such transitions are absent in the data. We use the same model and data as described in Sec. 4.5.1, where we have learned on separate sequences of walking and running. To generate, we use the same sampling procedure as before, except that at each time we stochastically choose the hidden states (given the current and previous three visible vectors) we add a small amount of Gaussian noise to the hidden state biases. This encourages the model to explore more of the hidden state space without deviating too far the current motion. Applying this "noisy" sampling approach, we see that the generated motion occasionally transitions between learned gaits. These transitions appear natural (see the video on the website).

4.5.4 Learning motion style

We have demonstrated that the CRBM can generate and transition between different gaits, but what about its ability to capture more subtle stylistic variation within a particular gait? We also seek to show the CRBM's ability to learn on data at a higher frame-rate (60Hz), and from a much larger training corpus. Finally, we incorporate label information into our training procedure.

From the CMU dataset, we extracted a series of 10 stylized walk sequences performed by subject 137. The walks were labeled as *cat*, *chicken*, *dinosaur*, *drunk*, *gangly*, *graceful*, *normal*, *old-man*, *sexy* and *strong*. We balanced the dataset by repeating the sequences three to six times (depending on the original length) so that our final dataset contained approximately 3000 frames of each style at 60fps.

In general, we used the same training procedure as above, but made a few notable exceptions:

- At each iteration of CD learning, we performed 10 steps of alternating Gibbs sampling (CD-10)

- We added a sparsity term to the energy function to gently encourage the hidden units, while driven by the data, to have an average activation of 0.2 (details below)

- At each iteration of CD learning, we added Gaussian noise with $\sigma = 1$ to each dimension of the past history, $v_{<t}$

These experiments were carried out considerably later than the experiments described in Sec. 4.5.1 to 4.5.3 and so represent a refinement to our learning method. This allows us to cope with the higher frame rate and larger degree of variability in the training set. Recent work on estimating the partition functions of RBMs and evaluating the log probability of held-out sets has shown that models trained with CD>1 , although more computationally demanding to train, are significantly better generative models (Salakhutdinov and Murray, 2008). We have chosen CD-10 as a compromise between resembling maximum likelihood learning and minimizing computational cost. Regularizing by adding noise to the inputs can aid in achieving robustness to noise during synthesis. This has similar effects to adding L2 weight decay to the directed weights (see Sec. 6.4.1 for more details).

The recent popularity of sparse, overcomplete latent representations has highlighted both the theoretical and practical motivations for their use in unsupervised learning (Olshausen and Field, 1997; Lee and Seung, 1999; Ranzato et al., 2006; Lee et al., 2008). Sparse representations are often more easy to interpret, and also more robust to noise. Furthermore, evidence suggests that they may be used in biological systems. Recent sparse "energy-based methods" (Ranzato et al., 2006, 2007, 2008) have proposed the use of sparsity as an alternative to contrastive divergence learning. The "contrastive term" in CD (which represents the role of the partition function) corresponds to pulling up on the energy (or pushing down on the probability) of points outside the training set. Another way to ensure that the energy surface is low only around the training set is to eliminate the partition function and replace it with a term that limits the volume of the input space over which the energy surface can take low value (Ranzato et al., 2008). Using sparse overcomplete latent representations is a means of limiting this volume by minimizing the information content of the latent representation. Using both a contrastive term and sparsity, as we have done here, is a two-fold approach to sculpting energy

surfaces.

To implement sparsity, we maintained a damped "average activation" estimate for each hidden unit. Each element of this vector was initialized to the target activation, 0.2. Every time we presented a mini-batch, we updated the estimate to be 0.9 times its current value plus 0.1 times the average activation of the hidden units while the visible units were clamped to the data. The average was taken over the mini-batch. After we calculated the positive-phase (data) and negative-phase (after K steps of Gibbs sampling) statistics for each parameter, we added the gradient of the cross-entropy error between the updated activity estimate and the target, 0.2, with respect to that parameter. Note that updates for visible-only parameters (e.g. autoregressive weights and visible biases) were unaffected by the sparsity term. For more details, see (Lee et al., 2008). Note that these authors used a squared-error penalty between average activation and target while we used cross-entropy error.

1-layer Model

A single-layer CRBM with 1200 hidden units and $N = 12$ was trained on the 10-style data for 200 epochs with the parameters being updated after every 100 training cases. Each training case was a window of 13 consecutive frames and the order of the training cases was randomly permuted. In addition to the real-valued mocap data, the hidden units received input from a "one-hot" encoding of the matching style label. Respecting the conditional nature of our application (generation of stylized motion, as opposed to, say classification) this label was not reconstructed during learning. After training the model, we generated motion by initializing with 12 frames of training data and holding the label units clamped to the style matching the initialization.

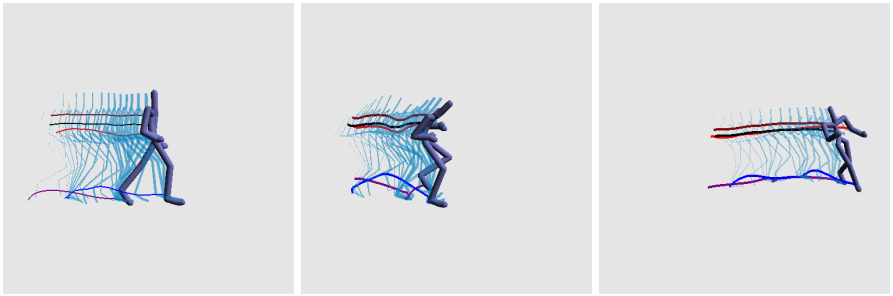
With a single layer we could generate high-quality motion of 9/10 styles (see the supplemental videos), however, the model failed to produce good generation of the *old-man* style. We believe that this relates to the subtle nature of this particular motion. In examining the activity of the hidden units over time while clamped to training data, we observed that the model devotes most of its hidden capacity to capturing the more "active" styles as it pays a higher cost for failing to model more pronounced frame-to-frame changes.

2-layer Model

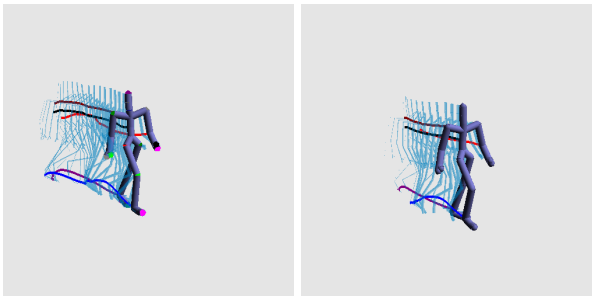
We also learned a deeper network by first training a CRBM with 600 binary hidden units and real-valued visible units and then training a higher-level CRBM with 600 binary hidden and 600 binary visible units. The data for training the higher-level CRBM was the activations of the hidden units of the first CRBM while driven by the training data. Style labels were only connected to the top-layer of this network, while training the second level CRBM. The first model was trained for 300 epochs, and the second level was trained for 120 epochs. After training, the 2-hidden-layer network was able to generate high-quality walks of all styles, including *old-man* (see see Figure 4.7 and the supplemental videos). The second level CRBM layer effectively replaces the prior over the first layer of hidden units, $p(\boldsymbol{h}_t|\boldsymbol{v}_{<t})$, that is implicitly defined by the parameters of the first CRBM. This provides a better model of the subtle correlations between the features that the first-level CRBM extracts from the motion.

4.5.5 Filling in missing data

Due to the nature of the motion capture process, which can be adversely affected by lighting and environmental effects, as well as noise during recording, motion capture data often contains missing or unusable data. Some markers may disappear ("dropout") for long periods of time due to sensor failure or occlusion. The majority of motion editing software packages contain interpolation methods to fill in missing data, but this leaves the data unnaturally smooth. These methods also rely on the starting and end points of the missing data. Hence, if a marker goes missing until the end of a sequence, naïve interpolation will not work. Such methods often only use the past and future data from the single missing marker to fill in that marker's missing values. Since joint angles are highly correlated, substantial information about the placement of one marker could be gained from the others. To demonstrate filling in, we trained a model exactly as described in Sec. 4.5.1, holding out one walking and one running sequence from the training data to be used as test data. For each of these walking and running test sequences, we erased two different sets of joint angles, starting halfway through the test sequence. These sets were the joints in (1) the left leg, and (2) the entire upper body.



(a) Cat (b) Dinosaur (c) Graceful



(d) Sexy (e) Strong

Figure 4.7. Generating different walking styles from the same Conditional Deep Belief Network with two hidden layers.

As seen in the video files on the website, the quality of the filled-in data is excellent and is hardly distinguishable from the original ground truth of the test sequence. Figure 4.8 demonstrates the model's ability to predict the three angles of rotation of the left hip.

We report results on the held-out walking sequence, of length 124 frames. We compared our model's performance to nearest neighbour interpolation, a simple method where for each frame, the values on known dimensions are compared to each example in the training set to find the closest match (measured by Euclidean distance in the normalized angle space). The unknown dimensions are then filled in using the matched example. As reconstruction from our model is stochastic, we repeated the experiment 100 times and report the mean. For the missing leg, mean squared reconstruction error per joint using our model was 8.78, measured in normalized joint angle space, and summed over the 62 frames of interest. Using nearest neighbour interpolation,

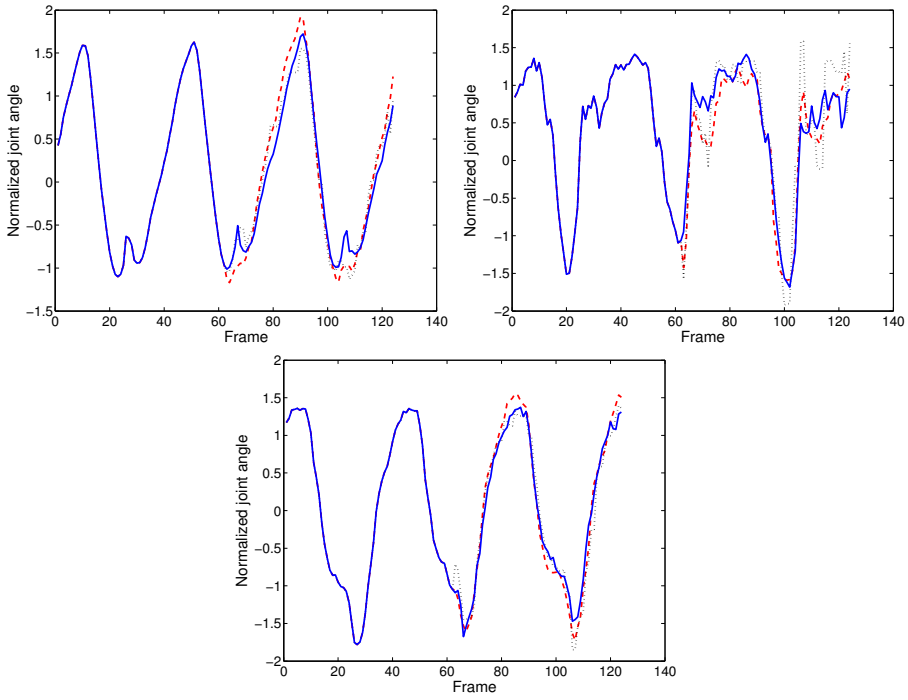


Figure 4.8. The model successfully fills in missing data using only the previous values of the joint angles (through the temporal connections) and the current angles of other joints (through the symmetric connections). Shown are the three angles of rotation for the left hip joint. The original data is shown on a solid line, the model's prediction is shown on a dashed line, and the results of nearest neighbour interpolation are shown on a dotted line.

the error was greater: 11.68. For the missing upper body, mean squared reconstruction error per joint using our model was 20.52. Using nearest neighbour interpolation, again the error was greater: 22.20.

We note that by adding additional neighbouring points, the nearest neighbour prediction can be significantly improved. For filling in the left leg, we found that $K = 8$ neighbours gave minimal error (8.63), while for the missing upper body, using $K = 6$ neighbours gave minimal error (12.67). These scores, especially in the case of the missing upper body, are, in fact, an improvement over using the CRBM for prediction. However, we note that in practice we would not be able to fine-tune the number of nearest neighbours nor

could we be expected to have access to a large database of extremely similar training data. In more realistic missing-data scenarios, we would expect the model-based approach to generalize much better. Furthermore, we have not optimized other tunable parameters such as the model order, number of Gibbs steps per CD iteration, and number of hidden units; all of which are expected to have an impact on the prediction error.

4.5.6 Video textures

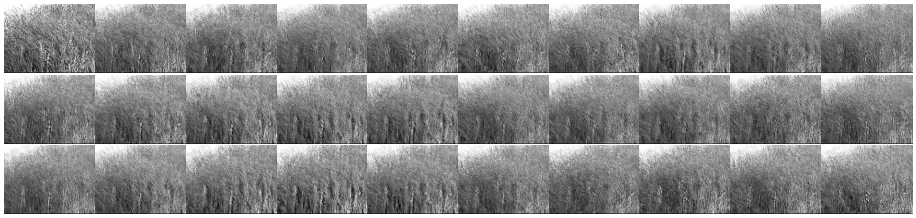
To demonstrate that our model is applicable to high-dimensional data beyond mocap, we have experimented with the synthesis of video textures (Schödl et al., 2000). The goal of video texture synthesis is to create new, longer video clips from existing data, where the originals are usually of a limited duration and repetitive nature. Simply looping sections of existing video, while producing realistic output frames, is noticeable to the viewer. Schödl et al. (2000) proposed a heuristic approach to finding sensible transition points between frames in an input sequence. By defining a matrix of transition probabilities, the authors were able to generate a random, but appropriate, re-ordering of the input. Using the same transition matrix, they could also construct loops that repeated within a fixed period.

A model-based approach offers many advantages over simply re-ordering the input. For example, with a model textures can be edited, classified or compressed. A popular approach has been to project the video frames onto an eigenspace using PCA and then model the lower-dimensional subspace, for example, as a linear-Gaussian low order autoregressive process (Doretto et al., 2003). Such an approach often leads to poor results, as the low-dimensional sequences can be nonlinearly correlated, exhibit non-Gaussian distributions and depend on events other than the recent past. Campbell et al. (2004) address this problem by decomposing the principal component vectors using a spline fitting or combined shape/texture appearance model into a representation having Gaussian statistics. We avoid the complexity of such an approach, and instead, apply our nonlinear model directly to the representation found by PCA.

We experimented with two video sequences, one containing color information and the other grayscale. The first video, “Grass”, had 148 frames of 144×192 pixels each. The second video, “Flame”, had 101 frames of 400×210

pixels each. On each video, we performed PCA and projected the video onto the eigenspace spanned by the first 40 principal component vectors (which captured almost all of the variance). Similar to our motion experiments, we chose $N = 3$, but used 150 hidden units because of the small number of frames. After the first layer was trained, in each model, we trained a second hidden layer with $N = 3$ and 150 hidden units, as described in Sec. 4.3. 400-frame sequences were then generated from the trained models and can be viewed on the website. Figures 4.9 and 4.10 show sample frames from the synthesized video. Subjectively, the video textures generated by our model are impressive. This illustrates, qualitatively, that without designing a specific model for the texture synthesis problem, the CRBM can be easily applied to new domains.

(a) Original

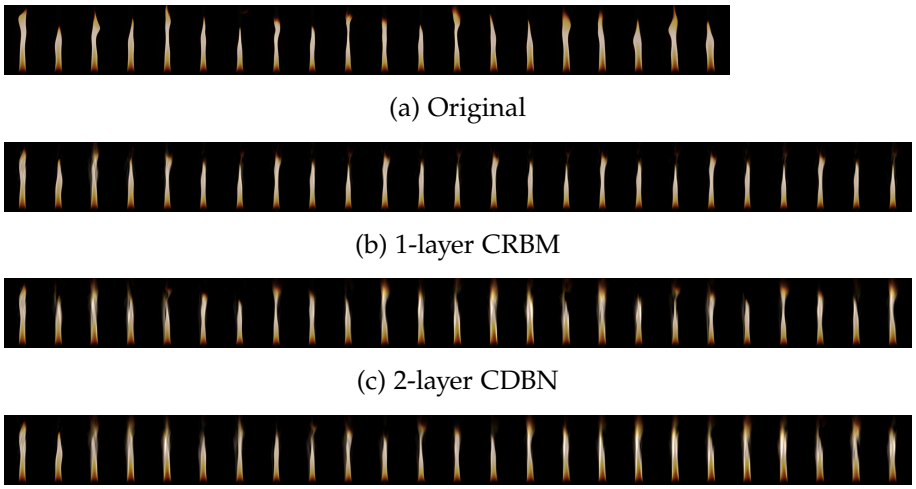


(b) 2-layer CDBN

Figure 4.9. Synthesis of "Grass" video texture. We show every 5th frame (0.2s).

4.6 Forward and backward models

The CRBM can only fill in missing data on-line because it does not do smoothing. One may, however, want to fill in missing data so that it agrees with both previous and future frames. As well, animators may wish to constrain sequences at two or more points and then let the model fill in data



(a) Original

(b) 1-layer CRBM

(c) 2-layer CDBN

(d) 2-layer CDBN, fine-tuned

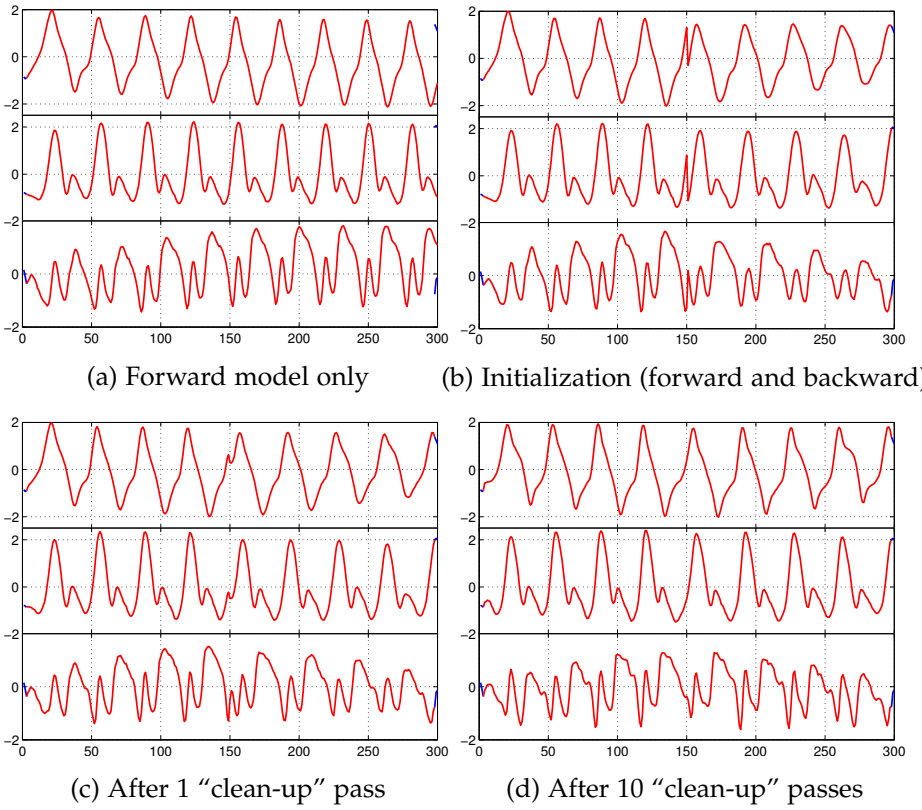
Figure 4.10. Synthesis of "Flame" video texture. We show every 5th frame (0.2s).

naturally between these "keyframes"[†]. By training both a model as described above (called the "forward" model), and another model on the reversed data (called the "backward" model), we can accomplish this task. However, we must share the undirected weights between both models to ensure consistency in the hidden features. The individual models are used to initialize a sequence, then a "clean-up" phase repeatedly averages the contributions made by the biases and directed connections of each model so that each frame agrees with both the past and future.

To provide an example of doubly-constrained generation, we trained a third-order CRBM with 400 hidden units on 3 walking sequences from the MIT database (file `Normal1_M`); a total of 3826 frames at 30Hz. Then we held the symmetric weights, W, fixed while training the other parameters of an identical CRBM on the reverse of the data. Each model was trained for 2000 epochs, and we verified that both could generate walking sequences, given a suitable initialization. We then chose two sets of three sequential frames, representing the beginning and end of a sequence. Nothing was special about

[†]Note that the CRBM has conditional links on a few frames, thus keyframes in this sense are not static poses, but a collection of poses from which velocities could be inferred.

the frames that we chose, except that they were chosen sufficiently out of phase so that generating 300 frames with a forward model alone (Figure 4.11a) resulted in a sequence that did not align with the specified end of the sequence.



(a) Forward model only (b) Initialization (forward and backward)

(c) After 1 "clean-up" pass (d) After 10 "clean-up" passes

Figure 4.11. Doubly-constrained generation (over 300 frames) with third-order "forward" and "backward" CRBMs. Frames 1:3 and 298:300 are provided as initialization (shown in blue). Top row: 1 dof from left femur. Middle row: 1 dof from left tibia. Bottom row: 1 dof from left foot. Units for all y-axes are normalized exponential map components.

We initialized the doubly-constrained generation by generating frames 4-150 with the forward model and 151-297 with the backward model (Figure 4.11b). We can see that the motion is now smooth near the clamped endpoints, but a discontinuity exists between frame 150-151. Figure 4.11c shows the result

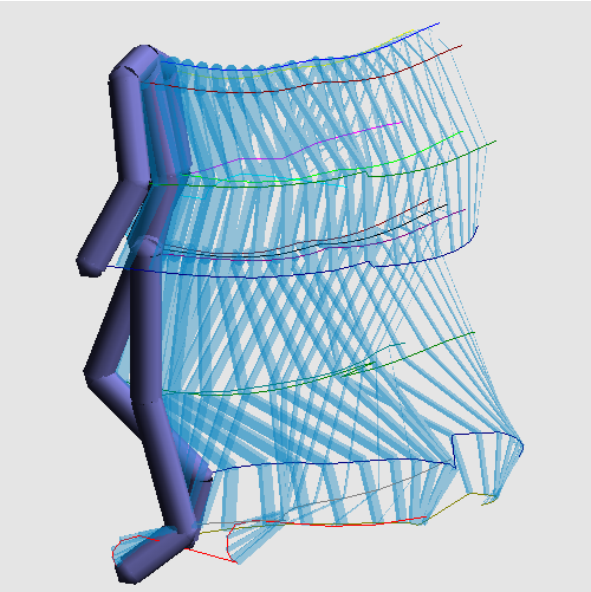
after a single pass through the entire sequence, where at each frame (except 1-3 and 298-300 which are never updated) we performed alternating Gibbs sampling where the inputs to the hidden and visible were the average of the input coming from the forward and backward models. We only used 10 Gibbs steps for the clean-up stage. After a single "clean-up" pass the trajectories are much smoother but the discontinuity is still present. Figure 4.11d shows the result after 10 passes through the sequence. The resulting trajectories are now smooth. Videos on the website demonstrate the motion after each stage. The motion is also visualized in Figure 4.12 where we compare the result before and after the "clean-up" phase.

