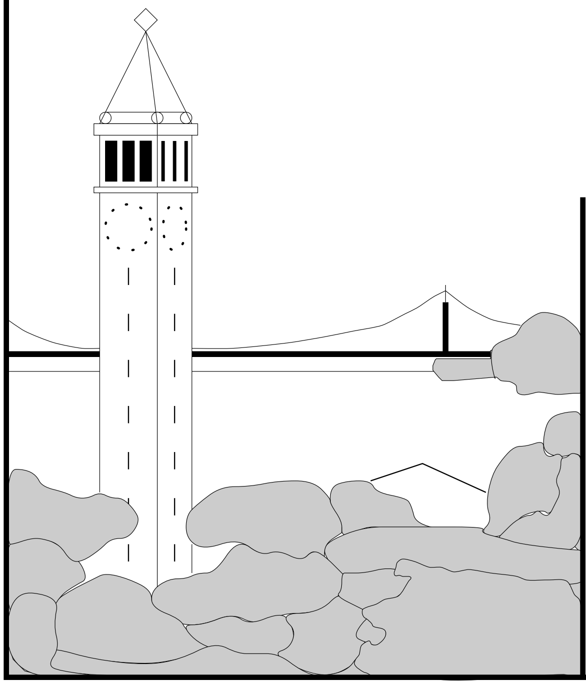
Compositional Modeling With DPNs

Geoffrey Zweig *Stuart Russell*



Report No. UCB/CSD-97-970

September 1997

Computer Science Division (EECS)
University of California
Berkeley, California 94720

Compositional Modeling With DPNs

Geoffrey Zweig * Stuart Russell †

Sept. 8, 1997

Abstract

Dynamic probabilistic networks (DPNs) are a powerful and efficient method for encoding stochastic temporal models. In the past, however, their use has been largely confined to the description of *uniform* temporal processes. In this paper we show how to combine specialized DPN models to represent *inhomogeneous* processes that progress through a sequence of different stages. We develop a method that takes a set of DPN submodels and a stochastic finite state automaton that defines a legal set of submodel concatenations, and constructs a composite DPN. The composite DPN is shown to represent correctly the intended probability distribution over possible histories of the temporal process. The use of DPNs allows us to take advantage of efficient, general-purpose inference and learning algorithms and can confer significant advantages over HMMs in terms of statistical efficiency and representational flexibility. We illustrate these advantages in the context of speech recognition.

1 Introduction

Many temporal processes evolve through a series of distinct stages, each of which is best represented by a separate model. For example, the process of uttering a word can be viewed as the sequential articulation of its constituent phonetic units. Similarly, the process of writing a word can be decomposed into the sequential formation of its letters. When modeling these processes, it is convenient to create submodels for each stage, and to model the entire process as a composition of these atomic parts. By factoring a complex model into a combination of simpler ones, composition allows a combinatorial reduction in the number of models that need to be learned from observations.

Model composition raises two crucial but orthogonal issues. The first is *specification of legal submodel sequences*. In this paper, we consider the use of stochastic finite-state automata (SFSAs) to describe a probability distribution over possible submodel sequences. This is a fairly standard choice in areas such as speech and handwriting recognition, although some systems also use stochastic context-free grammars (SCFGs) for representing the higher-level structure of whole sentences. The second issue is *submodel representation*. Here we describe the use of dynamic probabilistic networks (DPNs) [5] (also called factorial HMMs [7]), which decompose the state of the process being modeled into a set of state variables. DPNs include both HMMs and Kalman filters as special cases [12, 11].

The use of DPNs for submodels confers several advantages over HMMs, which we describe below. However, composing DPN submodels is not quite as simple as composing HMMs. Given an SFSA with HMM submodels, the HMMs can simply be strung together according to the SFSA structure and formed into a

*Research supported by ARO MURI DAAH04-96-1-0341 and ONR N00014-97-1-0941.
†Research supported by ARO MURI DAAH04-96-1-0341 and ONR N00014-97-1-0941.