4.7 Discussion

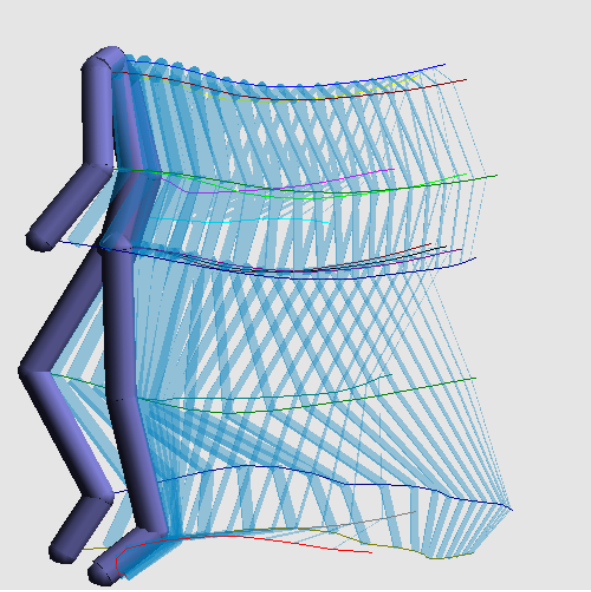
We have introduced a generative model for multivariate data based on the idea that local constraints and global dynamics can be learned efficiently by a Conditional Restricted Boltzmann Machine. Once trained, our models are able to efficiently capture complex nonlinearities in the data without sophisticated pre-processing or dimensionality reduction. The model has been designed with human motion in mind, but should lend itself well to other high-dimensional time series. In this chapter we have assumed that the input, $\boldsymbol{v}_{<t}$, is a vector of the N previous observations. The CRBM, however, permits much more general conditioning. We could, for instance, condition on a *function* of the recent past[†], or condition on external variables.

In relatively low-dimensional or unstructured data (for example if we were to model a single isolated joint) a single-layer model might be expected to have difficulty since such cyclic time series contain several subsequences which are locally very similar but occur in different phases of the overall cycle. It would be possible to preserve the global phase information by using a much higher order model, but for higher dimensional data such as full body motion capture this is unnecessary because the whole configuration of joint angles and angular velocities never has any phase ambiguity. Therefore, the single-layer version of our model actually performs much better on higher-dimensional data. Models with more hidden layers are able to implicitly model longer-term

[†]Or learn this function by adding a layer of hidden units connected to the past and backpropagating the gradients computed by contrastive divergence.



(a) Initialization (forward and backward)



(b) After 10 "clean-up" passes

Figure 4.12. Frames 140-160 of doubly-constrained generation. (a) The discontinuity is apparent in the initialization, where frames 140-150 are generated by the "forward" model, and 151-160 by the "backward" model. (b) We obtain much smoother motion after performing 10 passes through the sequence where predictions are made by combining the input from both models.

temporal information, and thus will mitigate this effect.

We have demonstrated that our model can effectively learn different styles of motion, as well as the transitions between these styles. This differentiates our approach from PCA-based approaches which only accurately model cyclic motion, and additionally must build separate models for each type of motion. The ability of the model to transition smoothly, however, is dependent on having sufficient examples of such transitions in the training data. If we augment the data with some static skeletal and identity parameters (in essence mapping a person's unique identity to a set of features), we should be able to use the same generative model for many different people, and generalize individual characteristics from one type of motion to another. Finally, our model is not limited to a single source of data. In the future, we hope to integrate low-level vision data captured at the same time as motion, thus learning the correlations between the vision stream and the joint angles.

5 Factored Conditional Restricted Boltzmann Machines

In this chapter we present a new model, based on the CRBM, that explicitly captures contextual information. Our model preserves the CRBM's most important computational properties and includes multiplicative three-way interactions that allow the effective interaction weight between two units to be modulated by the dynamic state of a third unit. We factor the three-way weight tensor implied by the multiplicative model, reducing the number of parameters from $O(N^3)$ to $O(N^2)$. Again we demonstrate our model's effectiveness by modeling human motion. The three-way interactions greatly improve the model's ability to blend motion styles or to transition smoothly among them.

5.1 Multiplicative interactions

A major motivation for the use of RBMs is that they can be used as the building blocks of deep belief networks (DBN), which are learned efficiently by training greedily, layer-by-layer (see Sec. 4.3). DBNs have been shown to learn very good generative models of handwritten digits (Hinton et al., 2006), but they fail to model patches of natural images. This is because RBMs have difficulty in capturing the smoothness constraint in natural images: a single pixel can usually be predicted very accurately by simply interpolating its neighbours.

To address this concern, Osindero and Hinton (2008) introduced the Semi-restricted Boltzmann Machine (SRBM). In an SRBM, the constraints on the

connectivity of the RBM are relaxed to allow lateral connections between the *visible* units in order to model the pair-wise correlations between inputs, thus allowing the hidden units to focus on modeling higher-order structure. Semi-restricted Boltzmann machines also permit deep networks. Each time a new level is added, the previous top layer of units is given lateral connections, so, after the layer-by-layer learning is complete, all layers except the topmost contain lateral connections between units. SRBMs make it possible to learn deep belief nets that model image patches much better, but they still have strong limitations that can be seen by considering the overall generative model. The equilibrium sample generated at each layer influences the layer below by *controlling its effective biases*. The model would be much more powerful if the equilibrium sample at the higher level could *control the lateral interactions* at the layer below using a three-way, multiplicative relationship. Memisevic and Hinton (2007) introduced the gated CRBM, which permitted such multiplicative interactions and was able to learn rich distributed representations of image transformations (see Sec. 5.3).

In this chapter, we explore the idea of multiplicative interactions in the context of a different type of CRBM. Instead of gating lateral interactions with hidden units, we allow a set of context variables to gate the three types of connections (which we call "sub-models") within the CRBM (Fig. 4.1). Our modification of the CRBM architecture does not change the desirable properties related to inference and learning but makes the model context-sensitive.

Like the CRBM, the multiplicative model is applicable to general time series where conditional data is available (e.g. seasonal variables for modeling rainfall occurrences, economic indicators for modeling financial instruments). However, we are largely motivated by our success thus far in modeling mocap data. In Chapter 4 we showed that a CRBM could capture many different styles with a single set of parameters. Generation of different styles was purely based on initialization, and the model architecture did not allow control of transitions between styles nor did it permit style blending. By using style variables to gate the connections of a CRBM, we can obtain a much more powerful generative model that permits controlled transitioning and blending. We demonstrate that in a conditional model, the gating approach is superior to simply using labels to bias the hidden units, which is the approach most

commonly used in static models (Hinton et al., 2006).

5.2 Style and content separation

This chapter is also part of a larger body of work related to the separation of style and content in motion. The ability to separately specify the style (e.g. sad) and the content (e.g. walk to location A) is highly desirable for animators. One approach to style and content separation is to guide a factor model (e.g. PCA, Factor Analysis, ICA) by giving it "side-information" related to the structure of the data. Tenenbaum and Freeman (2000) considered the problem of extracting exactly two types of factors, namely style and content, using a bilinear model. In a bilinear model, the effect of each factor on the output is linear when the other is held fixed, but together the effects are multiplicative. This model can be learned efficiently, but supports only a rigid, discrete definition of style and content requiring that the data be organized in a (style $\times$ content) grid.

Previous work has looked at applying user-specified style to an existing motion sequence (Urtasun et al., 2004; Hsu et al., 2005; Torresani et al., 2007). The drawback to these approaches is that the user must provide the content. We propose a generative model for content that adapts to stylistic controls. Recently, models based on the Gaussian Process Latent Variable Model (Lawrence, 2004) have been successfully applied to capturing style in human motion (Wang et al., 2007). The advantage of our approach over such methods is that our model does not need to retain the training dataset (just a few frames for initialization). The advantages of building compact models have been discussed in the previous chapter. Furthermore, training time increases linearly with the number of frames of training data, and so our model can scale up to massive datasets, unlike the kernel-based methods which are cubic in the number of frames. The powerful distributed hidden state of our model means that it does not suffer from the limited representational power of HMM-based methods (e.g. Brand and Hertzmann, 2000).

5.3 Gated Conditional Restricted Boltzmann Machines

Memisevic and Hinton (2007) introduced a way of implementing multiplicative interactions in a conditional model. The gated CRBM was developed

in the context of learning transformations between image pairs. The idea is to model an observation (the output) given its previous instance (the input) (e.g. neighbouring frames of video). The gated CRBM has two equivalent views: first, as gated regression (Fig. 5.1a), where hidden units can blend "slices" of a transformation matrix into a linear regression, and second as modulated filters (Fig. 5.1b) where input units gate a set of basis functions used to reconstruct the output. In the latter view, each setting of the input units defines an RBM (which means that conditional on the input, inference and learning in a gated CRBM are tractable). For ease of presentation, let us

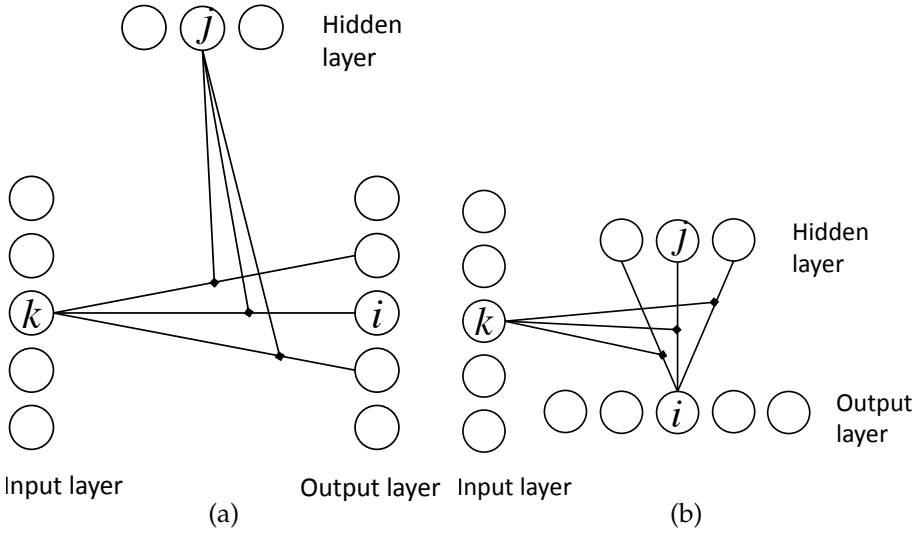


Figure 5.1. Two views of the gated CRBM. Reproduced from (Memisevic and Hinton, 2007).

consider the case where all input, output, and hidden variables are binary (the extension to real-valued input and output variables is straightforward). As in Eq. 4.13, the gated CRBM describes a joint probability distribution through exponentiating and renormalizing an energy function. This energy function captures all possible correlations between the components of the input, $\boldsymbol{x}$, the output, $\boldsymbol{v}$, and the hidden variables, $\boldsymbol{h}$:

$$E\left(\boldsymbol{v},\boldsymbol{h}|\boldsymbol{x}\right) = -\sum_{ijk} W_{ijk} v_i h_j x_k - \sum_{ij} c_{ij} v_i h_j - \sum_i a_i v_i - \sum_j b_j h_j \qquad (5.1)$$

where a_i, b_j index the standard biases on each unit and c_{ij} index the gated biases, which shift a unit conditionally. The parameters W_{ijk} are the components of a three-way weight tensor. The CD weight updates for learning a gated CRBM are similar to a standard CRBM. For example, the weight update rule for W_{ijk} is:

$$\Delta W_{ijk} \propto \langle v_i h_j x_k \rangle_{\text{data}} - \langle v_i h_j x_k \rangle_{\text{recon}}. \tag{5.2}$$

Considering the "modulated filters" view of the gated CRBM, we can fold the (given) inputs into the weights to express the input-dependent filters $\hat{W}_{ij} = \sum_k W_{ijk} x_k$. This allows us to rewrite the energy function (Eq. 5.1) as:

$$E(\boldsymbol{v}, \boldsymbol{h} | \boldsymbol{x}) = -\sum_{ij} \hat{W}_{ij} v_i h_j - \sum_{ij} c_{ij} v_i h_j - \sum_i a_i v_i - \sum_j b_j h_j. \tag{5.3}$$

Fixing the input, the first term is bilinear in $\boldsymbol{v}$ and $\boldsymbol{h}$. Therefore at first glance, the model appears similar to the bilinear factor model (Tenenbaum and Freeman, 2000). However, the two models differ considerably in both their learning method and structure. Note that the bilinearity only occurs in the energy function: the gated CRBM permits the learned transformations to be highly nonlinear functions of the data.

5.4 Factoring

To model time-series, we can consider the output of a gated CRBM to be the current frame of data, $\boldsymbol{v} = \boldsymbol{v}_t$, and the input to be the previous frame (or frames), $\boldsymbol{x} = \boldsymbol{v}_{<t} = \boldsymbol{v}_{t-N:t-1}$. In this sense, the gated CRBM is a kind of autoregressive model where a transformation is composed from a set of simpler transformations. The number of possible compositions is exponential in the number of hidden units, but the componential nature of the hidden units prevents the number of parameters in the model from becoming exponential, as it would in a mixture model. Because of the three-way weight tensor, the number of parameters is cubic (assuming that the numbers of input, output and hidden units are comparable).

In many applications, including human motion modeling, strong underlying regularities in the data suggest that structure can be captured using three-way, multiplicative interactions but with less than the cubically many parameters implied by the weight tensor. This motivates us to factor the interaction tensor into a product of pairwise interactions (Figure 5.2). Factoring

changes the energy function (Eq. 5.1) to:

$$E\left(\boldsymbol{v}, \boldsymbol{h} | \boldsymbol{x}\right) = -\sum_f \sum_{ijk} W_{if}^{\boldsymbol{v}} W_{jf}^{\boldsymbol{h}} W_{kf}^{\boldsymbol{x}} v_i h_j x_k - \sum_{ij} c_{ij} v_i h_j - \sum_i a_i v_i - \sum_j b_j h_j \quad (5.4)$$

where f indexes a set of deterministic factors. Superscripts differentiate the different types of pairwise interactions: $W_{if}^{\boldsymbol{v}}$ connect output units to factors (undirected), $W_{jf}^{\boldsymbol{h}}$ connect hidden units to factors (undirected), and $W_{kf}^{\boldsymbol{x}}$ connect input units to factors (directed).

The factors correspond to an intermediate layer of "simple cells" which modulate the interactions between units. Each factor is connected to all input units, all hidden units, and all output units. However, there are typically less factors than each type of unit, so the introduction of factors corresponds to a kind of unconstrained low-rank approximation to the interaction tensor, W. Factors are deterministic, and are therefore very different than the visible and hidden units, which have stochastic states. Factors always send the product of the total input from two types of units to the remaining third type of unit. For example, during inference, each factor collects the total input arriving at it from the input and output layers, respectively, multiplies these quantities together, and sends this input to each hidden unit. During reconstruction, each factor collects the total input arriving at it from the input and hidden layers, respectively, multiplies these quantities together, and sends this input to each visible unit. This is in contrast to the visible and hidden units, which must be sampled before sending their states to the factors.

If the number of factors is comparable to the number of other units, factoring reduces the number of parameters from $O(N^3)$ to $O(N^2)$. Although factoring has been motivated by the introduction of multiplicative interactions, models that only involve pairwise interactions can also be factored (e.g. Mnih and Hinton, 2007). To factor the CRBM, we change the energy function in Eq. 4.15 to:

$$E(\boldsymbol{v}_t, \boldsymbol{h}_t | \boldsymbol{v}_{<t}) = \frac{1}{2} \sum_i \left(v_{i,t} - \hat{a}_{i,t}\right)^2 - \sum_f \sum_{ij} W_{if}^{\boldsymbol{v}} W_{jf}^{\boldsymbol{h}} v_{i,t} h_{j,t} - \sum_i \hat{b}_{j,t} h_{j,t} \quad (5.5)$$

and additionally, factor the weights of the dynamic biases $\hat{\boldsymbol{a}}_t$ and $\hat{\boldsymbol{b}}_t$:

$$\hat{a}_{i,t} = a_i + \sum_m \sum_k A_{im}^{\boldsymbol{v}} A_{km}^{\boldsymbol{v}_{<t}} v_{k,<t}, \quad (5.6)$$

$$\hat{b}_{j,t} = b_j + \sum_n \sum_k B_{jn}^{\boldsymbol{h}} B_{kn}^{\boldsymbol{v}_{<t}} v_{k,<t}. \quad (5.7)$$

The indices m and n correspond to the factoring of directed connections, A and B. We may use a different number of factors for each of the three different types of connections in the CRBM. This procedure can be seen as a kind of learned low-rank matrix factorization on each of W,A, and B.

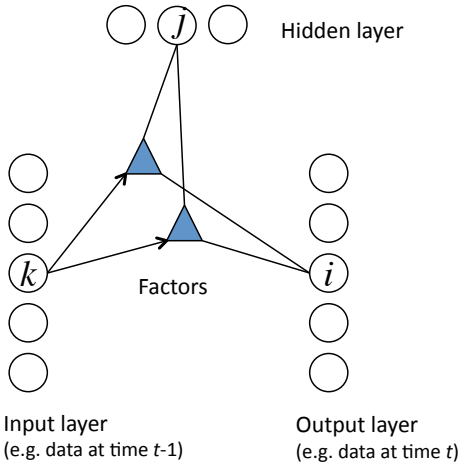
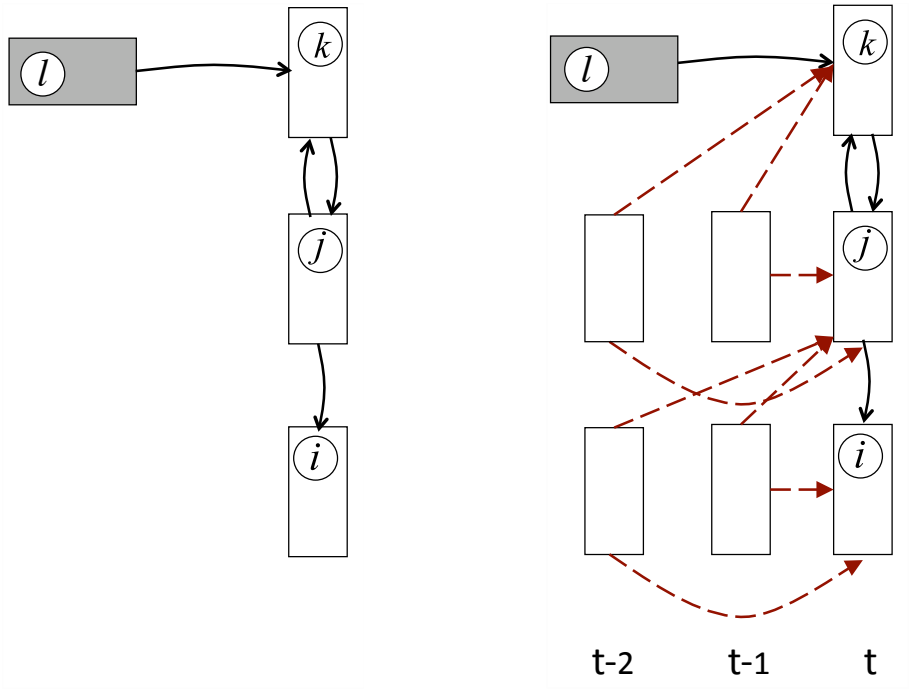


Figure 5.2. Factoring the gated CRBM.

5.5 A style-gated, factored model

We now consider modeling multiple styles of human motion using factored, multiplicative, three-way interactions. Hinton et al. (2006) showed that a good generative model of handwritten digits could be built by connecting a softmax label unit to the topmost hidden layer of a DBN (Fig. 5.3a). Clamping a label changed the energy landscape of the associative memory formed by the top two layers, such that performing alternating Gibbs sampling would produce a joint sample compatible with a particular digit class. It is easy to extend this modification to the CRBM, where discrete style labels bias the hidden units (see Sec. 4.5.4). In a CRBM, however, the hidden units also condition on information from the past that is much stronger than the information coming from the label (Fig. 5.3b). The model has learned to respect consistency of styles between frames and so will resist a transition introduced by changing the label units.



(a) (b)

Figure 5.3. a) In a deep belief network, clamping the label units changes the energy function. b) In a conditional model, label information is swamped by the signal coming from the past.

As in the gated CRBM, we are motivated to let style change the *interactions* of the units as opposed to simply their effective biases. Memisevic (2008) used factored three-way interactions to allow the hidden units of a gated CRBM to control the effect of one video frame on the subsequent video frame. Figure 5.4 shows a different way of using factored three-way interactions to allow real-valued style features, derived from discrete style labels, to control three different sets of pairwise interactions. Like the standard CRBM (Eq. 4.13), the model defines a joint probability distribution over $\boldsymbol{v}_t$ and $\boldsymbol{h}_t$, conditional on the past N observations, $\boldsymbol{v}_{<t}$. However, the distribution is also conditional on the style labels, $\boldsymbol{y}_t$. Similar to our discussion of the CRBM, we assume binary stochastic hidden units and real-valued visible units with additive Gaussian

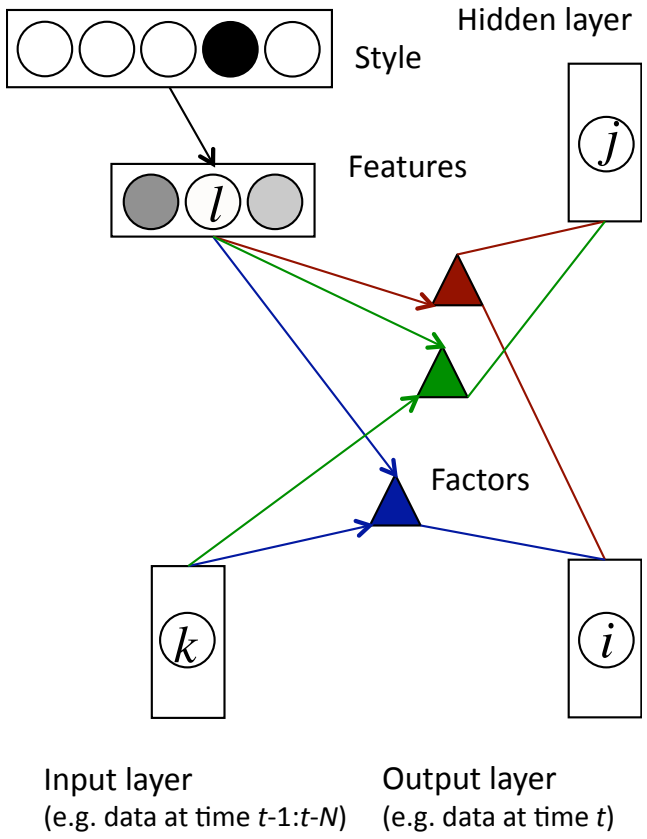


Figure 5.4. A factored CRBM whose interactions are gated by real-valued stylistic features.

noise and $\sigma_i = 1$. The energy function is:

$$E\left(\boldsymbol{v}_t, \boldsymbol{h}_t | \boldsymbol{v}_{<t}, \boldsymbol{y}_t\right) = \frac{1}{2}\sum_i \left(v_{i,t} - \hat{a}_{i,t}\right)^2 - \sum_f \sum_{ijl} W^{v}_{if} W^{\boldsymbol{h}}_{jf} W^{z}_{lf} v_{i,t} h_{j,t} z_{l,t} - \sum_j \hat{b}_{j,t} h_{j,t}. \tag{5.8}$$

The three terms in Eq. 5.8 correspond to the three sub-models (coloured blue, red, and green, respectively in Fig. 5.4). Note that for each sub-model, what was a matrix of weights is now replaced by three sets of weights connecting units to factors. The three types of weights are differentiated again by superscripts. For example, the matrix of undirected weights in the standard CRBM,

W_{ij}, has been replaced by three matrices involved in a factored, multiplicative interaction: $W^{\boldsymbol{v}}_{if}$, $W^{\boldsymbol{h}}_{jf}$, and $W^{\boldsymbol{z}}_{lf}$. The same process is applied to the other two sub-models. Note that the three sub-models may have a different number of factors (which we index by f, m, and n).

The dynamic biases become:

$$\begin{aligned}\hat{a}_{i,t} &= a_i + \sum_m \sum_{kl} A^{\boldsymbol{v}}_{im} A^{\boldsymbol{v}_{<t}}_{km} A^{\boldsymbol{z}}_{lm} v_{k,<t} z_{l,t} \\ &= a_i + \sum_m A^{\boldsymbol{v}}_{im} \sum_k A^{\boldsymbol{v}_{<t}}_{km} v_{k,<t} \sum_l A^{\boldsymbol{z}}_{lm} z_{l,t}, \qquad (5.9)\\ \hat{b}_{j,t} &= b_j + \sum_n \sum_{kl} B^{\boldsymbol{h}}_{jn} B^{\boldsymbol{v}_{<t}}_{kn} B^{\boldsymbol{z}}_{ln} v_{k,<t} z_{l,t} \\ &= b_j + \sum_n B^{\boldsymbol{h}}_{jn} \sum_k B^{\boldsymbol{v}_{<t}}_{kn} v_{k,<t} \sum_l B^{\boldsymbol{z}}_{ln} z_{l,t} \qquad (5.10)\end{aligned}$$

where the dynamic component of Eq. 5.9 and Eq. 5.10 is simply the total input to the visible/hidden unit via the factors. The total input is a three-way product between the input to the factors (coming from the past and from the style features) and the weight from the factors to the visible/hidden unit. The dynamic biases include a static component, $\boldsymbol{a}$ and $\boldsymbol{b}$. As in the gated CRBM, we could also add three types of gated biases, corresponding to the pairwise interactions in each of the sub-models. In our experiments, we have not used any gated biases.

The deterministic features, $\boldsymbol{z}_t$, are a linear function of the "one-hot" encoded style labels, $\boldsymbol{y}_t$:

$$z_{l,t} = \sum_p R_{pl} y_{p,t}. \qquad (5.11)$$

This resembles the use of componential word-features used in Mnih and Hinton's language model (Mnih and Hinton, 2007).

5.5.1 Inference and learning

Adding multiplicative interactions to the model and factoring does not change the property that the posterior distribution is factorial. Inference is performed by considering, in parallel, the total input to each hidden unit via the factors:

$$p(h_{j,t} = 1 | \boldsymbol{v}_t, \boldsymbol{v}_{<t}, \boldsymbol{y}_t) = \frac{1}{1 + \exp(-\hat{b}_{j,t} - \sum_f W^{\boldsymbol{h}}_{jf} \sum_f \sum_i W^{\boldsymbol{v}}_{ij} v_{i,t} \sum_l W^{\boldsymbol{z}}_{lf} z_{l,t})} \qquad (5.12)$$

where $\hat{b}_{j,t}$ is defined in Eq. 5.10. The reconstruction distribution is found by

considering the total input to each visible unit via the factors:

$$p(v_{i,t}|\boldsymbol{h}_t, \boldsymbol{v}_{<t}, \boldsymbol{y}_t) = \mathcal{N}\left(\hat{a}_{i,t} + \sum_f W^{\boldsymbol{v}}_{if} \sum_j W^{\boldsymbol{h}}_{jf} h_{j,t} \sum_l W^{\boldsymbol{z}}_{lf} z_{l,t}, 1\right) \tag{5.13}$$

where $\hat{a}_{i,t}$ is defined in Eq. 5.9.

As in the other models based on RBMs, exact maximum likelihood learning is intractable. However, applying contrastive divergence leads to a set of very simple gradient update rules which are the same for binary or real-valued Gaussian visible units. The gradient with respect to a weight that connects a unit to a factor is the difference of two expectations of products. Each product involves three terms: the activity of the respective unit, and the total input to the factor from each of the two other sets of units involved in the three-way relationship:

$$\Delta W^{\boldsymbol{v}}_{if} \propto \sum_t \left(\langle v_{i,t} \sum_j W^{\boldsymbol{h}}_{jf} h_{j,t} \sum_l W^{\boldsymbol{z}}_{lf} z_{l,t}\rangle_{\text{data}} - \langle v_{i,t} \sum_j W^{\boldsymbol{h}}_{jf} h_{j,t} \sum_l W^{\boldsymbol{z}}_{lf} z_{l,t}\rangle_{\text{recon}}\right), \tag{5.14}$$

$$\Delta W^{\boldsymbol{h}}_{jf} \propto \sum_t \left(\langle h_{j,t} \sum_i W^{\boldsymbol{v}}_{if} v_{i,t} \sum_l W^{\boldsymbol{z}}_{lf} z_{l,t}\rangle_{\text{data}} - \langle h_{j,t} \sum_i W^{\boldsymbol{v}}_{if} v_{i,t} \sum_l W^{\boldsymbol{z}}_{lf} z_{l,t}\rangle_{\text{recon}}\right), \tag{5.15}$$

$$\Delta W^{\boldsymbol{z}}_{lf} \propto \sum_t \left(\langle z_{l,t} \sum_i W^{\boldsymbol{v}}_{if} v_{i,t} \sum_j W^{\boldsymbol{h}}_{jf} h_{j,t}\rangle_{\text{data}} - \langle z_{l,t} \sum_i W^{\boldsymbol{v}}_{if} v_{i,t} \sum_j W^{\boldsymbol{h}}_{jf} h_{j,t}\rangle_{\text{recon}}\right), \tag{5.16}$$

$$\Delta A^{\boldsymbol{v}}_{im} \propto \sum_t \left(\langle v_{i,t} \sum_k A^{\boldsymbol{v}_{<t}}_{km} v_{k,<t} \sum_l A^{\boldsymbol{z}}_{lm} z_{l,t}\rangle_{\text{data}} - \langle v_{i,t} \sum_k A^{\boldsymbol{v}_{<t}}_{km} v_{k,<t} \sum_l A^{\boldsymbol{z}}_{lm} z_{l,t}\rangle_{\text{recon}}\right), \tag{5.17}$$

$$\Delta A^{\boldsymbol{v}_{<t}}_{km} \propto \sum_t \left(\langle v_{k,<t} \sum_i A^{\boldsymbol{v}}_{im} v_{i,t} \sum_l A^{\boldsymbol{z}}_{lm} z_{l,t}\rangle_{\text{data}} - \langle v_{k,<t} \sum_i A^{\boldsymbol{v}}_{im} v_{i,t} \sum_l A^{\boldsymbol{z}}_{lm} z_{l,t}\rangle_{\text{recon}}\right), \tag{5.18}$$

$$\Delta A^{\boldsymbol{z}}_{lm} \propto \sum_t \left(\langle z_{l,t} \sum_i A^{\boldsymbol{v}}_{im} v_{i,t} \sum_k A^{\boldsymbol{v}_{<t}}_{km} v_{k,<t}\rangle_{\text{data}} - \langle z_{l,t} \sum_i A^{\boldsymbol{v}}_{im} v_{i,t} \sum_k A^{\boldsymbol{v}_{<t}}_{km} v_{k,<t}\rangle_{\text{recon}}\right), \tag{5.19}$$

$$\Delta B^{\boldsymbol{h}}_{jn} \propto \sum_t \left(\langle h_{j,t} \sum_k B^{\boldsymbol{v}_{<t}}_{kn} v_{k,<t} \sum_l B^{\boldsymbol{z}}_{ln} z_{l,t}\rangle_{\text{data}} - \langle h_{j,t} \sum_k B^{\boldsymbol{v}_{<t}}_{kn} v_{k,<t} \sum_l B^{\boldsymbol{z}}_{ln} z_{l,t}\rangle_{\text{recon}}\right), \tag{5.20}$$

$$\Delta B^{\boldsymbol{v}_{<t}}_{kn} \propto \sum_t \left(\langle v_{k,<t} \sum_j B^{\boldsymbol{h}}_{jn} h_{j,t} \sum_l B^{\boldsymbol{z}}_{ln} z_{l,t}\rangle_{\text{data}} - \langle v_{k,<t} \sum_j B^{\boldsymbol{h}}_{jn} h_{j,t} \sum_l B^{\boldsymbol{z}}_{ln} z_{l,t}\rangle_{\text{recon}}\right), \tag{5.21}$$

$$\Delta B^{\boldsymbol{z}}_{ln} \propto \sum_t \left(\langle z_{l,t} \sum_j B^{\boldsymbol{h}}_{jn} h_{j,t} \sum_k B^{\boldsymbol{v}_{<t}}_{kn} v_{k,<t}\rangle_{\text{data}} - \langle z_{l,t} \sum_j B^{\boldsymbol{h}}_{jn} h_{j,t} \sum_k B^{\boldsymbol{v}_{<t}}_{kn} v_{k,<t}\rangle_{\text{recon}}\right). \tag{5.22}$$

The weights connecting labels to features, R, can simply be learned by backpropagating the gradients obtained by CD. Since these weights affect all

three sub-models, their updates are more complicated. Applying the chain rule, we obtain:

$$\begin{aligned}
\Delta R_{pl} &\propto \sum_t \left(\langle C_{l,t} y_{p,t} \rangle_{\text{data}} - \langle C_{l,t} y_{p,t} \rangle_{\text{recon}} \right), \\
C_{l,t} &= \sum_f W^z_{lf} \sum_i W^v_{if} v_{i,t} \sum_j W^{\boldsymbol{h}}_{jf} h_{j,t} + \sum_m A^z_{lm} \sum_i A^v_{im} v_{i,t} \sum_k A^{v_{<t}}_{km} v_{k,<t} \\
&\quad + \sum_n B^z_{ln} \sum_j B^{\boldsymbol{h}}_{jn} h_{j,t} \sum_k B^{v_{<t}}_{kn} v_{k,<t}. \qquad (5.23)
\end{aligned}$$

The updates for the static biases on the hidden and visible biases are the same as in the standard CRBM (Eq. 4.11 and 4.12).

5.5.2 Parameter sharing

In addition to the massive reduction in the number of free parameters obtained by factoring, further savings may be obtained by tying some sets of parameters together. In the fully parameterized model (Fig. 5.5a), there are 9 different sets (matrices) of weights but if we restrict the number of factors to be the same for each of the three sub-models, four sets of parameters are identical in dimension: the weights that originate from the inputs (past visible units), the outputs (visible units), the hidden units and the features. Any combination of the compatible parameters may be tied. Fig. 5.5b shows a fully-shared parameterization. This has slightly less than half the number of parameters of the fully parameterized model, assuming that the number of input, output, hidden, and feature units are comparable.

In comparing different reduced parameterizations, tying only the feature-factor parameters, W^z_{lf},A^z_{lm}, and B^z_{ln} led to models that synthesized the highest quality motion. When sharing the autoregressive weights $A^{v_{<t}}_{km}$ and A^v_{im} with non-autoregressive weights $B^{v_{<t}}_{km}$ and W^v_{if}, respectively, we found that the component of the gradient related to the autoregressive model tended to dominate the weight update early in learning. This was due to the strength of the correlation between past and present compared to hidden and present or hidden and past. Witholding the autoregressive component of the gradient for the first 100 epochs, until the hidden units were able to extract interesting structure from the data, solved this problem. In our reported experiments we trained models with only the feature-factor parameters tied.

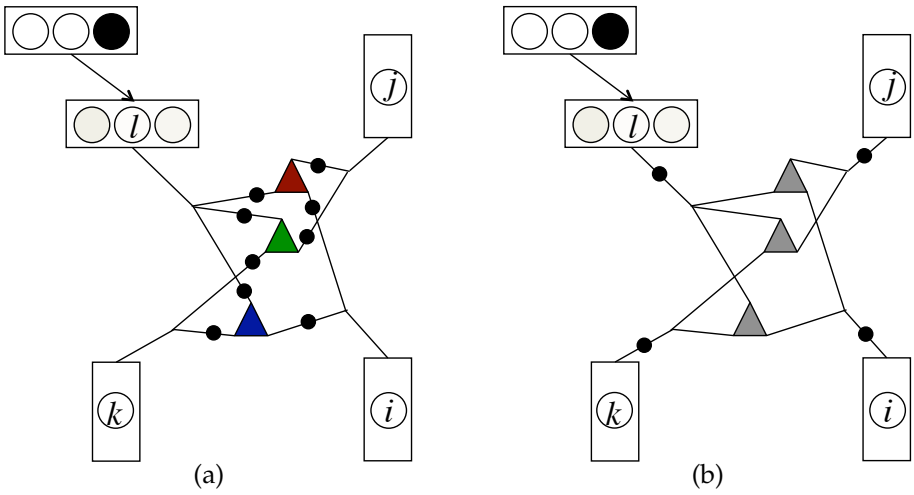


Figure 5.5. a) Fully parameterized model with each dot representing a different set of parameters and different colors denoting a different number of factors in each sub-model. b) Full parameter sharing where each dot represents a tied group of parameters. The number of factors is restricted to be the same for each sub-model.

5.6 Experiments

CRBM models share a common deficiency: biasing the hidden units with a style label is not a true integration of context into their architecture. Despite our attempts, we cannot prevent spurious transitions (see Sec. 4.5.4), nor does a change of label during generation allow us to transition or blend between styles. We carry out a set of experiments that demonstrate that this shortcoming can be addressed by using factored, multiplicative interactions.

5.6.1 Modeling with discrete style labels

Using the 10-styles dataset described in Sec. 4.5.4, we trained a factored CRBM with Gaussian visible units whose parameters were gated by 100 real-valued features driven by discrete style labels (Fig. 5.4). This model had 600 hidden units, 200 factors per sub-model and $N = 12$. Feature-to-factor parameters were also tied between sub-models. All parameters used a learning rate of 10^{-2}, except for the autoregressive parameters, A^v_{im}, $A^{v_{<t}}_{km}$, A^z_{lm} and the label-

to-feature parameters, R_{pl}, which used a learning rate of 10^{-3}. After training the model for 500 epochs, we tested its ability to synthesize realistic motion by initializing with 12 frames of training data and holding the label units clamped to the matching style. The single-layer model was able to generate stylized content as well as the 2-layer standard CRBM (see the supplemental videos). In addition, we were able to induce transitions between two or more styles by linearly blending the discrete style label from one setting to another over 200 frames[†]. We were further able to blend together styles (like *sexy* and *strong*) by applying a linear interpolation of the discrete labels. The resulting motion was more natural when a single style was dominant (e.g. an 0.8/0.2 blend). We believe this is simply a case of better performance when the desired motion more closely resembles the cases present in the training data set, so training on a few examples of blends should greatly improve their generation.

5.6.2 Modeling with real-valued style parameters

The motions considered thus far have been described by a single, discrete label such as *gangly* or *drunk*. Motion style, however, can be characterized by multiple discrete labels or even continuous factors such as the level of flow, weight, time and space formally defined in Laban Movement Analysis (Torresani et al., 2007). In the case of multiple discrete labels, our real-valued feature units, z, can receive input from multiple categories of labels. For continuous factors of style, we can connect real-valued style units to the real-valued feature units, or we can simply gate the model directly by the continuous description of style.

To test this latter configuration, we trained a model exactly as in Sec. 5.6.1, but instead of gating connections with 100 real-valued feature units, we gated with 2 real-valued style descriptors that were conditioned upon at every frame. Again we trained with walking data, but the data was captured specifically for this experiment. One style unit represented the speed of walking and the other, the stride length. The training data consisted of nine sequences at 60fps, each approximately 6000 frames corresponding to the cross-product of (slow, normal, fast) speed and (short,normal,long) stride length. The corresponding

[†]The number of frames was selected empirically and provided a smooth transition, but the model is not sensitive to this number. A quick (e.g. frame-to-frame) change of labels will simply produce a "jerky" transition.

labels each had values of 1, 2 or 3. These values were chosen to avoid the special case of all gating units being set at zero and nullifying the effective weights of the model. The model was trained for 500 epochs.

After training, the model could, as before, generate realistic motion according to the nine discrete combinations of speed and stride-length with which it was trained based on initialization and setting the label units to match the labels in the training set. Furthermore, the model supported both interpolation and extrapolation along the speed and stride length axes and did not appear overly sensitive to initialization (see the supplemental videos).

5.6.3 Quantitative evaluation

In our experiments so far, we have sought a qualitative comparison to the CRBM, based on the realism of synthesized motion. We have also focused on the ability of a factored model with multiplicative interactions to synthesize transitions as well as interpolate and extrapolate between styles present in the training data set. The application does not naturally present a quantitative comparison, but in the past, other time series models have been compared by their performance on the prediction of either full or partial held-out frames (e.g. Wang et al., 2006; Lawrence, 2007). We use the dataset first proposed by Hsu et al. (2005) which consists of labeled sequences of seven types of walking: (*crouch*, *jog*, *limp*, *normal*, *side-right*, *sway*, *waddle*) each at three different speeds (*slow*, *medium*, *fast*). We preprocessed the data to remove missing or extremely noisy sections, and smoothed with a low-pass filter before downsampling from 120 to 30fps.

For each architecture: CRBM, factored CRBM, style-gated unfactored CRBM, and style-gated factored CRBM, we trained 21 different models on all style/speed pairs except one, which we held out for testing. Then, for each model, we attempted to predict every subsequence of length M in the test set, given the past $N = 6$ frames. We repeated the experiments for each architecture, each time reporting results averaged over the 21 models. Prediction could be performed by initializing with the previous frame and Gibbs sampling in the same way we generated, but this approach is subject to noise. We found that in all cases, integrating out the hidden units and following the gradient of the negative free energy with respect to the visible units gave less prediction error (see Sec. 4.2.3). We minimized the free energy

using conjugate-gradient descent initialized with the previous frame. The architectures were subject to different learning rates and so the number of epochs for which to train each model were determined by setting aside 10% of the training set for validation.

We have also included a sixth-order autoregressive model as a baseline. This corresponds to the CRBM model without hidden units, except that it is trained using least squares instead of contrastive divergence. Simpler baselines are also considered in Chapter 6.

Fig. 5.6 presents the results. With almost half the number of free parameters, the 600-60 factored model performed as well as the fully parameterized CRBM. Gating with style information gives an advantage in longer-term prediction because it prevents the model from gradually changing the style. The unfactored model with style information performed slightly worse than the factored model and was extremely slow to train (it took two days to train whereas the other models were each trained in a few hours). The baseline autoregressive model performed extremely well in the short term, but was quickly eclipsed by the latent variable models for $N > 5$.

5.7 Discussion

Restricted Boltzmann Machines have several attractive computational properties which carry through to the deeper architectures of which they form the core. From a generative model standpoint, however, these deep networks have a deficiency. Regardless of whether or not the layers below contain lateral interactions, sampling the higher layers can only determine the effective biases of the layer below. The gated CRBM is a model in which the binary hidden units influence the lateral interactions of the layer below, providing an exponential number of (non-independent) possible autoregressive models at a cost that is cubic in the number of parameters.

If we only let contextual information (like style) determine the effective hidden biases of a CRBM, the signal is swamped by the information coming from the past. However, if we allow for a three-way, multiplicative relationship like in the gated CRBM, context becomes a natural part of the model, determining the effective weights. The potential blow-up in the number of parameters implied by such a model is solved by factorizing the three-way

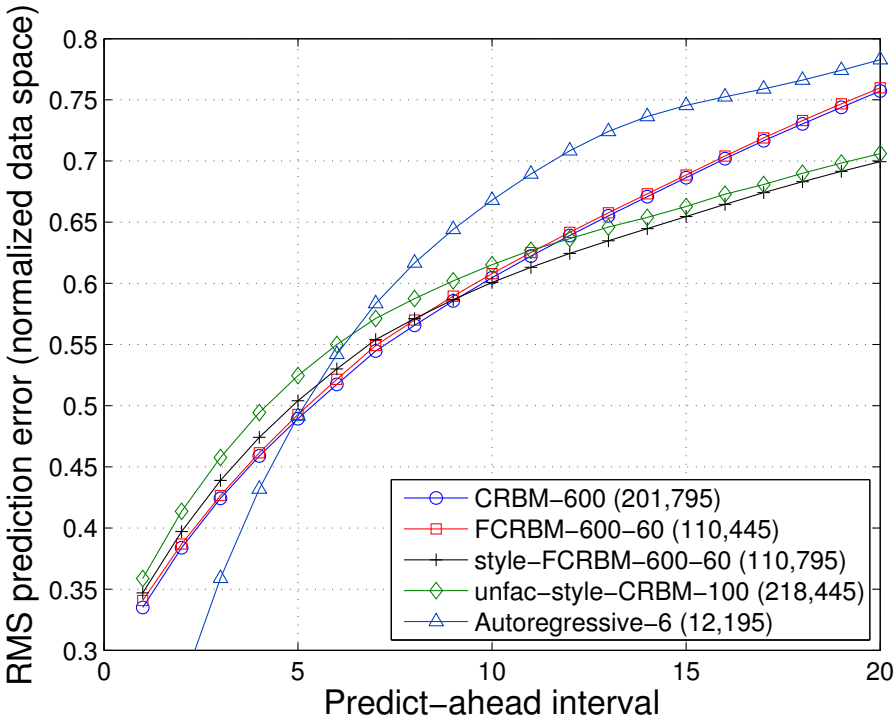


Figure 5.6. Prediction experiment. The number of free parameters are shown in parentheses. Error is reported in the normalized space in which the models are trained and is per-dimension, per-frame. The first two values for the autoregressive model (0.1506 and 0.2628) have been intentionally cut off.

weight tensors.