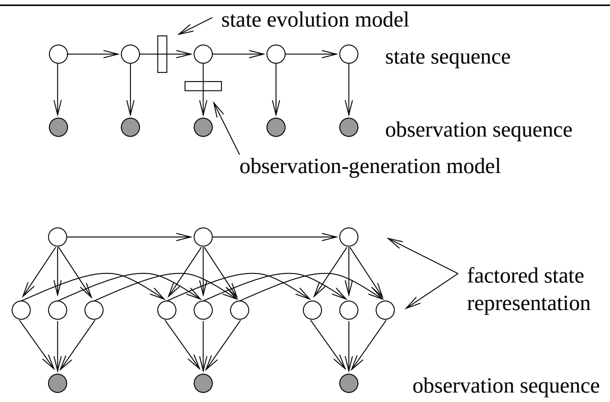


Figure 1: Top: A simple DPN, "unrolled" to show five time steps. Bottom: A DPN with a factored state representation. The factored representation can describe the evolution of an equal number of total states with exponentially fewer parameters. Variables whose values are known are shaded.

larger HMM, which can then be trained on observations and used for inference. The contribution of this paper is to show how to achieve the analogous result with DPNs. We present a method that takes a set of DPN submodels and an SFSA that defines a legal set of model concatenations, and constructs a composite DPN that represents correctly the intended probability distribution over possible histories of the temporal process. Our approach has the following advantages:

- Statistical efficiency. DPNs are factored representations of a probability distribution, and often have exponentially fewer parameters than unfactored representations such as standard HMMs. Hence these parameters can be estimated more accurately with a fixed amount of data [13, 7].
- There are efficient, general-purpose algorithms for doing inference [3, 12] and learning [2, 12] in DPNs. No special-purpose algorithms need be derived for handling representational extensions to HMMs that are required in areas such as speech recognition (see, e.g., [6]).
- Sharing variables between submodels leads to a natural way of describing transitional behavior. This is important for modeling coarticulation in speech recognition where the pronunciation of a phonetic unit depends on the positions of the tongue, jaw, and other articulators at the end of the preceding phonetic unit.

This paper examines only one aspect of what promises to be a fertile research area: the use of graphical models to specify complex hypotheses concerning causal structures in areas of scientific or commercial interest. Many other issues in temporal modeling with DPNs, especially the modeling of processes over multiple time scales, are discussed in [1]. The work of [12] also emphasizes the modeling power of DPNs in the context of speech recognition, but does not deal with submodel concatenation.

2 Probabilistic Networks and Dynamic Probabilistic Networks

In recent years, probabilistic or Bayesian networks [9] have emerged as the primary method for representing and manipulating probabilistic information in the AI community. These networks can be used to represent either static events, such as the co-occurrence of a set of diseases and symptoms, or to represent temporal processes such as the motion of an automobile in traffic.

A probabilistic network represents the joint probability distribution of a set of random variables $\{X_1, \ldots, X_n\}$. Denoting the assignment of a specific value to a variable by a lower-case letter, the probability of a joint assignment of values is specified with the chain rule of probabilities and a set of conditional independence assumptions as: $P(x_1, \ldots, x_n) = \prod_i P(x_i|Parents(X_i))$. Here $Parents(X_i)$ refers to a subset of the variables $X_1 \ldots X_{i-1}$; given values for its parents, X_i is assumed to be conditionally independent of all other lower-indexed variables. The conditional probabilities associated with each variable are often stored in tables referred to as CPTs. A probabilistic network has a convenient graphical representation in which the variables appear as nodes, and a variable's parents are specified by the arcs leading into it.

In the dynamic case, a probabilistic network models a system as it evolves over time [5]. At each point in time, the values $X_1, \ldots, X_n$ are of interest. For example, to model car-driving, we are interested in lane-position and speed at each point in time. A DPN uses a set of variables X_i^t to represent the value of the ith quantity at time t. DPNs are typically time-invariant so that the topology of the network is a repeating structure, and the CPTs do not change with time. The joint probability distribution is then represented as $\prod_{i,t} P(x_i^t|Parents(X_i^t))$. In networks with the Markov property, the parents of a variable in timeslice t must occur in either slice t or $t-1$. The conditional distributions within and between slices are repeated for all $t > 0$, so that DPNs can be specified simply by giving two slices and the links between them. When applied to an observation sequence of a given length, the DPN is notionally "unrolled" to produce a probabilistic network of the appropriate size to accommodate the observations.

The top of Figure 1 illustrates a generic DPN unrolled to show five time steps. The variables are divided into state variables, whose value is unknown, and observation variables, whose value is known. The state variables evolve in time according to a model encoded in their CPTs, which we refer to as the state evolution model. The state variables are related to the observation variables by the CPTs of the observation nodes. The bottom of Figure 1 shows a more realistic DPN in which the hidden state has been factored. We refer to a specific set of observations as $\mathbf{o}$; similarly, a specific assignment of values to the hidden state variables is denoted by $\mathbf{s}$.

Recent research has developed a set of algorithms for computing with probabilistic networks; [9, 2, 8] provide a good introduction. Specifically, there are dynamic programming procedures for

1. Computing the probability of an observation sequence: $P(\mathbf{o}) = \sum_{\mathbf{s}} P(\mathbf{o}, \mathbf{s})$.
2. Computing the most likely assignment of values to the hidden variables: $\max_{\mathbf{s}} P(\mathbf{o}, \mathbf{s})$.
3. Using a dataset to estimate the model parameters with the expectation-maximization (EM) procedure.

3 Network Structures for Model Composition

3.1 Why Model Composition with DPNs is Difficult

The difficulty in concatenating DPN models is best illustrated with an example. Suppose the word "no" is uttered, and there are separate models for the phonemes 'n' and 'o'. Figure 2 shows the simplest possible such submodels. There is a hidden state variable representing articulator positions, and a variable representing the sound observed at each point in time. The state CPTs specify articulator dynamics, and the observation CPTs link the articulator positions to the sounds made. Each submodel can be duplicated for an arbitrary number of timesteps.

Now consider constructing a composite model for a specific fixed-length utterance of the word "no." This is shown at the bottom of Figure 2. A naive concatenation of the two models would have to explicitly

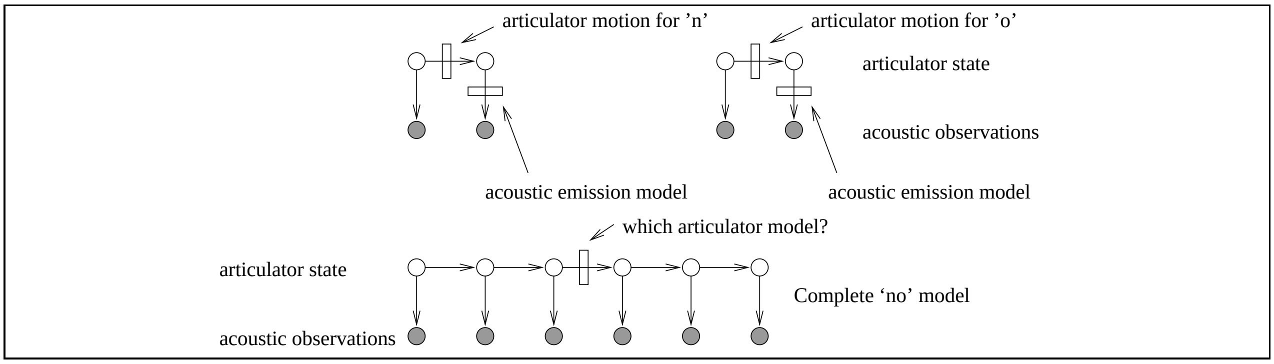


Figure 2: Concatenating submodels. Naive submodel concatenation requires specifying which state-evolution model to use at each point in time.

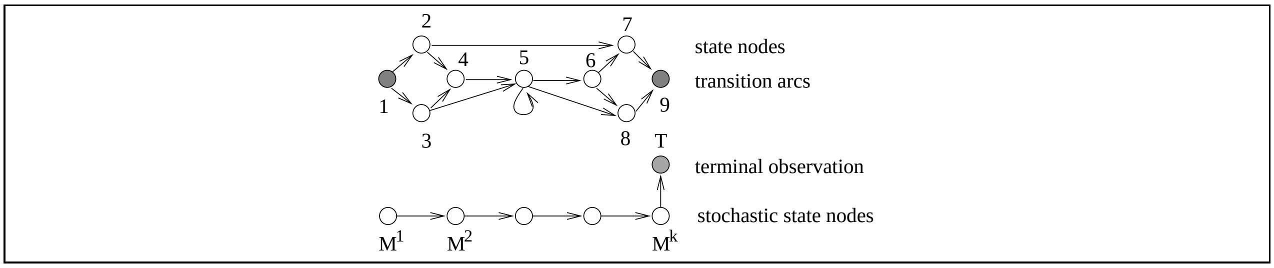


Figure 3: An SFSA and a DPN network representation for fixed-length observation sequences. Note that in the automaton the arcs represent transition probabilities while in the probabilistic network they represent conditional independence relations. The initial and final states of the SFSA are shaded. The shaded node in the DPN represents an artificial observation; the CPT of this variable will encode the length of the observation sequence.