When modeling human motion, our approach permits style to change the effective weights of the network via discrete or real-valued representations. Changing these style-based factors during generation can induce natural-looking transitions and permit interpolation and extrapolation of styles in the training data. In our experiments we always conditioned on style, and assumed that our training data had been labeled *a priori*. This added a supervised flavour to our otherwise unsupervised models. We believe the more interesting problem is the fully unsupervised setting including the discovery of style in the layers of binary features. Future work will continue

to focus on higher-order interactions (perhaps beyond third) but where style-based descriptors are inferred rather than provided.

6 Tracking People in Video with Rich Dynamical Priors

Bayesian filtering is an elegant probabilistic framework which is often applied to the recovery of human pose and motion from video sequences. Existing methods typically use weak priors that fail to account for the complex dynamics and strong constraints associated with human motion. More powerful priors can facilitate efficient inference over pose in the presence of noisy image observations, occlusions and inherent ambiguities. In Chapter 4 we showed that the Conditional Restricted Boltzmann Machine (CRBM) could synthesize and denoise human motion using binary latent variables. By defining a joint probability distribution over pose and binary latent features given a past history of pose observations, they fit naturally into the Bayesian filtering framework. This chapter explores CRBMs as motion priors in the context of Bayesian filtering.

6.1 Introduction

Human motion estimation has played a major role in vision research over the past decade. This has mainly been driven by applications such as human-computer interfaces, motion analysis for medical applications, and video-based motion capture for surveillance or animation. Despite some moderate successes, the problem remains challenging due to factors such as the complexity of motion, occlusions and image ambiguities.

While many methods have been proposed for tracking pose over time, Bayesian filtering (Doucet et al., 2000) is popular within the vision community

in large part due to its intuitive probabilistic formulation. Bayesian filtering expresses the posterior distribution by a simple recursion to which existing methods for approximate inference can be applied[†]. Sequential Monte Carlo methods, more commonly known as Particle Filtering (see Sec. 2.3.3), are particularly efficient in approximating the complex, multi-modal posterior distributions that arise due to ambiguities in pose. Particle filtering uses a set of weighted samples to represent the posterior distribution. The number of samples required to accurately represent the posterior grows exponentially with the dimensionality of the hidden variable to be estimated. For human pose, where the dimensionality is usually 30-100 dimensions, the need for a prior to restrict the volume for which the posterior is non-zero becomes more apparent. Therefore, within the past few years prior models of motion have become an integral part of Bayesian filtering architectures. In early years, simple priors were built by hand under the assumption of smoothness. However, the availability of motion capture data has allowed much richer models to be learned directly from examples. The key challenge is now to learn flexible, compact prior models that are able to generalize to previously unobserved subjects and motions.

CRBMs have a number of appealing properties with respect to other models explored in this context to date. Both learning and inference are extremely efficient with complexity linear[‡] in the number of training and testing samples, respectively. This allows them to learn from massive corpora of data. CRBMs can capture the complex nonlinearities inherent in human motion and they can be learned effectively from data of multiple subjects and/or motions. Furthermore, they can generalize to previously unobserved identities and actions. This results in flexible models that can be guided by the history of predictions rather than explicit activity knowledge.

6.2 Related work

Within the scope of this thesis, it is impossible to provide a complete overview of the literature on human motion and pose estimation. We instead

[†]The simplifying assumptions that would permit exact inference are generally abandoned in favour of approximate inference in a more realistic model.

[‡]To model larger, more complex data we need more hidden units and so we only loosely use the term “linear”.

focus on generative models of motion that have either been used or proposed as dynamical priors for tracking. We refer the reader to Forsyth et al. (2006) for a more thorough review of the field.

The most commonly employed prior is a simple, linear, low-order Markov model. While its parameters could be learned from data using the process described in Sec. 2.1.1, the models are typically configured by hand, under the assumption of smoothness. Typically first or second-order models are used. The most common first-order model is based on the assumption that the pose at the next frame is the same as the pose at the current frame, up to additive noise (the zero-velocity assumption). Within this model, motion is a simple random walk. The second-order model typically assumes that velocity is constant from one frame to the next. However, both of these priors are poor approximations to the true dynamics of human motion.

The problem with linear models in general (including Linear Dynamical Systems) is that human motion is inherently nonlinear, grossly violating the smoothness assumption at ground contacts. Switching Linear Dynamical Systems (SLDS, see Sec. 2.3.4) can represent motion as a collection of linear models (Pavolvic et al., 1999). However, exact inference and learning in SLDS models is intractable and approximations must be employed. A further issue with switching models is that it is difficult to ensure consistency in the latent variables when switching from one linear system to another. Modeling multiple activities with a distributed representation rather than a mixture permits us to model transitions much more naturally.

Many related methods are based on the assumption that, for a given activity, the data lies on a low-dimensional manifold within the high dimensional pose space (Lee and Elgammal, 2007). These methods learn low-dimensional embeddings or projections of the data and then learn dynamical models on the low-dimensional manifold. PCA-based methods (e.g. Sidenbladh et al., 2000) will fail in modeling any motion that is non-cyclic. Consequently, models based on nonlinear manifold learning have been proposed (Lu et al., 2008; Sminchisescu and Jepson, 2004), as well as mixtures of linear low-dimensional embeddings such as factor analyzers (Li et al., 2007). Typically after learning the manifold, such methods use simple (i.e. linear) dynamical models in the low-dimensional latent space.

Another promising family of models based on manifold learning is Gaus-

sian Process (GP)-based models. This includes Gaussian Process Latent Variable Models (GPLVM) (Urtasun et al., 2005) and Gaussian Process Dynamical Models (GPDM) (Urtasun et al., 2006). GP-based prior models have already proven effective for tracking (Urtasun et al., 2005, 2006) and were shown to generalize gracefully from small amounts of training data (Urtasun et al., 2005). The disadvantage of these models is their complexity which is quadratic in the data for inference and cubic in the data for learning. Sparse approximations to GPs have been proposed (Lawrence et al., 2003; Quiñonero Candela and Rasmussen, 2005) but the effectiveness of such approaches is still unclear. A major limitation to the GPLVM variants and manifold-based methods in general is their inability to capture multiple types of motion in a single model. Combining different classes of motion quickly destroys the simple manifold structure learned by the models. More recent extensions of GP-based models such as Topologically-constrained Latent Variable Models (Urtasun et al., 2008), Multifactor GPs (Wang et al., 2007), and Hierarchical Gaussian Process Latent Variable Models (H-GPLVM) (Lawrence and Moore, 2007) may offer a solution. However, to date, they have not been proven effective in tracking.

One issue when using low-dimensional manifolds to capture the dynamics of high-dimensional data is consistency. The mapping to the low-dimensional representation should be chosen in such a way that the dynamics are easy to model. This is what happens when fitting a linear dynamical system. Some recent non-linear models also simultaneously learn the mapping and the dynamics (Urtasun et al., 2006; Li et al., 2007). Yet these approaches still suffer from a more serious limitation of using low-dimensional representations: real data can occasionally depart radically from the manifold. A regular walk, for example, only has a few degrees of freedom, but if the walker occasionally scratches his nose or kicks a pebble, an *explicit* low-dimensional representation is hopelessly inadequate. To cope with these radical departures, it is much better to use *implicit* dimensionality reduction. The latent representation remains high-dimensional, but the model learns to construct "energy ravines" in the latent space. Along the floor of the ravine there are a few directions that correspond to the degrees of freedom in the manifold. The remaining directions correspond to the steep sides of the ravine. Moving in these directions makes the representation far less probable, but not impossible (Hinton, 2007b). By learning an energy function that penalizes surplus degrees

of freedom it is possible to have manifolds whose dimensionality is different in different places. It is also possible to have very low dimensional manifolds for very regular motions embedded within somewhat higher dimensional manifolds for less regular versions of the same motion. Also, one set of latent variables can model several different manifolds corresponding to different regular activities. CRBMs capture low-dimensional structure implicitly by learning energy ravines.

6.3 Bayesian filtering with the CRBM

Notation. *In Chapters 4 and 5 we used $\boldsymbol{v}_t$ to denote the observation at time t. To remain consistent with the tracking literature, in this chapter we use $\boldsymbol{x}_t$ to denote the pose (or "state") at time t. Note that in Chapters 4 and 5 we treated pose as observed. In online tracking, however, pose is inferred from image observations $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_t$.*

In Bayesian filtering, we seek to infer the posterior distribution[†] $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$, where $\boldsymbol{x}_t$ is the pose of the body at time t, from a sequence of image observations or features, $\boldsymbol{y}_{1:t} = \boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_t$. We typically assume a sensor Markov assumption, $p(\boldsymbol{y}_t|\boldsymbol{x}_{1:t}, \boldsymbol{y}_{1:t-1}) = p(\boldsymbol{y}_t|\boldsymbol{x}_t)$, which allows us to write the posterior as

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) \propto p(\boldsymbol{y}_t|\boldsymbol{x}_t)p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}), \tag{6.1}$$

where the likelihood, $p(\boldsymbol{y}_t|\boldsymbol{x}_t)$, connects pose to image features and may take many forms, depending on the implementation. Details are discussed in Sec. 6.3.3. The other term, $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1})$, is the predictive distribution. It seeks to predict the pose at time t given image observations up to but not including time t. If we assume first-order Markovian dynamics, $p(\boldsymbol{x}_t|\boldsymbol{x}_{1:t-1}, \boldsymbol{y}_{1:t-1}) = p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, we can write the predictive distribution in the following recursive form:

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}) = \int_{\boldsymbol{x}_{t-1}} p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}) \; d\boldsymbol{x}_{t-1}. \tag{6.2}$$

The term $p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1})$ is simply the posterior from the previous time frame (which is being estimated recursively). We will focus our discussion on $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$, which is called the *transition density* and represents the prior over motion. Note that it has the same form as the density defined by a first-order CRBM (Eq. 4.16). If we use a CRBM to represent the transition density, we need to marginalize over the binary latent variables, $\boldsymbol{h}_t$, such that the

[†] $p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t})$ is often called the filtering distribution. Since we only consider the case of online tracking, we will refer to it simply as the posterior.

predictive distribution becomes

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}) = \int_{\boldsymbol{x}_{t-1}} \sum_{\boldsymbol{h}_t} p(\boldsymbol{x}_t, \boldsymbol{h}_t|\boldsymbol{x}_{t-1}) p(\boldsymbol{x}_{t-1}|\boldsymbol{y}_{1:t-1}) \; d\boldsymbol{x}_{t-1}. \tag{6.3}$$

The binary latent variables either need to be summed out implicitly by sampling or explicitly by computing the free energy (see Sec. 4.2.2).

Eq. 6.2 can only be computed in closed form if both the transition density and the likelihood are Gaussian (this leads to the Kalman filter). However, both the CRBM dynamics and the likelihood term we use in practice are highly non-Gaussian. This permits a multi-modal posterior distribution but forces us to seek an approximation to this distribution. We employ a commonly used particle filtering algorithm called Sequential Importance Resampling (SIR) where the posterior is represented by a set of weighted samples or "particles", $\{\boldsymbol{x}_t^{(p)}, \boldsymbol{w}_t^{(p)}\}_{p=1}^{P}$. At every time frame we sample a set of particles from the predictive distribution and weight them according to the likelihood:

$$\begin{aligned} \boldsymbol{x}_t^{(p)} &\sim p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t-1}), \\ w_t^{(p)} &\propto p(\boldsymbol{y}_t|\boldsymbol{x}_t^{(p)}). \end{aligned} \tag{6.4}$$

This provides a discrete approximation to the posterior distribution:

$$p(\boldsymbol{x}_t|\boldsymbol{y}_{1:t}) \approx \sum_{p=1}^{N} w_t^{(p)} \delta(\boldsymbol{x}_t - \boldsymbol{x}_t^{(p)}) \tag{6.5}$$

where δ is the Dirac delta function. Periodic resampling is used to avoid the situation where all but one of the important weights are zero. Figure 6.1 visualizes our proposed method.

A final point to be made is that although we can only calculate $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ up to a multiplicative constant, this does not stop us from applying the CRBM as a dynamical prior. Since we only need the prior to draw samples from the predictive distribution, the unnormalized and on-line nature of the CRBM makes it a good fit for Bayesian filtering.

6.3.1 Higher-order dynamics

Considering higher-order CRBMs (where $N > 1$) are required for high-quality motion synthesis, we would expect them to make better priors for tracking. A model must see two or three previous frames to implicitly calculate velocities

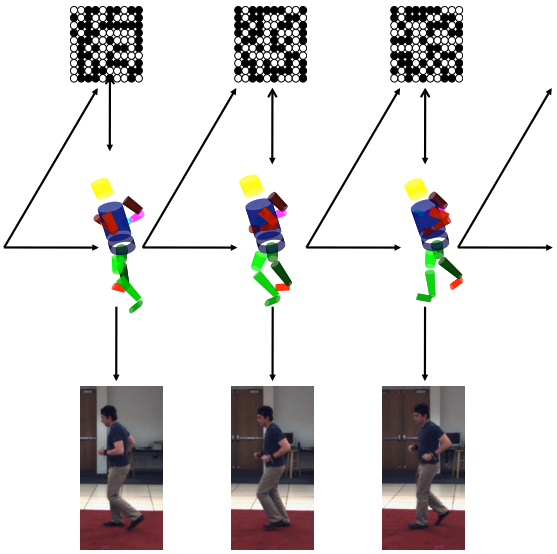


Figure 6.1. **Bayesian filtering with a CRBM motion prior.** Our motion prior is a joint distribution over binary latent features and observed pose conditioned on the past history of poses. The likelihood term describes a distribution over image features given pose. First order dynamics are shown. The grid structure of the hidden units is purely for display purposes.

or accelerations, and this information should be useful to a dynamical prior. In our discussion above, we have assumed first-order Markovian dynamics but can relax this assumption to the N-th order case by defining an augmented state (Isard and Blake, 1998) $\hat{\boldsymbol{x}}_t = \boldsymbol{x}_{(t-N):t}$. We also define a new transition density, $p(\hat{\boldsymbol{x}}_t|\hat{\boldsymbol{x}}_{t-1})$, that temporally shifts the elements of the augmented state by one step, and predicts (or samples) the next state via the N-th order CRBM conditioned on the past augmented state. In practice we use a freely available SIR implementation (Balan et al., 2007) which we have modified to support a sample-based representation of the history over the past N frames.

6.3.2 Modeling the body

We use a standard 3D kinematic chain representation of the body where each segment is represented by a truncated cone with an elliptical cross-section (Figure 6.2a). The body model consists of 15 segments: 2 segments

corresponding to the torso (pelvic region and upper body), 1 segment for the head, and the remaining segments are assigned to limbs: lower and upper arms and legs, feet and hands. We assume that the lengths, widths and cross-sectional scaling for all segments is fixed and known *a priori*. The pose has 40 degrees of freedom (dof): the global position and orientation of the body in world co-ordinates (6 dof), orientation of the hips, shoulders, head and abdomen (3 dof each, 18 dof total), orientation of the clavicle, elbows and knees (2 dof each, 12 dof total), and finally the orientation of wrists and ankles (1 dof each, 4 dof total).

To achieve invariance to ground-plane translations and rotations about the gravitational vertical, we convert the pose to our standard invariant representation (see Chapter 3) before learning CRBMs. During tracking we also use our invariant representation to store and propagate the particles. The likelihood term only requires the 3D locations of the cylinders which are easy to compute from either the original representation or our invariant representation.

The body parameterization which we have chosen, though flexible in terms of kinematics, is a gross oversimplification in terms of true geometry. We expect that richer body models, such as those proposed in (Balan et al., 2007) will further improve the performance of our method.

6.3.3 Likelihood

The focus of our work is to develop good prior models of dynamics, and therefore we apply simple, generic likelihoods that are commonly used for tracking. We employ two types of likelihoods based on edges or silhouettes (Figure 6.2b-c). Given pose, we first compute the locations of either edges or silhouettes by computing cylinders for each segment and projecting these into the camera view. A point-based edge-map or silhouette computed from sampling around or within the projected cylinders is then matched with edge-maps or silhouettes extracted from the images. Scores are summed over camera views, weighting edge and silhouette scores equally. Details can be found in (Deutscher and Reid, 2005). We expect that better results may be obtained by using richer likelihoods that are robust to lighting variations, occlusions and sensor noise (e.g. optical flow or adaptive appearance regions (Urtasun et al., 2005, 2006)). However, the use of standard likelihoods allow

(a) (b) (c)

Figure 6.2. (a) The cylindrical body model. (b) Edge-based likelihood. Uniformly-spaced points along the exterior of the projected cylinders are compared with an edge-map extracted from a camera image. (c) Silhouette-based likelihood. Uniformly-spaced points inside the projected cylinders are compared with a silhouette computed by background subtraction on a camera image. Images reproduced from (Sigal et al., 2009).

us to fairly evaluate our approach with respect to the prior art.

6.4 Experiments

We first carried out a series of synthetic experiments to measure the effective predictive power of the CRBM in the context of tracking (Sec. 6.4.1). Then we tested its effectiveness as a dynamical prior in real multi-view and monocular 3D tracking (Sec. 6.4.2 and 6.4.3). We evaluated the performance of our proposed method on a variety of sequences from the HumanEva-I/II datasets (Sigal and Black, 2006).

Datasets. HumanEva is a publicly available dataset consisting of a set of multi-view image sequences with synchronized motion capture data to allow quantitative evaluation of performance. The HumanEva dataset consists of six different motions, performed by four different subjects (denoted S1-S4). We used sequences of walking, jogging, boxing and combo (walk transitioning to a jog) in our experiments. To supplement our dataset, we used an earlier

synchronized walking sequence from a different subject, denoted S5, made available by the same group (Balan et al., 2005).

Evaluation. To quantitatively evaluate the performance of our method, we adopted the measure proposed in (Sigal and Black, 2006), which computes an average Euclidian distance between 15 virtual markers on the body. These markers correspond to joint centres or end points of the segments described in Sec. 6.3.2. The use of HumanEva and this standard error measure allowed us to easily compare the performance of our method to other published results.

Baseline. The baseline we considered was the standard particle filter but with smooth zero-order dynamics (i.e. $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t$ up to additive noise). For fairness, we always utilized the same number of samples to approximate the posterior and the same likelihoods between the baseline and the proposed approach.

Details of learning. Except where noted, we used the same learning procedure as described in Sec. 4.5. Models were trained using CD-10 (See Sec. 2.4.3) for 5000 complete passes through the data.

Where higher order models are used, we first "grow" the trajectory using a low order CRBM. For example, we initialize with frames 1 and 2 and infer the pose at frame 3 and 4 using a first-order CRBM. Now that we have a four frame history (for each particle) we can calculate the velocities needed by a third-order CRBM to infer pose at frames 5, 6 and 7. Then we can apply a sixth-order CRBM to track the remainder of the sequence.

Initialization. To initialize the tracker we used the first two frames from the provided motion capture data (converted into our representation). The use of two frames, as opposed to one was required to calculate the three velocities that are part of our invariant representation of pose (see Sec. 3.2).

6.4.1 Quality of predictions

While the CRBM is a generative model and is not trained to minimize predictive error, we can still apply it to the N-step prediction task as a means of quantitatively comparing model structure. To assess the performance of CRBMs conditioned on different-length histories, we have used the dataset from (Hsu et al., 2005) described in Sec. 5.6.3. The dataset contains labeled sequences of seven types of walking: *crouch*, *jog*, *limp*, *normal*, *side-right*, *sway*, and *waddle* each at three different speeds: *slow*, *medium*, and *fast*.

We compared 100 hidden-unit CRBMs of first, third and sixth order as well as two simple baselines: predicting the last frame of known data, and the constant-velocity method. For each method, we trained[†] 21 different models

[†]The baselines did not need to be trained but we used the same testing methodology.

on all style/speed pairs except one, which we withheld for testing. Then, for each model, we attempted to predict every subsequence of length M in the test set, given the past N frames (N being the order of CRBM; $N = 1, 2$ for the two respective baselines). For each CRBM architecture, we compared two methods of prediction: Gibbs sampling and following the gradient of the free energy with respect to the visible units using conjugate-gradient optimization (see Sec. 4.2.3). We report results averaged over the 21 validation sequences. All CRBM models were trained for 500 epochs (passes through the training data). The results are presented in Figure 6.3. Increasing the order of the model has a positive effect on predictive performance; though the gain from first to third-order is much more significant. We believe that this is due to the weights of the model implicitly calculating both velocity and accelerational information once a history of $N = 3$ is available. The simple baselines are accurate at predicting a few frames into the future ($M < 4$) but quickly deteriorate. We think the reason for the slight improvement of the "last frame" baseline after frame 18 is due to the cyclic nature of the data.

Predictions in the presence of noise. In a real tracking environment, we expect that the inputs provided to the motion prior will differ considerably from the inputs observed during training. To this effect, we wish to build a motion prior that is robust to the noise present during tracking. One way to achieve robustness is to regularize the model by adding Gaussian noise to the history during training. This would have an identical effect to adding L2 weight decay to the directed weights if we used linear, rather than logistic units. Figure 6.4 shows a typical training phase of a third-order CRBM. We trained on subject S1 walking (2207 frames). After every 100 epochs we tested the model's ability to predict each frame in the validation sequence given the recent history. The model was able to achieve a low prediction error rate but clearly suffered when noise was applied to the input. We applied noise independently on each dimension with the standard deviation set to the maximum inter-frame change in the training set, neglecting the 1% outliers. However, when the same noise was added to the input during training, the model became much more robust. In the presence of no noise at test time, regularization obscures some of the structure that can be learned by the model. This is, however, offset by the gain in the more realistic, noisy scenario.