partition the model into a fixed-length 'n' prefix followed by a fixed-length 'o' suffix. In practice, however, such segmentations are unavailable. Therefore, when doing inference or learning, all reasonable partitionings of an observation sequence between the models must be considered. Since the number of partitionings grows exponentially with the number of submodels, this is a demanding task that must be solved in an efficient way.

In the following sections we show how to construct a DPN that represents all the partitionings allowed by an arbitrary SFSA. The key problem is that the CPTs must be uniform across time, and the solution is to introduce an extra "index" variable in each timeslice on which to condition the state-evolution CPT. This creates a uniform structure that can generate the appropriate switching behaviors between models.

3.2 Specifying Legal Model Compositions

One way of specifying a partitioning between models is with an SFSA, where each path through the automaton specifies an acceptable partitioning. In this section, we show how to structure a DPN so that the inference and learning processes implicitly sum over all the partitionings allowed by an arbitrary SFSA; we then show how this enables model composition. We focus on fixed-length observation sequences because the training examples for most problems consist of fixed-length sequences. Similar results can be shown for infinite-length sequences.

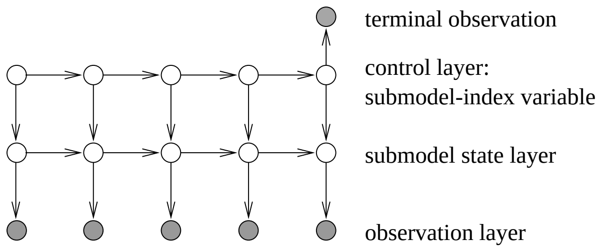


Figure 4: A DPN structured for model composition. The submodel-index variable specifies which submodel to use at each point in time.

We begin by constructing a trivial translation from SFSAs to degenerate DPNs. Consider the SFSA shown at the top of Figure 3. The nodes in this diagram represent states, and there are transition probabilities associated with the arcs. The initial and final states are shaded. The probability of a length k path $s_1 s_2 \ldots s_k$ through the automaton is given by $P_{s_2 s_1} P_{s_3 s_2} \ldots P_{s_k s_{k-1}}$ where $P_{s_i s_j}$ is the probability of a transition from state s_j to state s_i. The simple chain in the bottom of Figure 3 is the most straightforward way to represent paths of length k through this structure with a DPN. Each node represents a variable whose value is the state of the SFSA at a specific time. Each DPN variable $M^1 \ldots M^k$ has a distinct value for each state in the automaton, and there is a one-to-one mapping between paths of length k through the automaton and assignments of values to the variables in the DPN. So, for example, the path $1, 3, 5, 8, 9$ through the SFSA corresponds to the assignments $M^1 = 1, M^2 = 3, M^3 = 5, M^4 = 8, M^5 = 9$. The shaded DPN node represents a binary-valued variable whose value is always observed to be 1. The CPT of this "terminal observation" will encode the fact that the observation sequence is k steps long.

The transition probabilities associated with the arcs in the automaton are reflected in the CPTs associated with the variables in the DPN. If the probability of transitioning from state i to state j in the automaton is $P_{s_i s_j}$, then $P(M^{t+1} = j | M^t = i) = P_{s_i s_j}$ in the CPT associated with M^{t+1}.

All paths through the SFSA must start in the specified initial state, and end in the single final state. Suppose the SFSA states are numbered $1 \ldots n$, with 1 corresponding to the initial state and n to the final state. By setting the prior distribution on M^1 to $P(M^1 = 1) = 1$, the constraint on initial states is satisfied. By setting the conditional probability on the terminal observation T to $P(T = 1 | M^k = n) = 1$, and $P(T = 1 | M^k \neq n) = 0$, we ensure that any assignments of values to the variables which do not terminate in the final state are assigned a probability of 0. We summarize with the following proposition (proof omitted).

Proposition 1. Every assignment of values to the nodes in the DPN either 1) corresponds to a legal path through the SFSA and is assigned a probability equal to the probability of the path in the SFSA, or 2) corresponds to an illegal path in the SFSA and is assigned a probability of 0.

3.3 Model Composition

Figure 4 illustrates the way in which model composition is achieved. The submodel-index variable specifies which submodel to use at each point in time. The constraints on legal sequences of submodels are encoded in the CPTs of this layer, as described above. The submodel state layer represents the hidden variables in the DPN submodels.[1] By conditioning the submodel state variables on the submodel-index variable in the control layer, the desired switching behavior between models is achieved.

[1] For convenience, we assume that each submodel has the same state variables, but that their behavior differs. This is reasonable when there is an underlying physical system such as the vocal tract which behaves differently in different phonetic contexts, but which nevertheless has the same parts. The assumption can be relaxed easily [1].

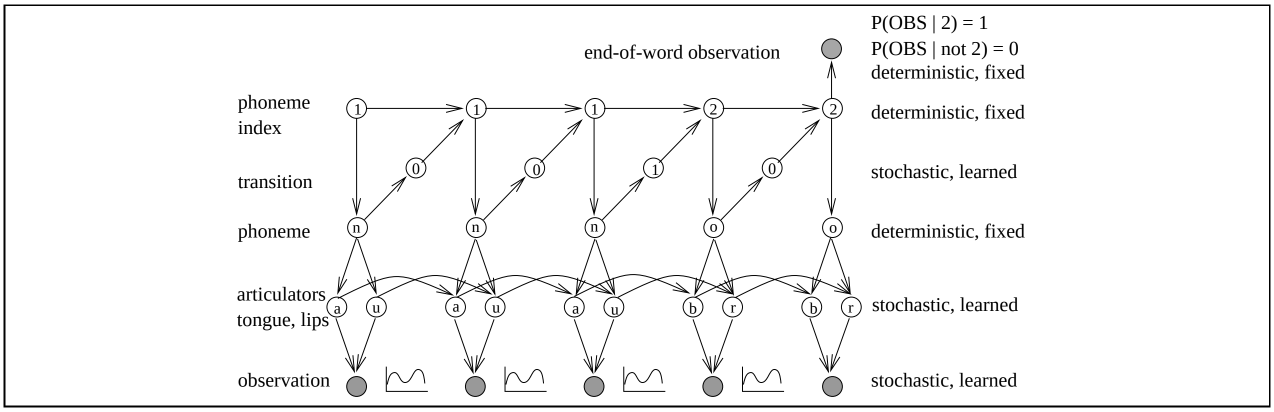


Figure 5: A DPN structured for speech recognition. The unshaded variables are hidden, and the labels attached represent just one acceptable assignment of values to them. The tongue moves from the alveolar ridge to the back of the mouth; the lips move from an unrounded to a rounded configuration. The properties of each node are shown to the right. Nodes whose CPTs are fixed are fixed on a per-example basis.

This result can be stated somewhat more formally as follows. Let $\mathbf{y}^t$ denote an assignment of values to the observation and submodel state variables at time t, and let m^t denote an assignment of a value to the submodel index variable at time t. The combination of a SFSA together with a set of DPN submodels specifies a probability distribution over sequences of $\mathbf{y}$ values. The probability of a particular sequence of $\mathbf{y}$ values is given by the weighted sum over all possible submodel sequences of the probability that each submodel sequence generates the given sequence of $\mathbf{y}$ values:

$$P(\mathbf{y}^1 \dots \mathbf{y}^k) = \sum_{m_1 \dots m_k} P(m^1 \dots m^k) P(\mathbf{y}^1 \dots \mathbf{y}^k | m^1 \dots m^k)$$

By the Markov property of the SFSA and the DPN submodels, we then have

$$P(\mathbf{y}^1 \dots \mathbf{y}^k) = \sum_{m_1 \dots m_k} P(m^1 \dots m^k) P(\mathbf{y}^1 | m^1) P(\mathbf{y}^2 | \mathbf{y}^1, m^2) \cdots P(\mathbf{y}^k | \mathbf{y}^{k-1}, m^k)$$

The first factor in each term corresponds to the probability of a particular path through the SFSA and is determined by the CPTs of the control layer; the remaining factors correspond to the probability of the specified behavior of the submodel variables, and is determined by the CPTs of the submodel state and observation layers. From the semantics of probabilistic networks given in Section 2, applied to the composite DPN model proposed above, we then have the following:

Proposition 2. The probability distribution over assignments to the observation and submodel state variables $\mathbf{y}^1 \dots \mathbf{y}^k$ that is represented by the composite DPN is identical to that represented by the input SFSA and DPN submodels.