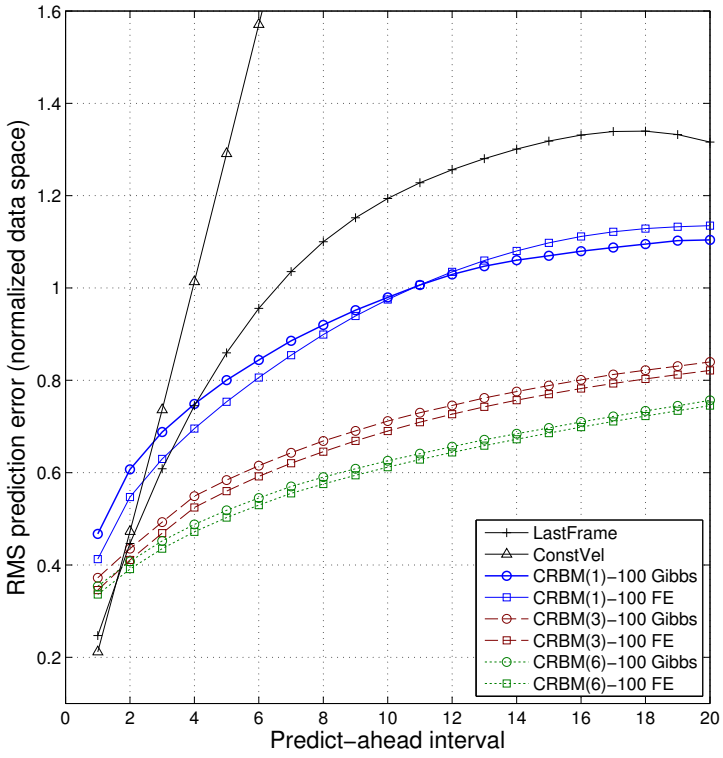


Figure 6.3. *N*-step prediction task on the MIT style dataset.

6.4.2 Multi-view tracking

Generalization across subjects. We repeated the experiments first proposed in (Xu and Li, 2007) to demonstrate the insensitivity of the proposed dynamical model to the identity of the subject. Xu and Li's approach was based on exploiting natural symmetry by learning a body correlation model to infer the pose of the right side of the body, given the left. This allowed them to reduce the state space used in particle filtering by the number of dof assigned to the right side of the body. They maintained a posterior over the reduced state-space and used their correlation model to estimate the missing dof. To compare with the previously published results, we maintained the same experimental setup as in (Xu and Li, 2007) but instead of reducing the state space, we applied a first-order CRBM prior to the full state space. In all cases,

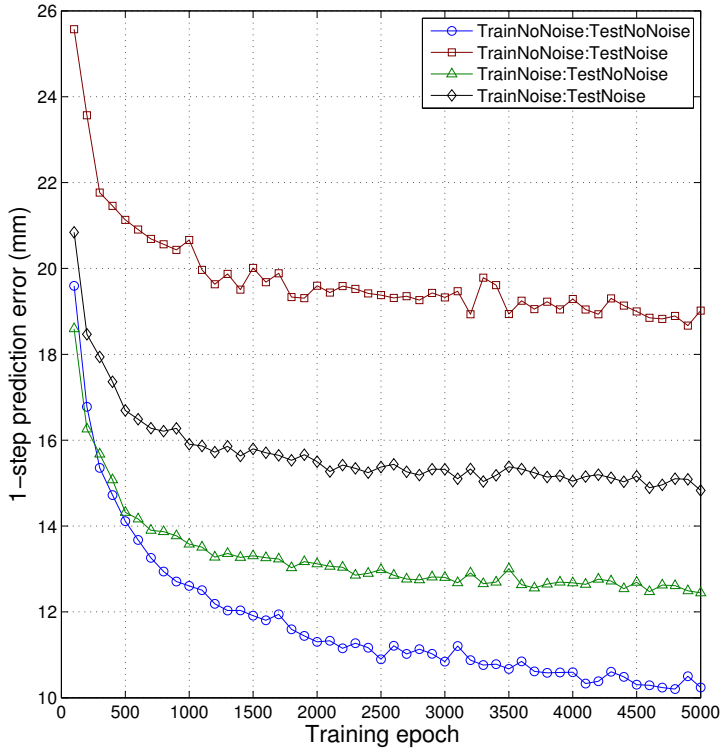


Figure 6.4. Effects of Gaussian noise added to inputs during training and testing on the one-step prediction task.

we tracked subject S5 (first 150 frames of the validation sequence), and we repeated the experiment, training with three different datasets: S5 walking, S1 walking, and the combined walking data of subjects S1,S2 and S3. We used 4 camera views and 1000 particles for all three sequences. To the best of our knowledge, we utilized the same edge and silhouette-based likelihood, same basic inference architecture, same number of samples, and the same test and training sets. Results (averaged over 10 runs) are shown in Figure 6.5 and Table 6.1.

In each case, our method outperforms standard particle filtering and Xu and Li's approach. Surprisingly, we perform slightly better on subject S5 using a prior model trained on subject S1. This may be due to the smaller S5 training dataset (921 frames compared to 2232 for S1).

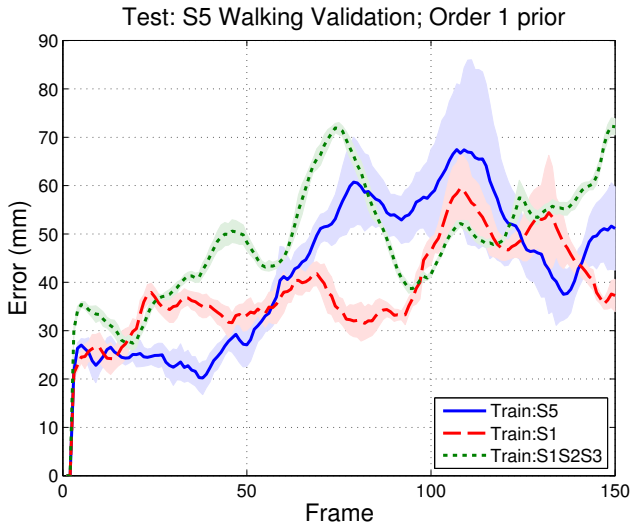


Figure 6.5. Tracking of subject S5 using a first-order prior model learned from S5, S1, and S1+S2+S3 training data. Results are averaged over 10 runs (per-frame standard deviation is shaded).

HumanEva-I walking. Next, we applied our model to the walking sequences in the HumanEva-I dataset. As in (Xu and Li, 2007), we tracked each of subjects S1,S2 and S3 (walking validation) using a first-order dynamics model trained on the combined walking data of all three subjects. Additionally, we repeated the experiment using a subject-specific prior. Our method is identical to that of the previous experiment, except that we used three views instead of four (which incidentally, makes the task more difficult). Figure 6.6 shows the results of tracking each subject. Our results for subjects S1 and S3 show insensitivity to subject identity. Subject S2 shows more sensitivity, but in all cases the performance of our method is considerably better than standard particle filtering or Xu and Li's approach (Table 6.2).

HumanEva-I boxing. Li et al. (2007) proposed a nonlinear dynamical model based on a generalized Switching Linear Dynamical System. They integrated their model into a tracking architecture significantly different than particle filtering and achieved favourable results compared to a standard SLDS and a Dynamic Global Coordination Model (Lin et al., 2006). They evaluated

Training set	Baseline	Xu and Li (2007)	CRBM
S5	91.37±6.29	48.98	41.97±3.57
S1		51.66	38.45±0.80
S1S2S3		55.30	48.03±0.29

Table 6.1. Tracking of subject S5 using a first-order prior model learned from S5, S1, and S1+S2+S3 training data. Note that the baseline does not involve learning a model and therefore does not depend on training set.

Training:Test	Baseline	Xu and Li (2007)	CRBM
S1S2S3:S1	129.18±19.47	140.35	55.43±0.79
S1:S1		-	48.75±3.72
S1S2S3:S2	162.75±15.36	149.37	99.13±22.98
S2:S2		-	47.43±2.86
S1S2S3:S3	180.11±24.02	156.30	70.89±2.10
S3:S3		-	49.81±2.19

Table 6.2. Tracking of subjects S1,S2,S3 using various prior models.

their model on HumanEva-I subject S1 boxing. We trained first, third and sixth-order CRBM models using the S1 training boxing data and test on the subject S1 boxing validation sequence. For this experiment we only used 200 particles. Tracking error is summarized in Table 6.3. We report the average over 10 runs using each order model. Again our method shows significant improvement over standard particle filtering and Li et. al's approach.

Model	Order 1	Order 3	Order 6
Li et al. (2007)	187.50	-	-
Baseline PF	116.95±5.54	-	-
PF + CRBM	75.35±9.71	82.40±8.26	82.91±5.15

Table 6.3. Tracking of subject S1 boxing.

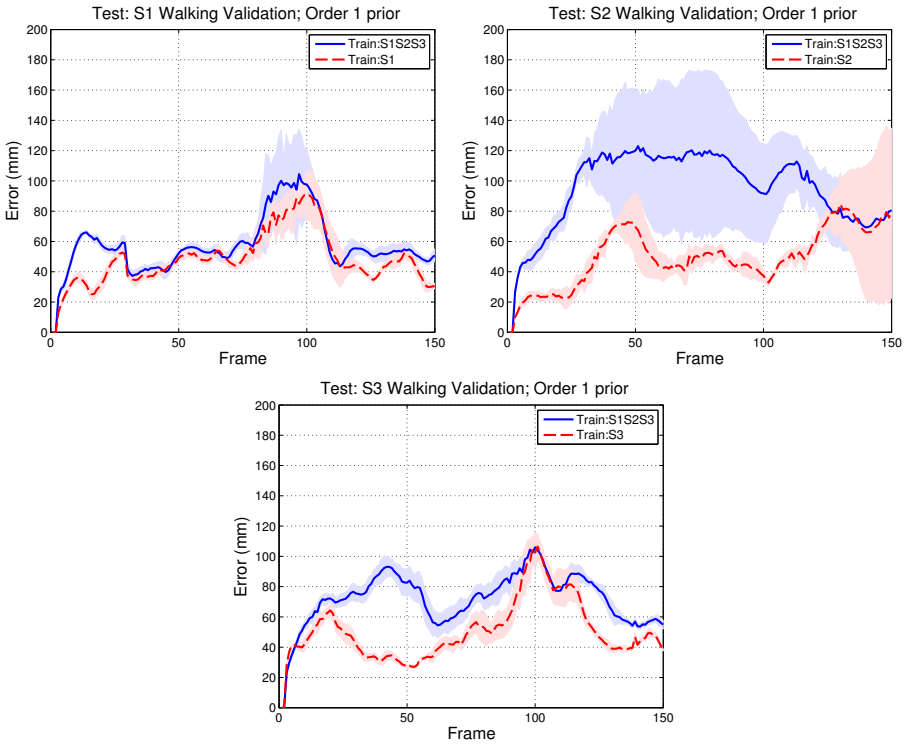


Figure 6.6. Tracking the walking validation sequences in the HumanEvaI dataset using subject-specific and non-specific priors. Results are averaged over 10 runs (per-frame standard deviation is shaded).

HumanEva-I combo. We trained a first-order CRBM model on all subject S3 walking and jogging training sequences. Then we applied particle filtering with 1000 particles to the first 900 frames of the S3 "combo" sequence which consists of walking transitioning to jogging. Our method tracked reliably through the entire sequence and was able to capture both gaits. A single run is visualized in Figure 6.7 (baseline) and 6.8 (CRBM dynamical prior). Note that our method had some difficulty in capturing the arms during the transition (frames 376-556). We believe this is due to the absence of transition examples in the training set. The prior has a tendency to enforce consistency and so it treats the transition as walking until sampling noise allows it to mix enough to find the jogging mode. Tracking with a sixth-order dynamical prior performed reliably in the "walk" and "jog" sections of the combo sequence,

but the higher-order model makes a stronger bid for consistency when the walking transitions to jogging, and therefore is slower to find the jogging mode. The CRBM is able to synthesize transitions, and so we expect that adding transitions to the training set will improve our tracker.

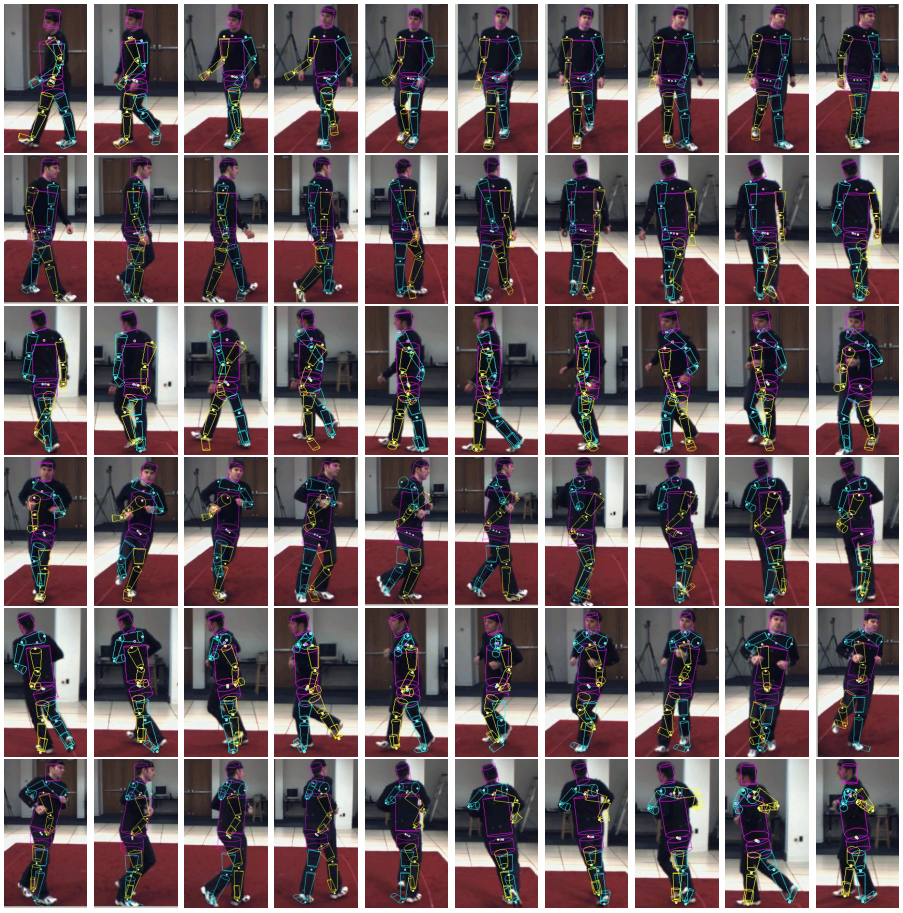


Figure 6.7. Multi-view tracking of the first 900 frames of subject S3 "combo" validation using the baseline particle filter (1000 particles, smooth zero-order dynamics). We used three cameras but only camera 1 is shown. We display every 15th frame.

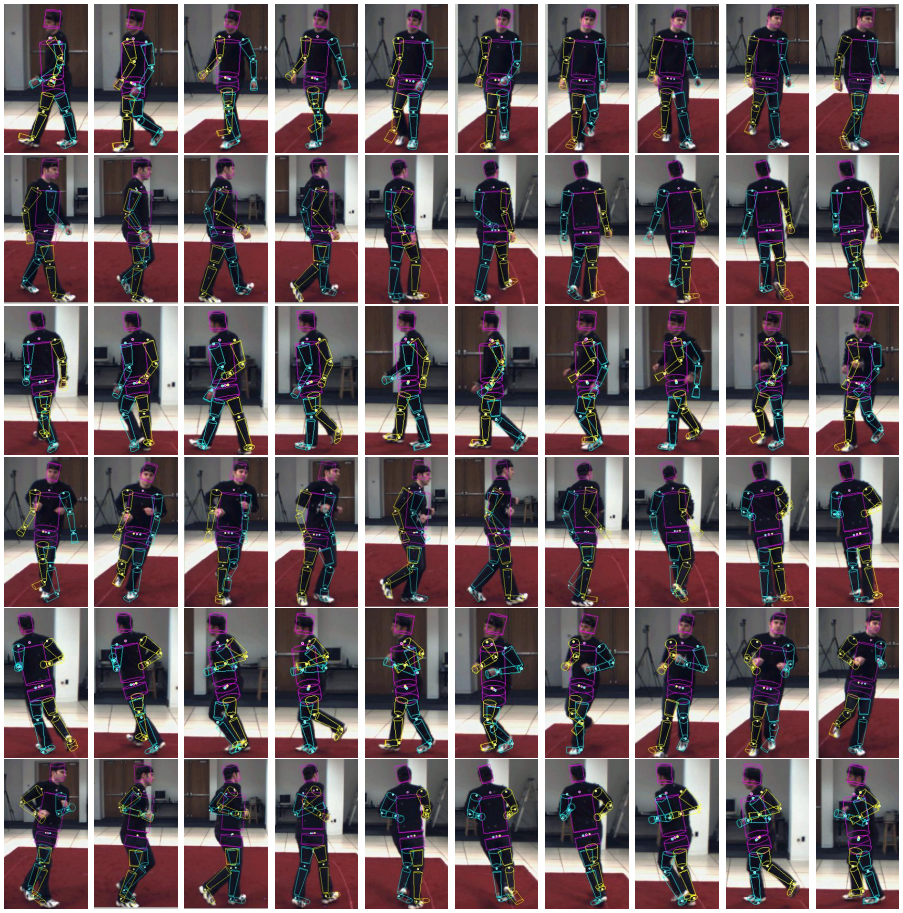


Figure 6.8. Multi-view tracking of the first 900 frames of subject S3 "combo" validation using particle filtering with a first-order CRBM dynamical prior. We used three cameras but only camera 1 is shown. We display every 15th frame.

6.4.3 Monocular tracking

Tracking with a single camera is a much more challenging task and is beyond the reach of most methods. Balan et al. (2005) report that monocular tracking using an Annealed Particle Filter (5 layers, 200 particles per layer) with edge and silhouette-based likelihoods fails, on average, after 40 frames of tracking the subject S5 validation sequence. They report an error of 263$\pm$60mm tracking the first 150 frames of the sequence. We applied standard particle filtering with a CRBM motion prior trained on S5 training data to the same validation sequence. We used an identical likelihood, and used the same total number of particles (1000). Averaging errors over all four cameras, and five runs per camera, using a first-order CRBM gives an average error of 133.93$\pm$55.62mm. If we apply a sixth-order CRBM, our results improve to 112.25$\pm$79.52mm. For each camera, at least one run successfully tracks the entire sequence. All camera 1 runs are successfully tracked for the entire sequence.

We also applied our method to track subject S1 (walking validation sequence). Using a first order CRBM, and a bi-directional silhouette likelihood term, we obtained an error of 90.98$\pm$32.70 averaged over 5 runs per camera (Figure 6.10). When tracking was reasonably successful, using a higher-order model helped considerably. For example, using camera 1 resulted in an average error of 47.29$\pm$4.95mm with a sixth-order model (compared to 70.50$\pm$24.19 for the first-order model). A single run is visualized in Figure 6.11.

6.5 Discussion

In this chapter, we demonstrated that a CRBM works effectively as a prior in Bayesian filtering, allowing 3D tracking of people from multi-view and monocular observations. In contrast to the many approaches which learn a low-dimensional manifold, the CRBM uses a high-dimensional, nonlinear representation which captures low-dimensional structure by learning energy ravines. This allows us to learn models from many different types of motion and subjects using the same set of latent variables. Our method performs favourably when compared to existing approaches on standard datasets from the literature (HumanEva).

We have yet to incorporate deep architectures or models with multiplicative interactions into our tracking framework. Both should integrate naturally

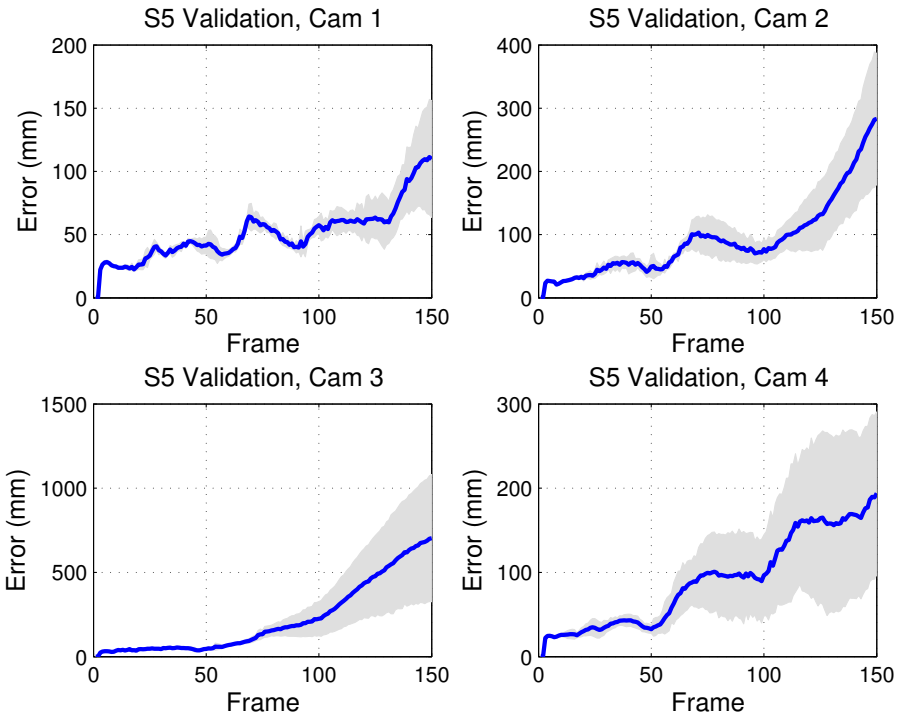


Figure 6.9. Monocular tracking of subject S5. Results are averaged over 5 runs per camera (per-frame standard deviation is shaded).

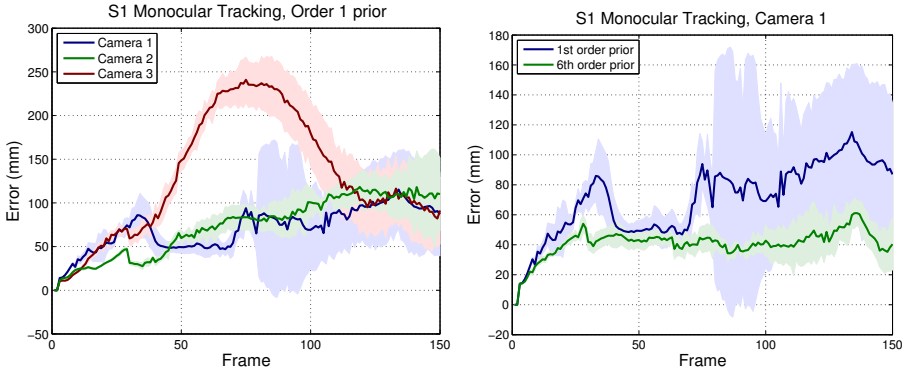


Figure 6.10. Left: Monocular tracking of subject S1, all cameras. Right: Using a sixth order model improved the results for cameras 1 and 2 (camera 1 shown). Results are averaged over 5 runs per camera (per-frame standard deviation is shaded).