This proposition sanctions the use of generic DPN inference and learning algorithms, together with the composite DPN structure, to handle models specified by any SFSA sequence model and DPN submodels.

4 A Speech Recognition Example

In this section we illustrate the process of model composition with the example of speech recognition. In this case, the training data consists of a set of utterances $\{U_1, U_2 .. U_n\}$, each with an associated phonetic transcription: $\{\langle p_{11}, p_{12}..p_{1q}\rangle, \langle p_{21}, p_{22}..p_{2r}\rangle..\langle p_{n1}, p_{n2}..p_{ns}\rangle\}$. The goal is to learn a probabilistic model for each phoneme, and then to construct word and sentence-level models through composition. Note that the SFSA corresponding to an utterance is particularly simple in this case: it is a linear structure proceeding in the appropriate order though the phonemes of the utterance.

Figure 5 illustrates a DPN that is structured for model composition in speech recognition. For clarity, we explicitly distinguish between CPTs that encode deterministic relationships and those which encode stochastic relationships. Proper handling of the deterministic variables also leads to computational efficiency. The phoneme-index nodes specify the index of the phoneme being uttered, i.e. the first, second, etc. For a given utterance, there is a deterministic mapping from the phoneme index to the actual phoneme. For example, in the word "no," the first phoneme is 'n' and the second is 'o'. This mapping is encoded in the CPT of the phoneme node, and is adjusted for each training utterance or recognition hypothesis. The terminal observation ensures that the phonetic transcription is respected. Since the number of phonemes varies from utterance to utterance, the terminal observation's CPT is also adjusted on a per-utterance basis. During the training phase, the number of phonemes is given for each utterance, and during the recognition phase the network computes the probability of different hypotheses, each of which also has a specific number of phonemes. The transition nodes are stochastic, and here depend on the phoneme being uttered. The acoustic observations depend on the articulator variables, which in this simple model represent the tongue and lips.

In linguistic terms, the CPTs of the transition variables represent the expected duration of different phonetic units. The CPTs associated with the articulator variables describe both linguistic knowledge about the target positions of the articulators for the various phonetic units, and additionally the basic physics of the vocal apparatus; these CPTs explicitly model the way in which the constraints imposed by this physical model (e.g. inertia) modulate the target positions. The CPTs associated with the observation variables describe the sounds generated by particular physical configurations of the vocal apparatus. The precise topology and initial parameter estimates for these connections embody in a very explicit way a phonological theory. Obviously, the better the initial estimates, the easier it will be to learn.

DPN models can express a number of speech phenomena in a natural way—for example, the constancy of accent and gender during speech, and in particular coarticulatory effects. For example, the pronunciation of the "s" in "nose" differs slightly from that in "cheese" because the preceding vowel leaves the tongue and lips in different states. This is automatically handled by the DPN model—the concatenation includes the "inertia" model for tongue and lips, hence the distribution over articulator states during the "s" will reflect the preceding vowel. In contrast, the handling of coarticulatory effects in the HMM context requires the use of diphones or triphones to represent combinations of phonemes. This results in a much larger set of phonetic states.

We have implemented a package for doing compositional modeling with DPNs, and are in the process of applying it to speech recognition. We have tested it with a multi-speaker database of digits utterances. Without an explicit articulatory model, we achieve a word error rate of approximately 5%, and we are in the process of training a detailed articulatory model with articulatory data obtained from real-time, low-power radar imaging of the vocal tract [4].