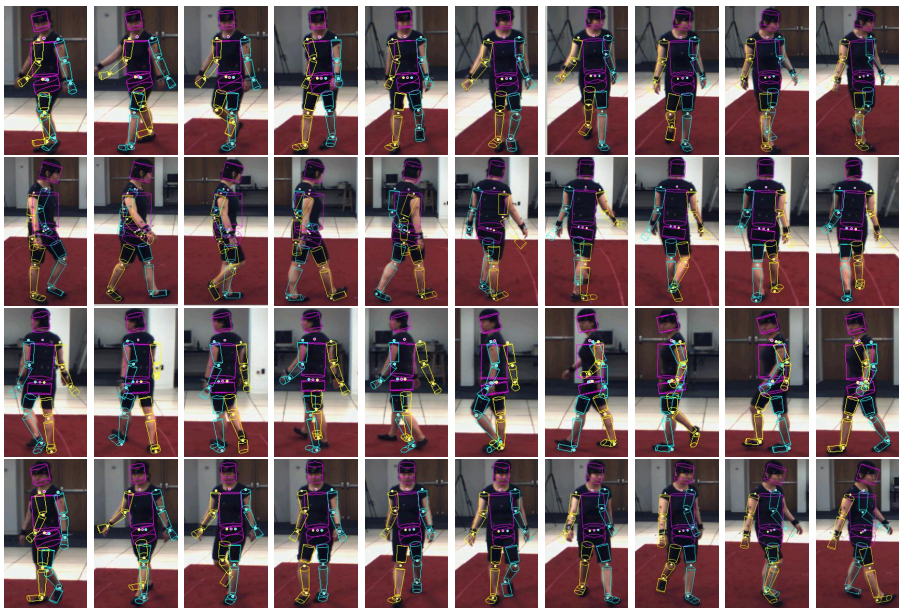


Figure 6.11. Monocular tracking of the subject S1 validation sequence using camera 1. We use a sixth-order CRBM dynamical prior and 1000 particles. Every 15th frame is shown. The last frame is also shown. Note the error in initialization from which we recover.

with Bayesian filtering and we intend to pursue these methods as future work. When applying CRBMs to synthesize data, the identity of the subject and activity are always assumed known *a priori*. This is in contrast to the tracking problem, where these variables typically are unobserved. If we model these variables while learning the prior (instead of conditioning) we should be able to include them in the state-space and maintain an approximate posterior over both pose and subject or activity.

In our experiments, we have used simple, generic likelihoods based on edges and silhouettes. It may be possible to learn an RBM or CRBM over pose and image features to either supplement or replace the current likelihoods. Another alternative would be to abandon the Bayesian filtering framework and track with a conditional deep belief network whose upper layers receive input from two sensory pathways defined by motion and image-based feature extractors.

7 Evaluating Products of Hidden Markov Models
(It Takes N>1 to Tango)

Products of Hidden Markov Models (PoHMMs) are an interesting class of models which have received little attention since their introduction. This may be in part due to their more computationally expensive gradient-based learning algorithm, and the intractability of computing the log likelihood of sequences under the model due to the partition function. In this chapter, we demonstrate how the partition function can be estimated reliably via Annealed Importance Sampling. We perform experiments using contrastive divergence learning on rainfall data and data captured from pairs of people dancing. Our results suggest that advances in learning and evaluation for undirected graphical models and recent increases in available computing power make PoHMMs worth considering for complex time series modeling tasks.

7.1 Introduction

The advantage of Hidden Markov Models (HMMs) over fully observed models is that they introduce a multinomial hidden state that summarizes the history of observations. However, in Chapter 1, we posited that many high-dimensional data sets with intricate componential structure cannot be modeled efficiently by HMMs because the representational power of this state is so limited. The Input-Output Hidden Markov Model (Sec. 2.2.1) implicitly adds capacity to the HMM by conditioning its transition and emission distributions on another input sequence, such that the resulting distributions are non-stationary. A more natural way to achieve greater representational capacity

is to use an exponentially more powerful distributed hidden state. Factorial Hidden Markov Models (FHMMs) (Sec. 2.2.2) take this approach but suffer from "explaining away". Inference in these models is difficult.

POHMMs (Brown and Hinton, 2001) are another member of the HMM-based family that have distributed hidden state. At a high level they appear similar to FHMMs, but have an undirected, rather than causal, relationship between hidden state and observations. Observing a sequence decouples each of the HMMs. Exact inference reduces to independently running the forward-backward algorithm in each HMM, and learning can be done efficiently by minimizing contrastive divergence (CD). Surprisingly, since their introduction PoHMMs have received little attention apart from some demonstrated success on "toy" language tasks. They have yet to be applied to high-dimensional, real-world data. Part of the reluctance in adopting PoHMMs may have been due to the computationally demanding nature of gradient-based CD learning compared to the efficiency of the Baum-Welch algorithm for standard HMMs. However, since the introduction of PoHMMs, available computing power has greatly increased, and further insights have been made with respect to training models by CD (Carreira-Perpinan and Hinton, 2005; Tieleman, 2008).

Another concern with undirected models in general has been the intractability of calculating the log likelihood of an observed sequence due to the existence of a normalizing factor (the "partition function") in the likelihood. In this chapter, we demonstrate how the likelihood of sequences under a PoHMM can be estimated to a high degree of accuracy using a method of partition function estimation based on Annealed Importance Sampling (Neal, 2001). Through experiments in modeling both rainfall occurrence and human motion, we show that PoHMMs are often an attractive choice over other members of the HMM-based family.

7.2 Products of Hidden Markov Models

One way of modeling a complicated, high-dimensional data distribution is to combine a number of relatively simple models, or "experts". Mixtures are an example of this methodology. They can approximate complicated smooth distributions arbitrarily accurately but are very inefficient in high dimensions. The mixture cannot produce a resultant distribution sharper

than any of the individual experts. An alternative approach is to multiply the individual probability distributions together and then renormalize. This is called a Product of Experts (PoE) (Hinton, 2002). PoEs allow us to produce much sharper distributions than the individual models. They are defined by the following likelihood over observations:

$$p(\boldsymbol{y}|\Theta) = \frac{\prod_{m=1}^{M} p^{(m)}(\boldsymbol{y}|\theta^{(m)})}{\sum_{\boldsymbol{y}'} \prod_{m=1}^{M} p^{(m)}(\boldsymbol{y}'|\theta^{(m)})} \tag{7.1}$$

where m indexes the individual experts, $\Theta = \{\theta^{(m)}\}$ represents parameters, and the probability distribution of the individual experts may be of any type. Experts may be proper distributions or unnormalized potential functions. The Restricted Boltzmann Machine and its extensions (see Chapters 4 and 5) are Products of Experts where each hidden unit is an expert. This can be seen by considering Eq. 2.34, where each hidden unit makes an additive contribution to the energy (i.e. multiplicative in probability space). The partition function that appears in the expression for the likelihood and consequently the gradients for maximum likelihood learning affects the entire PoE family. However, contrastive divergence learning, which we discussed for RBM models, applies to PoEs in general – provided the individual experts themselves are tractable.

If we take each of the experts to be an HMM, the resulting PoE is called a Product of Hidden Markov Models (PoHMM) (Brown and Hinton, 2001). The graphical model for a PoHMM is shown in Fig. 2.1d. PoHMMs are ideal for modeling data that is caused by multiple underlying influences. Each HMM can model some aspect of the data and constrain it such that the combination of the models will have a very sharp distribution. And if each HMM can remember a different piece of information about the past, the PoHMM should be able to capture more long-range structure than a standard HMM.

Note that in a PoHMM each expert defines a distribution over sequences, $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_T$, rather than static observations, $\boldsymbol{y}$.

7.2.1 Contrastive divergence learning for a PoHMM

Contrastive divergence learning for a PoHMM updates the parameters according to:

$$\Delta\theta^{(m)} \propto \left\langle \frac{\partial}{\partial\theta^{(m)}} \log p(\boldsymbol{y}_{1:T}|\theta^{(m)}) \right\rangle_{\text{data}} \quad - \left\langle \frac{\partial}{\partial\theta^{(m)}} \log p(\boldsymbol{y}_{1:T}|\theta^{(m)}) \right\rangle_{\text{recon}} \tag{7.2}$$

where, as usual, the first expectation is with respect to the data distribution, and the second expectation is with respect to the distribution of samples

obtained after K steps of alternating Gibbs sampling, initialized at the data (i.e. CD-K). The gradient terms in Eq. 7.2 are obtained by the following procedure:

1. Set $\boldsymbol{y}^0_{1:T}$ to the observed sequence. Perform the forward-backward algorithm *independently* in each HMM to compute the sufficient statistics. Calculate each HMM's gradient, $\frac{\partial}{\partial \theta^{(m)}} \log p(\boldsymbol{y}^0_{1:T}|\theta^{(m)})$, on this sequence, using the sufficient statistics. For details on the gradients, see (Brown, 2001). This leads to the first term in Eq. 7.2.

2. **Repeat for** $k = 1, \ldots, K$**:** Sample from the posterior distribution of paths through state space for each of the HMMs given $\boldsymbol{y}^{k-1}_{1:T}$ (where the posterior is obtained by performing the forward-backward algorithm in each HMM). Multiply together the emission distributions defined by the sampled paths in each HMM and renormalize. Reconstruct a visible sequence, $\boldsymbol{y}^k_{1:T}$, by drawing a sample from this distribution.

3. For each HMM, calculate $\frac{\partial}{\partial \theta^{(m)}} \log p(\boldsymbol{y}^K_{1:T}|\theta^{(m)})$ using the sufficient statistics obtained by performing the forward-backward algorithm on the final K-step reconstructed sequence. This gives the second term in Eq. 7.2.

The learning rule has the disadvantage that it is a stochastic gradient rule which is subject to noise, so we cannot employ more sophisticated second-order methods or conjugate gradient optimization. The convergence time will be much slower than for fixed-point methods such as the EM algorithm used to fit tractable models like the standard HMM. However, the exponential gain in representational power should more than offset this shortcoming.

7.3 Estimating the Partition Function

Following Eq. 7.1 the log likelihood that a PoHMM assigns to a sequence,$\boldsymbol{y}_{1:T}$, is given by:

$$\log p(\boldsymbol{y}_{1:T}|\Theta) = \sum_{m=1}^{M} \log p^{(m)}(\boldsymbol{y}_{1:T}|\theta^{(m)}) - \log Z(T, \Theta) \tag{7.3}$$

where Z is the partition function which depends on both the length, T, of the sequence and the model parameters, Θ. A key feature of the PoHMM is the fact that given an observation, the experts decouple and the first term

can be easily computed by evaluating $\log p^{(m)}(\boldsymbol{y}_{1:T}|\theta^{(m)})$ for each expert. The partition function, however, involves an intractable sum (or integral, in the case of continuous emission distributions) over all possible observations of length T. For all but the smallest discrete models, it is not practical to compute.

A good estimate of the partition function would aid us in both model selection and controlling model complexity since it allows us to compute an estimate of the log probability of an observation under the model. It would also permit the comparison of PoHMMs to other generative models, such as standard HMMs. Salakhutdinov and Murray (2008) have recently reported success in applying a particular type of Monte Carlo method, Annealed Importance Sampling (AIS), to RBMs. Given the similarity between members of the products of experts family, we are motivated to extend their work to PoHMMs. We first give a brief review of AIS but refer the reader to Neal (2001) for a more thorough discussion.

7.3.1 Importance sampling

In Sec. 2.3.3 we briefly discussed importance sampling, a Monte Carlo method method that allows us to either generate samples from a complicated distribution of interest, or calculate statistics with respect to that distribution. For the purposes of our discussion, let us call this distribution p_B. Importance sampling is based on the idea of generating independent points from some simpler approximating distribution, p_A, and associating a weight with each point to compensate for the use of the wrong distribution. A byproduct of importance sampling is that it also provides an estimate of the ratio of the normalizing constants (partition functions) of the two distributions. This ratio is a useful quantity because it allows us to compare two different models by the likelihood they assign to some test data.

Let us suppose that $p_A(\boldsymbol{y}_{1:T}|\theta_A) = \frac{p^*_A(\boldsymbol{y}_{1:T}|\theta_A)}{Z_A(\theta_A,T)}$ and $p_B(\boldsymbol{y}_{1:T}|\theta_B) = \frac{p^*_B(\boldsymbol{y}_{1:T}|\theta_B)}{Z_B(\theta_B,T)}$ are models of sequences (e.g. PoHMMs) where p^*_A and p^*_B are computable but Z_A and Z_B are not. Comparing the models based on likelihood leads to:

$$\frac{p_A(\boldsymbol{y}_{1:T}|\theta_A)}{p_B(\boldsymbol{y}_{1:T}|\theta_B)} = \frac{p^*_A(\boldsymbol{y}_{1:T}|\theta_A)}{p^*_B(\boldsymbol{y}_{1:T}|\theta_B)} \frac{Z_B(\theta_B,T)}{Z_A(\theta_A,T)} = \frac{p^*_A(\boldsymbol{y}_{1:T})}{p^*_B(\boldsymbol{y}_{1:T})} \frac{Z_B}{Z_A} \tag{7.4}$$

where we will use the shorthand notation introduced in the rightmost term to simplify the presentation. Eq. 7.4 states that to compare the models based

on likelihood, we must be able to compute the ratio of partition functions, Z_B/Z_A. Note that we can rewrite the ratio as:

$$\frac{Z_B}{Z_A} = \frac{\int p_B^*(\boldsymbol{y}_{1:T})}{Z_A} d\boldsymbol{y}_{1:T} = \int \frac{p_B^*(\boldsymbol{y}_{1:T})}{p_A^*(\boldsymbol{y}_{1:T})} p_A(\boldsymbol{y}_{1:T}) d\boldsymbol{y}_{1:T} = \mathbb{E}_{p_A}\left[\frac{p_B^*(\boldsymbol{y}_{1:T})}{p_A^*(\boldsymbol{y}_{1:T})}\right] \quad (7.5)$$

where the integral is taken over all sequences of length T and assuming that $p_A^*(\boldsymbol{y}_{1:T}) \neq 0$ whenever $p_B^*(\boldsymbol{y}_{1:T}) \neq 0$.

If we can draw independent samples from p_A, importance sampling allows us to estimate Eq. 7.5 by:

$$\frac{Z_A}{Z_B} \approx \frac{1}{P}\sum_{p=1}^{P} \frac{p_B^*(\boldsymbol{y}_{1:T}^{(p)})}{p_A^*(\boldsymbol{y}_{1:T}^{(p)})} \equiv \frac{1}{P}\sum_{p=1}^{P} w^{(p)} \quad (7.6)$$

where $\boldsymbol{y}_{1:T}^{(p)} \sim p_A$ and $w^{(p)}$ is called the importance weight of sample p (see Sec. 2.3.3). However, if p_A and p_B differ considerably then Eq. 7.6 is a poor estimate of the true ratio Z_B/Z_A.

7.3.2 Annealed importance sampling

Annealed importance sampling (Neal, 2001) can achieve better estimates of Z_A/Z_B by defining a series of intermediate distributions, $p_0, p_1, \ldots, p_N$ which slowly anneal from $p_0 = p_A$ to $p_N = p_B$. The intermediate distributions must be defined such that we can easily evaluate the unnormalized probability, $p_n^*(\boldsymbol{y}_{1:T}); \forall \boldsymbol{y}_{1:T}, n = 0, 1, \ldots, N$. We also must define a Markov Chain transition operator, $T_n(\boldsymbol{y}_{1:T}, \boldsymbol{y}'_{1:T})$ that, at each intermediate distribution allows us to sample a new sequence $\boldsymbol{y}'_{1:T}$, given $\boldsymbol{y}_{1:T}$, while leaving $p_n(\boldsymbol{y}_{1:T})$ invariant:

$$\int T_n(\boldsymbol{y}_{1:T}, \boldsymbol{y}'_{1:T}) p_n(\boldsymbol{y}_{1:T}) d\boldsymbol{y}_{1:T} = p_n(\boldsymbol{y}'_{1:T}). \quad (7.7)$$

AIS proceeds by a series of P runs, one per sample. Given the definition of $p_n(\boldsymbol{y}_{1:T})$ and $T_n(\boldsymbol{y}_{1:T}, \boldsymbol{y}'_{1:T})$, a single AIS run proceeds as follows:

- Sample $\boldsymbol{y}_{1:T}^0$ from $p_0 = p_A$
- **Repeat for** $n = 1, 2, \ldots, N-1$:
  Sample $\boldsymbol{y}_{1:T}^n$ from $\boldsymbol{y}_{1:T}^{n-1}$ using T_{n-1}
- Set $w^{(p)} = \frac{p_1^*(\boldsymbol{y}_{1:T}^0)}{p_0^*(\boldsymbol{y}_{1:T}^0)} \frac{p_2^*(\boldsymbol{y}_{1:T}^1)}{p_1^*(\boldsymbol{y}_{1:T}^1)} \cdots \frac{p_{N-1}^*(\boldsymbol{y}_{1:T}^{N-2})}{p_{N-2}^*(\boldsymbol{y}_{1:T}^{N-2})} \frac{p_N^*(\boldsymbol{y}_{1:T}^{N-1})}{p_{N-1}^*(\boldsymbol{y}_{1:T}^{N-1})}$

After performing P runs of AIS we can estimate Z_B/Z_A using Eq. 7.6. The variance of our estimate will depend on the number of runs, P, and the number of intermediate distributions, N. For more details, see (Neal, 2001).

7.3.3 AIS for POHMMs

AIS for any particular model depends on the form of the intermediate distributions, $p_n(\boldsymbol{y}_{1:T})$, and the Markov chain transition operator, $T_n(\boldsymbol{y}_{1:T}, \boldsymbol{y}'_{1:T})$. A general form for the intermediate distributions is:

$$p_n(\boldsymbol{y}_{1:T}) \propto p_A^*(\boldsymbol{y}_{1:T})^{(1-\beta_n)} p_B^*(\boldsymbol{y}_{1:T})^{\beta_n} \tag{7.8}$$

where $0 = \beta_0 < \beta_1 < \ldots < \beta_N = 1$ is the user-specified annealing schedule. If p_A and p_B are defined by PoHMMs, this leads to an unnormalized distribution for which both evaluating p_n^* and sampling are not straightforward. However, we can use the following distribution:

$$\begin{aligned} p_n(\boldsymbol{y}_{1:T}) &= \frac{p_n^*(\boldsymbol{y}_{1:T})}{Z_n(T)} \\ &= \left[\prod_{m=1}^{M_A} \sum_{\{s_t^{(m)}\}} p^{(m)}(s_1^{(m)}) \prod_{t=2}^{T} p^{(m)}(s_t^{(m)}|s_{t-1}^{(m)}) \prod_{t=1}^{T} p^{(m)}(\boldsymbol{y}_t|s_t^{(m)})^{(1-\beta_n)} \right] \\ &\times \left[\prod_{m=1}^{M_B} \sum_{\{s_t^{(m)}\}} p^{(m)}(s_1^{(m)}) \prod_{t=2}^{T} p^{(m)}(s_t^{(m)}|s_{t-1}^{(m)}) \prod_{t=1}^{T} p^{(m)}(\boldsymbol{y}_t|s_t^{(m)})^{\beta_n} \right] \Big/ Z_n(T) \end{aligned} \tag{7.9}$$

where m, n index the M_A, M_B HMMs in PoHMM A, B, respectively. Eq. 7.9 defines a PoHMM with $M_A + M_B$ chains, one group whose emission distributions are scaled by $1 - \beta_n$ and the other group whose emission distributions are scaled by β_n. When $\beta_n = 0$, p_n reduces to PoHMM A with M_A chains, and when $\beta_n = 1$, p_n reduces to PoHMM B with M_B chains.

Similar to (Salakhutdinov and Murray, 2008), $T_n(\boldsymbol{y}_{1:T}, \boldsymbol{y}'_{1:T})$ is defined by alternating Gibbs sampling between the observed variables and the $M_A + M_B$ hidden states:

- Given the current sample, $\boldsymbol{y}_{1:T}$, perform inference in each of the M_A experts in PoHMM A using the forward-backward algorithm with the scaled emission distribution, $p^{(m)}(\boldsymbol{y}_t|s_t^{(m)})^{(1-\beta_n)}$. Sample a hidden state sequence, $\{s_t^{(m)}\}$, from the posterior.
- Repeat the above step for each of the M_B HMMs in PoHMM B, but using $p^{(m)}(\boldsymbol{y}_t|s_t^{(m)})^{\beta_n}$.
- Conditional on $\{s_t^{(m)}\} \forall m \in M_A$ and $\{s_t^{(m)}\} \forall m \in M_B$, the contribution to the distribution over observables from each of the HMMs will be

$p^{(m)}(\boldsymbol{y}_t|s_t^{(m)})^{(1-\beta_n)}$ and $p^{(m)}(\boldsymbol{y}_t|s_t^{(m)})^{\beta_n}$, respectively. For each t, multiply the $(M_A + M_B)$ conditional distributions together and renormalize.

- Draw $\boldsymbol{y}'_{1:T}$ from the normalized conditional distribution over observables.

Our definition of $T_n(\boldsymbol{y}_{1:T}, \boldsymbol{y}'_{1:T})$ leaves p_n invariant. Since p_n defines a PoHMM, p_n^* is easily evaluated via the forward-backward algorithm using the scaled emission probability distributions, as in inference. The re-sampling and likelihood estimation steps necessary for AIS can be combined into a single forward-backward pass (per annealing step) in each HMM. Although we can obtain samples from p_A by alternating Gibbs sampling, they will not be independent. Though it is preferable to use independent samples, AIS will still converge to the correct estimate, provided that the Markov chain is ergodic (Neal, 2001).

If we wish to calculate $Z_B(T)$ instead of $Z_B(T)/Z_A(T)$, we can select PoHMM A to have a $Z_A(T)$ that is easily evaluated. For example, if $M_A = 1$ then PoHMM A is just an HMM and $Z_A = 1$. In practice, we use a single-state HMM whose emission distribution is estimated to be the base rates of the training data, smoothed so that $p_A(\cdot)$ is never zero. We choose the β_k such that we gently anneal the base-rate HMM to the PoHMM of interest. This is why in Eq. 7.9 we do not need to scale the terms related to the dynamics: the dynamics are completely defined by the PoHMM.