5 Discussion

Compositional DPN modeling seems to be a flexible and expressive way to represent and learn complex stochastic temporal models. In addition to speech, we are applying this technology to automobile control and human driver modeling. In our experiments, we have consistently found that whenever a factored state representation is an accurate reflection of the process being modeled, it confers great computational and

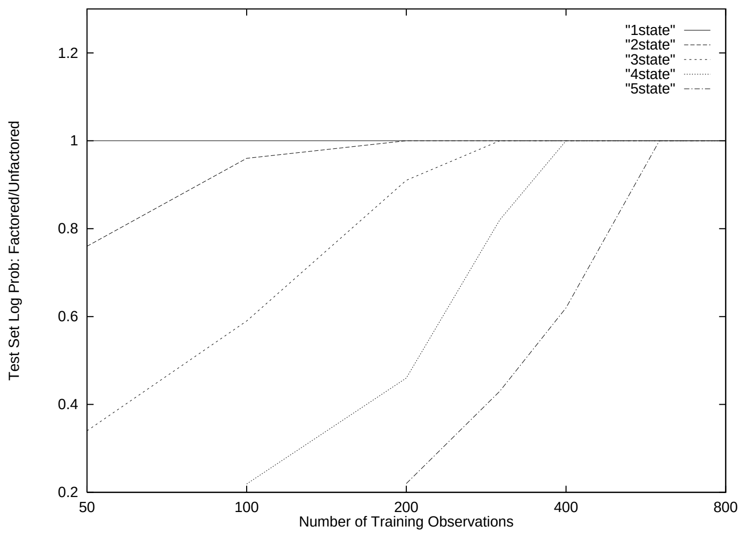


Figure 6: Comparison of learning ability for factored and unfactored representations. A ratio near 1 indicates that the unfactored representation is doing well. The unfactored representation has exponentially more state-evolution parameters, and requires exponentially more examples to achieve equal performance. With just one hidden state variable, the two representations are identical.

statistical advantages over a simpler unfactored representation, as suggested in [7, 2]. Figure 6 is illustrative. We generated data from a model with the topology shown at the bottom of Figure 1. We then learned the model parameters for both this factored model and an unfactored model with an equal amount of hidden state. The state variables in the factored representation were binary. The observation variable could take 2^P values where P is the number of its parents. We varied the number of hidden state variables in the factored representation from 1 to 5. For each number of hidden states, we generated data and test sets from 5 randomly initialized sets of CPTs. The figure of merit for each representation is the sum of the log-probabilities on the 5 test sets; Figure 6 shows the ratio of these numbers, and illustrates the expected behavior: as the number of state variables increases, there is an exponential increase in the number of training examples needed for the unfactored model to reach the same prediction performance. Virtually identical results also hold for computation time in the learning algorithm.

In summary, we have shown how to combine the advantages of a DPN's factored state representation with the flexibility of a SFSA in representing legal model compositions. This results in an expressive, concise, and efficient system for modeling complex stochastic processes.

References

[1] Constantin F. Aliferis and Gregory F. Cooper. A structurally and temporally extended Bayesian belief network model: Definitions, properties and modeling techniques. In *Proc. UAI-96*, 1996.

[2] J. Binder, D. Koller, S. Russell, K. Kanazawa. Adaptive Probabilistic Networks with Hidden Variables. To appear in *Machine Learning*.

[3] J. Binder, K. Murphy, S. Russell. Space-efficient inference in dynamic probabilistic networks.

[4] C. Burnett, J. Holzrichter, L. Ng, R. Leonard, and W. Lea. Micropower Radar Measurements of Human Vocal Articulator Motions. Lawrence Livermore National Laboratory report UCRL-JC-124821. 1996.

[5] T. Dean and K. Kanazawa. Probabilistic Temporal Reasoning. *Proceedings of the Seventh National Conference on Artificial Intelligence.* 524-528. 1988.

[6] J. Erler and G. Freeman. An HMM-based Speech Recognizer using Overlapping Articulatory Features. *Journal of the Acoustical Society of America.* **100**(1) 2500-2513. 1996.

[7] Z. Ghahramani and M. Jordan. Factorial Hidden Markov Models. *Machine Learning*, to appear.

[8] D. Heckerman. A Tutorial on Learning Bayesian Networks. *Microsoft Research*, MSR-TR-95-06. 1995.

[9] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann Publishers. 1988.

[10] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition.* Prentice-Hall Signal Processing Series. 1993.

[11] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[12] P. Smyth, D. Heckerman, and M. Jordan. Probabilistic independence networks for hidden Markov probability models. *Neural Computation* **9**(2), 227–269. 1997.

[13] G. Zweig. A Forward-Backward Algorithm for Inference in Bayesian Networks and An Empirical Comparison with HMMs. Master's Thesis, University of California at Berkeley. 1996.