7.4 Modeling daily rainfall occurrence

Precipitation modeling is of interest to fields such as hydrology, geophysics, and agriculture. Existing models are challenged by the high variance of observations over time, as well as the spatially local effects present in the data. Recently, HMMs have been proposed as a way to describe daily rainfall occurrence data collected at several weather stations (Robertson et al., 2004; Kirshner, 2005), as they are able to capture both spatial dependencies between weather stations and temporal regularities. The hidden states of the HMMs can also be interpreted as representing "wet-dry" or directional effects and the state sequences can be analyzed for the existence of seasonal, intraseasonal, interannual and longer-scale time patterns. Furthermore, the generative nature of the HMM allows for the production of station-scale daily rainfall simulations for input into other systems such as crop models.

For comparison purposes, we considered a rainfall dataset that has already been extensively analyzed in the context of HMMs. The Ceará data set contains 24 sequences of 90 binary observations for each of 10 weather stations located in northeast Brazil (Fig. 7.1). The sequences correspond to the wet seasons from the years 1975-2002, however, four of the years (1976, 78, 84, 86) contained a significant number of missing observations and were removed. All seasons start Feb 1 and end in April. As a baseline model, we considered a simple HMM with Bernoulli emission distributions on each dimension that are conditionally independent, given the state (denoted HMM-CI). In this model, the spatial dependencies between weather stations are captured entirely by the hidden state, and the emission parameters for each dimension can be estimated independently. Robertson et al. (2004) and Kirshner (2005) explored two ways of extending this HMM baseline. The first was to employ an IOHMM, conditioning the transition distributions of the HMM on the output of a coarser-resolution General Circulation Model that predicted average seasonal precipitation. Secondly, they experimented with emission distributions with richer dependency structure which can capture interactions not only between weather stations on a given day, but between neighbouring observations. In both cases, the multinomial hidden state was left intact. Instead, we propose maintaining the simple conditionally-independent Bernoulli emission distribution but working with a much richer, componential hidden state.

7.4.1 Toy experiment

As a first experiment, we trained Products of HMMs on a subset of the Ceará binary rainfall data. The number of dimensions and length of sequences were small enough to permit us to calculate the partition function exactly, by evaluating the likelihood of every possible length-T binary sequence. All PoHMMs were trained for 2000 epochs using CD-1.

The first PoHMM (two two-state HMMs) was trained on all 10 dimensions, but the sequences of length 90 were converted into subsequences of length two before training. This resulted in 2^{20} possible length-two sequences.

The second PoHMM (three three-state HMMs) was trained on only the first five dimensions, and the sequences of length 90 were converted into subsequences of length five before training. This resulted in 2^{25} length-five sequences. We split up the calculation of the true partition function onto

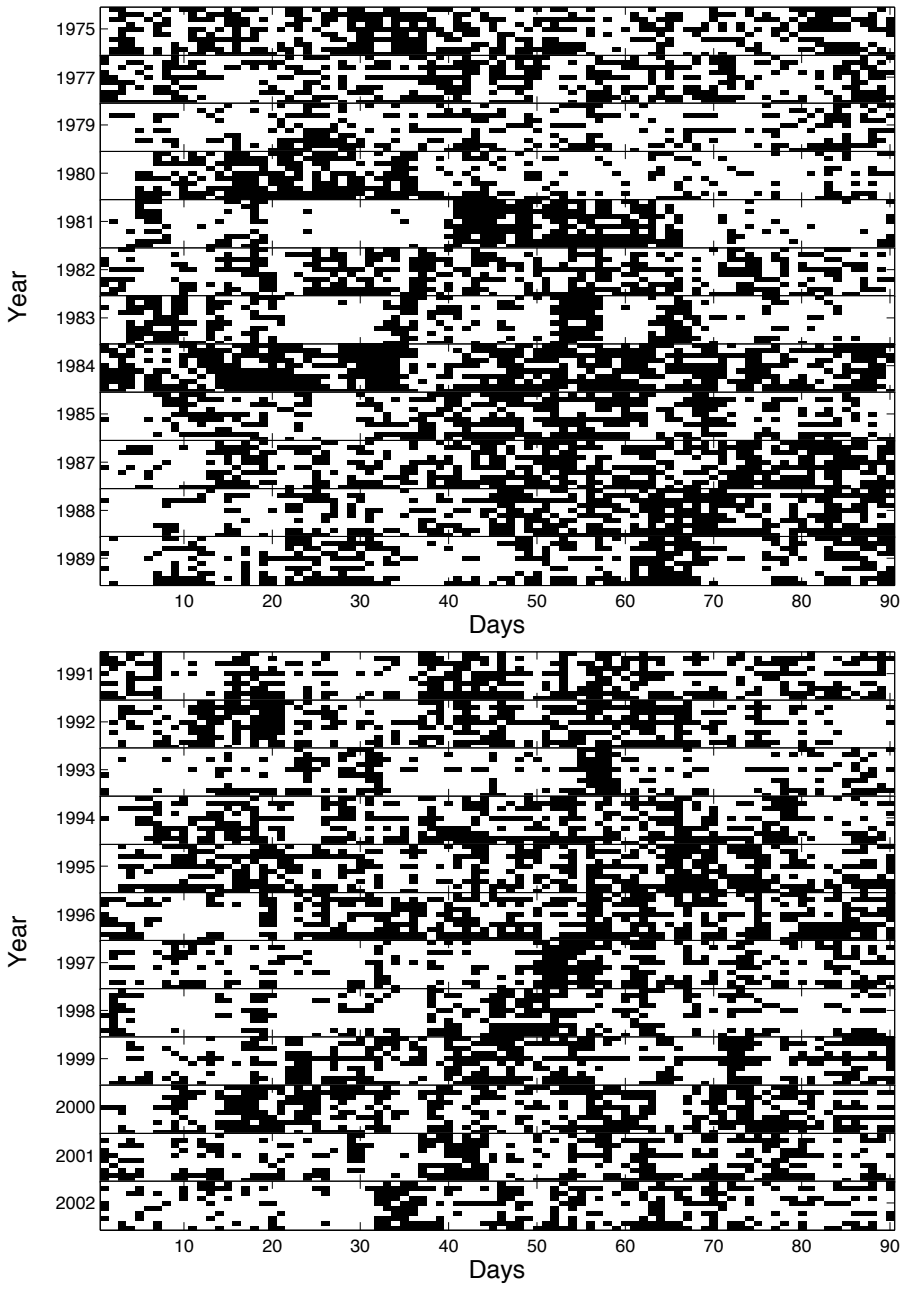


Figure 7.1. Visualization of the Ceará binary rainfall data. For each year, ten individual stations are shown top to bottom. Black indicates rain observed at that station while white indicates no rain for that day.

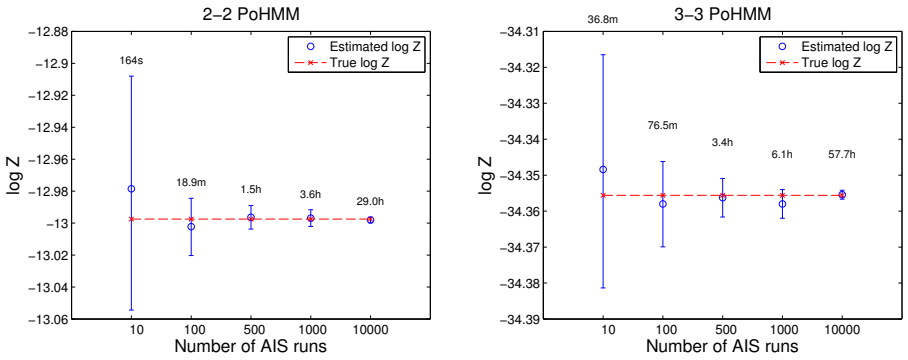


Figure 7.2. Estimates of the log partition functions $\log \hat{Z}$ of two different PoHMMs as we increase the number of annealing runs. The error bars show $\log(\hat{Z} \pm 3\hat{\sigma})$.

multiple processors.

Fig. 7.2 shows the results of the toy experiment. We used an annealing schedule of 5000 uniformly-spaced bins and vary the number of annealing runs. For both PoHMMs, AIS gives reliable estimates of the true log partition function, and as expected, the estimated variance decreases as we increase the number of annealing runs.

7.4.2 Comparison to HMMs

We compared baseline HMMs of {2,3,...,8} states to a product of two HMMs of {2,3,...,6} states and a product of three HMMs of {2,3,...,6} states. In addition to HMM-CI (as noted above), we experimented with a first-order Autoregressive HMM with dependence on the previous observation for the same station (HMM-Chains), an HMM with Chow-Liu tree emissions (HMM-CL), an HMM with conditional Chow-Liu forest emissions (HMM-CCL) and an HMM whose output distribution is a full bivariate maximum entropy model (HMM-MaxEnt). Note that we use the same notation as Kirshner (2005) for the various HMMs.

Parameters were initialized randomly. The HMMs were trained with the EM algorithm until convergence was observed in the training log likelihood. The POHMMs were trained with CD-1 for a fixed number of epochs at a conservative learning rate (0.001 for all parameters). Following Kirshner

(2005), we used *leave-6-out* cross-validation. Under *leave-6-out* we trained four models, one for each set obtained by leaving out six non-overlapping consecutive sequences. Each model was evaluated on the corresponding left-out six sequences. We report the mean of the four models. This entire process was repeated 100 times (using different random initializations) and we report the mean of these runs. We employed four different evaluation metrics:

- **Scaled log likelihood:** For the HMM, the log likelihoods of the held out sequences were computed exactly using the forward-backward algorithm. For the PoHMMs, we used 100 AIS runs with 500 intermediate distributions of uniform spacing to estimate the partition function. Log likelihoods of the held out sets were divided by the number of binary events occurring in the held out data ($6 \times 90 \times 10$).

- **Classification accuracy:** We report the average classification error in predicting observations that were removed (one dimension, one sequence at a time) from the held out data. Since the observations were binary, we simply compared the (unnormalized) log likelihood of the sequence with the correct observation filled in vs. the incorrect observation.

- **Difference in precipitation persistence for observed and simulated data:** Persistence is defined as the probability of a precipitation occurrence for a particular station given a precipitation event at that station at the previous observation. We simply compared the mean of this result over 500 simulated 90-day sequences to the mean calculated over the held-out observations and report the absolute difference.

- **Difference in linear correlation for observed and simulated data:** The spatial correlation between a pair of stations is defined as Pearson's correlation coefficient of their respective daily rainfall binary occurrences. We computed this value separately for the held-out sequences and the 500 simulated sequences and report the absolute difference.

The results are shown in Fig. 7.3. Based on normalized log likelihood of held-out data and classification accuracy, the PoHMMs outperformed the various HMMs with a similar number of parameters. In the case of log likelihood, the PoHMMs also exhibited less overfitting of the training data (though this could also be an artifact of fixed number of epochs vs. training until

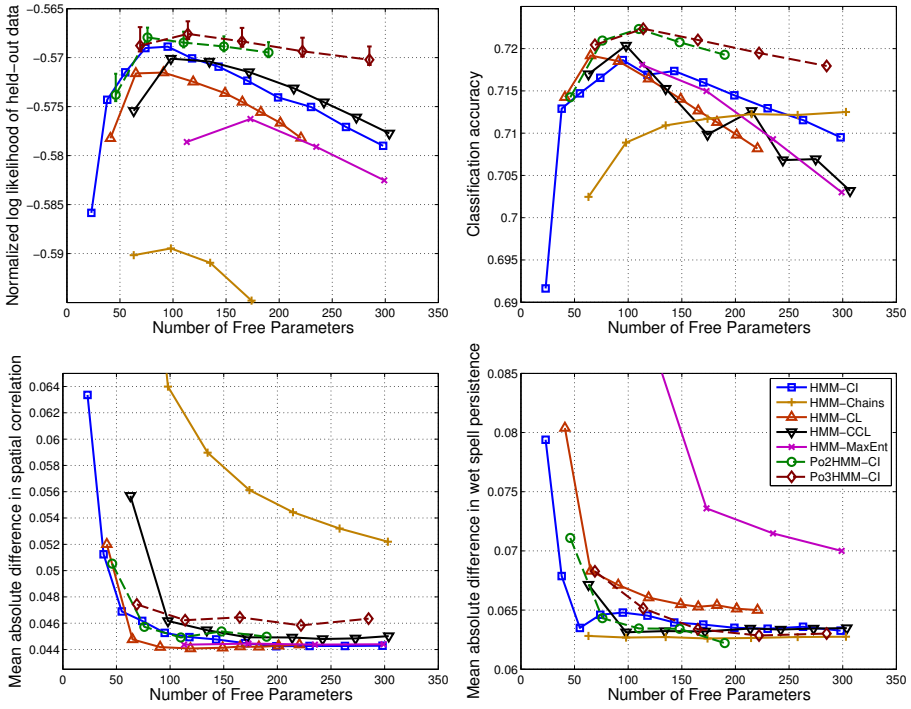


Figure 7.3. Rainfall modeling. We have compared several types of HMMs (solid lines), to a product of two or three HMMs (dashed lines) under a variety of metrics. In the top two plots, higher is better. In the bottom two plots, lower is better. Log likelihoods are per-sequence, per-frame, per-dimension. Error bars of three standard deviations of the partition function estimate are plotted for log likelihood. Note that the single legend applies to all plots.

convergence). The PoHMMs performed well on the persistence task, though to a lesser degree. Not surprisingly, the best performing model according to the difference in persistence was HMM-Chains which has stronger temporal structure. PoHMMs did not perform as well according to the difference in spatial correlation. Perhaps this was to be expected as they retained the same simple emission model as HMM-CI. Models with stronger spatial structure (HMM-CL, HMM-MaxEnt) performed better under this metric.

If we examine the emission parameters of the PoHMMs, we can see that individual HMMs specialize locally (Fig. 7.4). Two states of HMM1 are specializing on the eastern weather stations 7 and 2. All states of HMM2 are

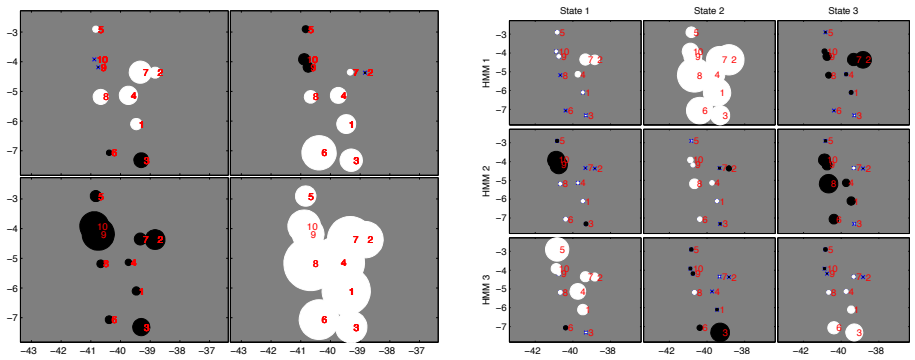


Figure 7.4. Log-domain emission parameters of a four-state HMM (left) and product of three three-state HMMs (right). Each rectangle reflects the spatial layout of the numbered weather stations (input dimensions). White represents a belief in a dry observation while black represents a belief in a wet observation, conditional on that state. The larger the radius, the stronger the belief.

more tuned towards the western weather stations 10,9,8,4,1 and 6. In HMM3, we see that two of the states are specialized to the southern weather stations 6 and 3, while the other state is tuned to the remaining northern stations.

7.5 Modeling human dance

We again turn to data captured from human motion (mocap) to explore the capabilities of HMMs and Products of HMMs. Motion such as a single subject walking or jogging exhibits strong global coherence and should not pose as great a difficulty to an HMM as it can allocate subsets of its states to the various regimes. At each frame, it is not difficult to predict what one part of the body is doing from another, as the motion is quite regular. The advantages of componential state should be more apparent in modeling data that is the result of multiple underlying processes, such as mocap from multiple subjects.

Thus, we have carried out a series of experiments where we have compared HMMs and PoHMMs on data with both *weakly* and *strongly* componential structure. We started with mocap data of a pair of people salsa dancing. We used all frames from subjects 60 and 61 in the CMU Graphics Lab motion capture database. To facilitate training and comparison, the data was partitioned

into fixed-length sequences of 100 frames. The ordering of sequences was randomly permuted so that training and test sets were balanced and randomly assigned. We set aside 23 sequences for testing and used the remaining 48 sequences for training. We used the data representation described in Sec. 3.2 where all joint angles were encoded via the exponential map and a local co-ordinate system was employed to ensure invariance to translations in the ground plane and rotations about the gravitational vertical. In order to work with discrete HMMs while still maintaining the multivariate and componential properties of the mocap data, we performed a type of vector quantization. Within each subject, dimensions were assigned to one of six local groups: Left leg, Right leg, Torso, Upper body, Left arm and Right arm. We applied K-means clustering with $K = 25$ independently to the vectors corresponding to each group, across all frames. This resulted in 6 or 12-dimensional discrete sequences for one or two subjects, respectively.

All emission distributions were 6 or 12 25-symbol multinomials conditionally independent given hidden state. Baseline HMMs of 2^N states where $N = 2, 3, \ldots, 6$ were trained with EM until convergence based on training log likelihood. We trained several PoHMM models, where the number of experts and states were selected such that the number of free parameters were comparable to the baseline HMMs. All PoHMMs were trained using CD-10 which gave slightly better results than our initial models obtained using CD-1. Each PoHMM was trained for 1000 epochs, with a learning rate of 0.01 for all parameters. Momentum was also used: 0.9 of the previous accumulated gradient was added to the current gradient.

Log likelihoods under the HMMs were computed exactly. To estimate log likelihoods under the PoHMMs, we used 100 AIS runs. Quick initial runs encouraged us to set a final annealing schedule of 5000 uniformly spaced intervals. Figure 7.5 shows our results obtained modeling single-subject mocap data (weakly componential) and two-subject mocap data (strongly componential). All log likelihoods are per sequence, and have been normalized by dividing by the number of dimensions (6 and 12 respectively) so that the two plots are comparable. In both cases, products of HMMs compare favourably to the baseline. As expected, the advantage is more evident when the data has strongly componential structure. This is achieved despite a weaker gradient-based method of training. We note that HMMs trained by

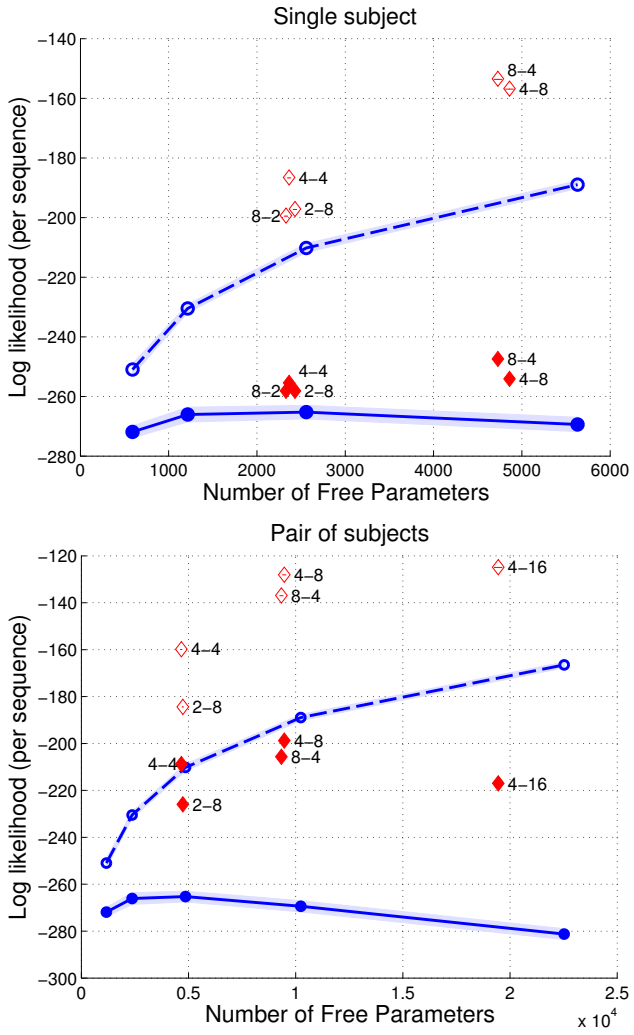


Figure 7.5. Modeling a single subject (top) and partners (bottom) salsa dancing. Open and solid circles correspond to the log likelihood scores of the baseline HMMs on the training and test data. Open and solid diamonds denote training and test log likelihood scores for the PoHMMs. The small text beside the PoHMM scores indicates the number of HMMs and number of states per HMM in the product. Again error bars corresponding to three standard deviations from the partition function estimate have been plotted (visible in open diamonds) but they are so tight that they are difficult to see at this scale. Shading indicates unit standard deviation error across HMM models.

conjugate-gradient or CD-1 gave worse baseline results than EM. All of these methods rely on inference via the forward-backward algorithm whose running time is quadratic in the number of hidden states. This suggests that combining HMMs with a smaller number of states via a product where training time is linear in the number of HMMs is a sensible thing to do. However, the cost of gradient-based training compared to EM means that this advantage will not be seen until the number of states are considerably higher than those considered in our experiments. All of the PoHMMs we trained took on the order of an hour, while the HMMs were trained within a few minutes.

7.6 Discussion

Advances in computing hardware and improvements to the CD learning algorithm have made Products of Hidden Markov Models a more attractive option for time series modeling than when they were first introduced. This is especially true for data that is high-dimensional and has strongly componential structure. With a reliable means of approximating the intractable partition function, log likelihood estimates of sequences under the model can aid in model selection, complexity control and comparing PoHMMs to other generative models.

In this chapter, we have departed from the domain of "toy" problems and demonstrated the effectiveness of PoHMMs on real multivariate data. In precipitation modeling, PoHMMs were shown to improve performance over HMMs of a similar number of parameters under a variety of metrics and also captured interesting local regularities in the data. When trained on data captured from human motion, PoHMMs were also shown to compare favourably to HMMs in terms of log likelihood. This was most apparent when the data was a product of multiple underlying influences.

8 Summary and Future Work

Modeling time series data has engaged researchers since Yule first proposed the autoregressive method. In the following decades, the amount and complexity of available data has increased but so has the computational power for its analysis. Chapter 1 introduced some of the challenges in this field as well as the properties good models should exhibit. Distributed hidden state, undirected graphical models and achieving deep, hierarchical representations by composing simpler models are the motivating themes we put forth. Chapter 2 reviewed the relevant literature on time series analysis and enables the reader to understand the models we later present in the context of existing methods.

Modeling human motion is the most prominent field to which we apply our models and so we introduced motion capture (mocap) and state-of-the-art motion synthesis methods in Chapter 3. We also derived an invariant representation for mocap that highlights the most statistically salient features of the data and suppresses irrelevant variation. The representation should be useful beyond the work discussed herein.

Chapter 4 presented the Conditional Restricted Boltzmann Machine (CRBM). The key properties of the CRBM are that it permits rich distributed representations to be learned from time series, and that exact inference is simple and efficient. We derived the contrastive divergence (CD) learning rules for CRBMs and showed how CRBMs can be stacked to form conditional deep belief nets. We demonstrated that a single model can generate many different styles of motion. We also used the CRBM to synthesize video textures which

hints at its applicability to other high-dimensional time series.

Perhaps the two greatest limitations of CRBMs (and RBMs in general) are first, evaluating the quality of trained models, and second, the learning algorithm with which they are trained. Though we have explored different methods of model evaluation, such as N-step forward prediction and the subjective assessment of synthesized data, the most natural way to evaluate a generative model is to compute the log-likelihood it assigns to a held-out test set. For all but the smallest models, this is impossible to do exactly due to the intractability of computing the partition function. Salakhutdinov and Murray (2008) have successfully applied Annealed Importance Sampling (AIS) to RBMs. However, conditioning changes the partition function which implies that we would need to perform AIS for every possible configuration of N-frame histories (where N is the order of the CRBM) if we wish to evaluate the likelihood assigned by the model to an arbitrary sequence. Fortunately, to evaluate models we are often interested in computing likelihoods for a fixed test set rather than arbitrary sequences. This means that we need only to concern ourselves with conditioning on all possible N-frame histories in the test set. If we are evaluating M sequences whose maximum length is T, we would need to make in the order of $M\,(T - N)$ complete AIS estimates[†].

A major criticism of contrastive divergence learning is that by "pulling up" on the energy of individual reconstructed data points, the algorithm fails to visit regions far away from the training data. Consequently, the bulk of the energy surface is left arbitrarily low. One solution is to abandon CD altogether, and pursue other learning methodologies, such as the sparse "energy-based methods" discussed in Sec. 4.5.4. The alternative is to improve CD. Tieleman (2008) recently introduced Persistent CD (PCD) which was shown to outperform both CD-1 and CD-10 at a computational cost comparable to CD-1. The idea behind PCD is to exploit the fact that the weights change slowly by not resetting the Markov chain at each weight update. We retain "fantasy" data equal to the size of a "mini-batch" which transitions one Gibbs step per weight update. As we reduce the learning rate, the fantasies asymptotically approach true samples from the model rather than reconstructions. Applying PCD to conditional models is not immediately straightforward as the fantasy

[†]Note that for each of these "conditional" estimates we would still perform P runs of AIS (see Sec. 7).

data is itself conditional on the inputs. Retaining a single fantasy point per training case and updating only with that training case (as opposed to sharing fantasy data across mini-batches) diminishes the fast mixing that makes PCD attractive. Deriving a form of PCD for conditional models is something that we would like to pursue.

In Chapter 5 we extended the CRBM to permit context units to modulate the existing pairwise interactions. The resulting multiplicative model implies cardinality of parameters cubic in the number of units. However, we factorized the weights to make the parameterization quadratic and further reduced this number by tying weights. We demonstrated that the resulting model could capture several different motion styles, as well as transition and blend naturally between them. A sensible and natural extension of this work is to the fully unsupervised setting, where stylistic parameters are learned rather than provided (c.f. Brand and Hertzmann, 2000).

Chapter 6 described how the CRBM can integrate with Bayesian filtering to track people in 3D. Our experiments demonstrated that the prior aided considerably in challenging tasks such as multi-view tracking across transitions and monocular tracking. We expect that our results can further be improved by adding additional layers of hidden units to our dynamical priors. Future work includes augmenting our model with identity or action variables (the context) and performing inference over the joint space of pose and context. Factored, multiplicative models (Sec. 5) should play a key role in this extension.

In Chapter 7 we applied Annealed Importance Sampling to estimate the partition function of Products of Hidden Markov Models (PoHMMs). Our main contribution has been to propose a series of intermediate distributions and a Markov chain transition operator that makes AIS on PoHMMs feasible. We also conducted an empirical comparison between PoHMMs and HMMs on binary rainfall occurrence data and motion captured from pairs of people dancing. Worth noting is that the use of AIS for PoHMMs was first suggested by Brown (2001) in his doctoral thesis. Nearly a decade later, we have confirmed, quantitatively, that PoHMMs are better generative models than similarly-sized HMMs.

This thesis has focused on developing generative models for time series. We have empirically investigated the proposed models in domains such as motion synthesis and tracking to demonstrate their key properties. While

we have emphasized the composable nature of the CRBM and its variants, we have yet to perform a quantitative comparison between shallow and deep architectures. Salakhutdinov and Murray (2008) have made progress for static models, but as already mentioned, applying AIS to conditional models is computationally much more expensive. At this point, we can state that adding a second hidden layer greatly improved the quality of synthesized transitions (Sec. 4.5.2) and permitted us to capture subtle movement (Sec. 4.5.4). Despite solid theoretical arguments (Hinton et al., 2006; Hinton, 2007a; Le Roux and Bengio, 2008), much more work is required, in both the static and temporal domains, to quantify the benefit of using deep, hierarchical representations. We expect this to be a major focus of research over the next few years.

References

O. Arikan and D. A. Forsyth. Interactive motion generation from examples. In *Proceedings of the 29th annual conference on computer graphics and interactive techniques (SIGGRAPH 2002)*, pages 483–490. ACM Press, 2002.

O. Arikan, D. A. Forsyth, and J. F. O'Brien. Motion synthesis from annotations. In *Proceedings of the 30th annual conference on computer graphics and interactive techniques (SIGGRAPH 2003)*, pages 402–408. ACM Press, 2003.

A. Balan, L. Sigal, and M. Black. A quantitative evaluation of video-based 3D person tracking. In *Proceedings of the 2nd IEEE Joint Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance (VS-PETS 2005)*, pages 349–356, 2005.

A. Balan, L. Sigal, M. J. Black, J. Davis, and H. Haussecker. Detailed human shape and pose from images. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*. IEEE, 2007.

S. Bengio and Y. Bengio. An EM algorithm for asynchronous input/output hidden Markov models. In L. Xu, editor, *Proceedings of the International Conference On Neural Information Processing*, pages 328–334, Hong-Kong, 1996.

Y. Bengio. Markovian models for sequential data. *Neural Computing Surveys*, 2: 129–162, 1999.

Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(1):1–21, 2008.

Y. Bengio and P. Frasconi. An input/output HMM architecture. In G. Tesauro, D. S. Touretzky, and T. K. Leen, editors, *Advances in Neural Information Processing Systems (NIPS 7): Proceedings of the 1994 Conference*, pages 427–434. MIT Press, 1995.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS 19): Proceedings of the 2006 Conference*, pages 153–160. MIT Press, 2007.

A. Bissacco. Modeling and learning contact dynamics in human motion. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, pages 421–428. IEEE, 2005.

M. Brand and A. Hertzmann. Style machines. In *Proceedings of the 27th annual conference on computer graphics and interactive techniques (SIGGRAPH 2000)*, pages 183–192. ACM Press, 2000.

A. D. Brown. *Product Models for Sequences*. PhD thesis, University of Toronto, 2001.

A. D. Brown and G. E. Hinton. Products of hidden Markov models. In T. Jaakkola and T. Richardson, editors, *Proceedings of the 8th International Conference on Artificial Intelligence and Statistics (AISTATS 2001)*, pages 3–11. Morgan Kaufmann, 2001.

N. Campbell, C. Dalton, D. Gibson, D. Oziem, and B. Thomas. Practical generation of video textures using the auto-regressive process. *Image and Vision Computing*, 22(10):819–827, 2004.

M. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. In *Proceedings of the 10th International Conference on Artificial Intelligence and Statistics (AISTATS 2005)*, pages 59–66, 2005.

V. Cheung, B. J. Frey, and N. Jojic. Video epitomes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, pages 42–49. IEEE, 2005.

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley-Interscience, 2nd edition, 2006.

J. Deutscher and I. Reid. Articulated body motion capture by stochastic search. *International Journal of Computer Vision*, 61(2):185–205, 2005.

G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003.

A. Doucet, S. Godsill, and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

J. L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990.

D. Forsyth, O. Arikan, L. Ikemoto, J. O'Brien, and D. Ramanan. Computational studies of human motion: Part 1, tracking and motion synthesis. *Foundations and Trends in Computer Graphics and Vision*, 1(2–3):77–254, 2006.

Y. Freund and D. Haussler. Unsupervised learning of distributions of binary vectors using 2-layer networks. In J. Moody, S. H. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems (NIPS 4): Proceedings of the 1991 Conference*, pages 912–919. Morgan-Kaufmann, 1992.

B. J. Frey, P. Dayan, and G. E. Hinton. A simple algorithm that discovers efficient perceptual codes. In M. Jenkin and L. R. Harris, editors, *Computational and Psychophysical Mechanisms of Visual Coding*. Cambridge University Press, 1997.

P. V. Gehler, A. D. Holub, and M. Welling. The rate adapting poisson model for information retrieval and object recognition. In *Proceedings of the 23rd international conference on Machine Learning (ICML 2006)*, pages 337–344. ACM Press, 2006.

Z. Ghahramani. Learning dynamic Bayesian networks. In C. Giles and M. Gori, editors, *Adaptive Processing of Sequences and Data Structures*, pages 168–197. Springer-Verlag, Berlin, 1998.

Z. Ghahramani and G. E. Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2, University of Toronto, 1996.

Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12(4):831–864, 2000.

Z. Ghahramani and M. I. Jordan. Factorial hidden markov models. *Machine Learning*, 29(2–3):245–273, 1997.

F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10):428–434, 2007a.

G. E. Hinton. To recognize shapes, first learn to generate images. In T. D. P. Cisek and J. Kalaska, editors, *Computational Neuroscience: Theoretical insights into brain function*. Elsevier, 2007b.

G. E. Hinton and A. D. Brown. Spiking Boltzmann machines. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems (NIPS 12): Proceedings of the 1999 Conference*, pages 122–128. MIT Press, 2000.

G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

G. E. Hinton and T. J. Sejnowski. Learning and relearning in Boltzmann machines. In D. E. Rumelhart, J. L. McClelland, et al., editors, *Parallel Distributed Processing: Volume 1: Foundations*, pages 282–317. MIT Press, Cambridge, MA, 1987.

G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

E. Hsu, K. Pulli, and J. Popović. Style translation for human motion. In *Proceedings of the 32nd annual conference on computer graphics and interactive techniques (SIGGRAPH 2005)*, pages 1082–1089. ACM Press, 2005.

M. Isard and A. Blake. CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

R. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

M. I. Jordan. Serial order: A parallel distributed processing approach. ICS report 8608, Institute for Cognitive Science, UCSD, La Jolla, 1986.

S. Kirshner. *Modeling of multivariate time series using hidden Markov models*. PhD thesis, University of California, Irvine, 2005.

L. Kovar and M. Gleicher. Automated extraction and parameterization of motions in large data sets. In *Proceedings of the 31st annual conference on computer graphics and interactive techniques (SIGGRAPH 2004)*, pages 559–568. ACM Press, 2004.

L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings of the 29th annual conference on computer graphics and interactive techniques (SIGGRAPH 2002)*, pages 473–482. ACM Press, 2002.

H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th international conference on Machine Learning (ICML 2007)*, pages 473–480. ACM Press, 2007.

N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS 16): Proceedings of the 2003 Conference*, pages 329–326. MIT Press, 2004.

N. D. Lawrence. Learning for larger datasets with the gaussian process latent variable model. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, pages 243–250, 2007.

N. D. Lawrence and A. J. Moore. Hierarchical gaussian process latent variable models. In Z. Ghahramani, editor, *Proceedings of the 24th international conference on Machine Learning (ICML 2007)*, pages 481–488. ACM Press, 2007.

N. D. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. In S. Becker, S. Thrun, and

K. Obermayer, editors, *Advances in Neural Information Processing Systems (NIPS 15): Proceedings of the 2002 Conference*, pages 625–632. MIT Press, 2003.

N. Le Roux and Y. Bengio. Representational power of restricted Boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

C. Lee and A. Elgammal. Modeling view and posture manifolds for tracking. In *Proceedings of the 10th International Conference on Computer Vision (ICCV 2007)*. IEEE, 2007.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

H. Lee, C. Ekanadham, and A. Y. Ng. Sparse deep belief net model for visual area V2. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS 20): Proceedings of the 2007 Conference*, pages 873–880. MIT Press, 2008.

J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on computer graphics and interactive techniques (SIGGRAPH 2002)*, pages 491–500. ACM Press, 2002.

R. Li, T. Tian, and S. Sclaroff. Simultaneous learning of non-linear manifold and dynamical models for high-dimensional time series. In *Proceedings of the 10th International Conference on Computer Vision (ICCV 2007)*. IEEE, 2007.

Y. Li, T. Wang, and H.-Y. Shum. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of the 29th annual conference on computer graphics and interactive techniques (SIGGRAPH 2002)*, pages 465–472. ACM Press, 2002.

R.-S. Lin, C.-B. Liu, M.-H. Yang, N. Ahuja, and S. Levinson. Learning nonlinear manifolds from time series. In *Proceedings of the 9th European Conference on Computer Vision (ECCV 2006)*, pages 245–256, 2006.

C. K. Liu, A. Hertzmann, and Z. Popovic. Learning physics-based motion style with nonlinear inverse optimization. In *Proceedings of the 32nd annual conference on computer graphics and interactive techniques (SIGGRAPH 2005)*, pages 1071–1081. ACM Press, 2005.

Z. Lu, M. Carreira-Perpinan, and C. Sminchisescu. People tracking with the Laplacian eigenmaps latent variable model. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS 20): Proceedings of the 2007 Conference*, pages 1705–1712. MIT Press, 2008.

R. Memisevic. *Non-linear latent factor models for revealing structure in high-dimensional data*. PhD thesis, University of Toronto, 2008.

R. Memisevic and G. E. Hinton. Unsupervised learning of image transformations. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, 2007.

T. Minka. From hidden Markov models to linear dynamical systems. Technical report, MIT, 1999.

A. Mnih and G. E. Hinton. Three new graphical models for statistical language modelling. In Z. Ghahramani, editor, *Proceedings of the 24th international conference on Machine Learning (ICML 2007)*, pages 641–648. ACM Press, 2007.

T. Mukai and S. Kuriyama. Geostatistical motion interpolation. In *Proceedings of the 32nd annual conference on computer graphics and interactive techniques (SIGGRAPH 2005)*, pages 1062–1070. ACM Press, 2005.

K. P. Murphy. *Dynamic bayesian networks : representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

R. Neal. Annealed importance sampling. *Statistics and Computing*, 11(2): 125–139, 2001.

A. Neumaier and T. Schneider. Estimation of parameters and eigenmodes of multivariate autoregressive models. *ACM Transactions on Mathematical Software*, 27(1):27–57, 2001.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

S. Osindero and G. E. Hinton. Modeling image patches with a directed hierarchy of Markov random fields. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS 20): Proceedings of the 2007 Conference*, pages 1121–1128. MIT Press, 2008.

S. I. Park, H. J. Shin, and S. Y. Shin. On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on computer animation (SCA 02)*, pages 105–111. ACM Press, 2002.

V. Pavlovic, J. M. Rehg, and J. MacCormick. Learning switching linear models of human motion. In *Advances in Neural Information Processing Systems (NIPS 13): Proceedings of the 2000 Conference*, pages 981–987. MIT Press, 2001.

V. Pavolvic, J. Rehg, T. J. Cham, and K. Murphy. A dynamic Bayesian network approach to figure tracking using learned dynamic models. In *Proceedings of the 2nd International Conference on Computer Vision (ICCV 1999)*, pages 94–101. IEEE, 1999.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, Santa Mateo, CA, USA, September 1988.

K. Pullen and C. Bregler. Motion capture assisted animation: texturing and synthesis. In *Proceedings of the 29th annual conference on computer graphics and interactive techniques (SIGGRAPH 2002)*, pages 501–508. ACM Press, 2002.

J. Quiñonero Candela and C. E. Rasmussen. A unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research*, 6:1939–1959, 2005.

L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

M. Ranzato, C. S. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS 19): Proceedings of the 2006 Conference*, pages 1137–1144. MIT Press, 2006.

M. Ranzato, Y. Boureau, S. Chopra, and Y. LeCun. A unified energy-based framework for unsupervised learning. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, 2007.

M. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS 20): Proceedings of the 2007 Conference*. MIT Press, 2008.

A. W. Robertson, S. Kirshner, and P. Smyth. Downscaling of daily rainfall occurrence over northeast Brazil using a hidden Markov model. *Journal of Climate*, 17(22):4407–4424, 2004.

C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5): 32–40, 1998.

S. Roweis and Z. Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2):305–345, 1999.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th international conference on Machine Learning (ICML 2008)*, pages 872–879. ACM Press, 2008.

R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted Boltzmann machines for collaborative filtering. In Z. Ghahramani, editor, *Proceedings of the 24th international conference on Machine Learning (ICML 2007)*, pages 791–798. ACM Press, 2007.

L. K. Saul and M. I. Jordan. Exploiting tractable substructures in intractable networks. In D. S. Touretzky, M. Mozer, and M. E. Hasselmo, editors, *Advances in Neural Information Processing Systems (NIPS 8): Proceedings of the 1995 Conference*, pages 486–492. MIT Press, 1996.

A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *Proceedings of the 27th annual conference on computer graphics and interactive techniques (SIGGRAPH 2000)*, pages 489–498. ACM Press, 2000.

R. Shumway and D. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4): 253–264, 1982.

H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3D human figures using 2D image motion. In *Proceedings of the 5th European Conference on Computer Vision (ECCV 2000)*, pages 702–718, 2000.

L. Sigal and M. Black. HumanEva: Synchronized video and motion capture dataset for evaluation of articulated human motion. Technical Report CS-06-08, Brown University, 2006.

L. Sigal, A. Balan, and M. J. Black. HumanEva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 2009. To Appear.

C. Sminchisescu and A. Jepson. Generative modeling for continuous non-linearly embedded visual inference. In *Proceedings of the 21st international conference on Machine Learning (ICML 2004)*, pages 759–766. ACM Press, 2004.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, et al., editors, *Parallel Distributed Processing: Volume 1: Foundations*, pages 194–281. MIT Press, Cambridge, MA, 1986.

I. Sutskever and G. E. Hinton. Learning multilevel distributed representations for high-dimensional sequences. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, 2007.

I. Sutskever and G. E. Hinton. Deep narrow sigmoid belief networks are universal approximators. *Neural Computation*, 20(11):2629–2636, 2008.

I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted Boltzmann machine. In *Advances in Neural Information Processing Systems (NIPS 21): Proceedings of the 2008 Conference*, volume 21. MIT Press, 2009.

L. Tanco and A. Hilton. Realistic synthesis of novel human movements from a database of motion capture examples. In *Proceedings of the Workshop on Human Motion (HUMO '00)*, pages 137–142. IEEE Computer Society, 2000.

Y. W. Teh and G. E. Hinton. Rate-coded restricted Boltzmann machines for face recognition. In *Advances in Neural Information Processing Systems (NIPS 13): Proceedings of the 2000 Conference*. MIT Press, 2001.

J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th international conference on Machine Learning (ICML 2008)*, pages 1064–1071. ACM Press, 2008.

T. Tieleman and G. E. Hinton. Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th international conference on Machine Learning (ICML 2009)*. ACM Press, 2009. To appear.

L. Torresani, P. Hackney, and C. Bregler. Learning motion style synthesis from perceptual observations. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS 19): Proceedings of the 2006 Conference*, pages 1393–1400. MIT Press, 2007.

R. Urtasun, P. Glardon, R. Boulic, D. Thalmann, and P. Fua. Style-based Motion Synthesis. *Computer Graphics Forum*, 23(4):1–14, 2004.

R. Urtasun, D. Fleet, A. Hertzmann, and P. Fua. Priors for people tracking from small training sets. In *Proceedings of the 8th International Conference on Computer Vision (ICCV 2005)*, pages 403–410. IEEE, 2005.

R. Urtasun, D. Fleet, and P. F. P. 3D people tracking with gaussian process dynamical models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, pages 238–245. IEEE, 2006.

R. Urtasun, D. J. Fleet, A. Geiger, J. Popović, T. Darrell, and N. D. Lawrence. Topologically-constrained latent variable models. In *Proceedings of the 25th international conference on Machine Learning (ICML 2008)*, pages 1080–1087. ACM Press, 2008.

R. van der Merwe. *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*. PhD thesis, OGI School of Science & Engineering, Oregon Health & Science University, 2004.

A. Waibel, T. Hanazawa, G. E. Hinton, K. Shikano, and K. J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, 1989.

E. Wan. Time series prediction using a neural network with embedded tapped delay-lines. In A. S. Weigend and N. A. Gershenfeld, editors, *Time Series*

Prediction: Forecasting the Future and Understanding the Past. Addison Wesley, 1994.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models. In *Advances in Neural Information Processing Systems (NIPS 18): Proceedings of the 2005 Conference*, pages 1441–1448. MIT Press, 2006.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Multifactor Gaussian process models for style-content separation. In *Proceedings of the 24th international conference on Machine Learning (ICML 2007)*, pages 975–982. ACM Press, 2007.

A. S. Weigend and N. A. Gershenfeld, editors. *Time Series Prediction: Forecasting the Future and Understanding the Past*. Addison Wesley, 1994.

M. Welling, M. Rosen-Zvi, and G. E. Hinton. Exponential family harmoniums with an application to information retrieval. In *Advances in Neural Information Processing Systems (NIPS 17): Proceedings of the 2004 Conference*, pages 1481–1488. MIT Press, 2005.

C. K. I. Williams and G. E. Hinton. Mean field networks that learn to discriminate temporally distorted strings. In D. S. Touretzky, J. L. Elman, T. J. Sejnowski, and G. E. Hinton, editors, *Connectionist Models: Proceedings of the 1990 Connectionist Summer School*. Morgan Kauffman, 1990.

C. S. Wong and W. K. Li. On a mixture autoregressive model. *Journal Of The Royal Statistical Society Series B-Statistical Methodology*, 62:95–115, 2000.

X. Xu and B. Li. Learning motion correlation for tracking articulated human body with a Rao-Blackwellised particle filter. In *Proceedings of the 8th International Conference on Computer Vision (ICCV 2005)*, pages 1–8. IEEE, 2007.