The Helmholtz Machine Through Time

Geoffrey E. Hinton, Peter Dayan, Ava To and Radford Neal
Department of Computer Science, University of Toronto
Toronto, Ontario, M5S 1A4, Canada
{hinton, dayan, avato, radford}@cs.toronto.edu

Abstract

We describe the "wake-sleep" algorithm that allows a multilayer, unsupervised, stochastic neural network to build a hierarchical, top-down generative model of an ensemble of data vectors. Because the generative model uses distributed representations that are a non-linear function of the input, it is intractable to compute the posterior probability distribution over hidden representations given the generative model and the current data vector. It is therefore intractable to fit the generative model to data using standard techniques such as gradient descent or EM. Instead of computing the posterior distribution exactly, a "Helmholtz Machine" uses a separate set of bottom-up "recognition" connections to produce a compact approximation to the posterior distribution. The wake-sleep algorithm uses the top-down generative connections to provide training data for the bottom-up recognition connections and *vice versa*. In this paper, we show that the wake-sleep algorithm can be generalized to model the temporal structure in sequences of data vectors. This gives a very simple online algorithm that fits generative models which have distributed hidden representations which can be exponentially more powerful than conventional Hidden Markov Models.

1 Introduction

Neural networks are often used as bottom-up recognition devices that transform input vectors into representations of those vectors in one or more hidden layers. But multilayer networks of stochastic neurons can also be used as top-down generative models that transform random uncorrelated noise in the top hidden layer into highly structured patterns in the bottom, visible layer. In this paper we consider generative models composed of layers of stochastic binary units with top-down generative connections. Given a generative model parameterized by the top-down weights there is an obvious way to perform unsupervised learning. The generative weights are adjusted to maximize the probability that the visible vectors generated by the model would match the observed data. Unfortunately, to compute the derivatives of the log probability of a visible vector, $\mathbf{d}$, with

respect to the generative weights, θ, it is necessary to consider all possible ways in which $\mathbf{d}$ could be generated. For each possible binary representation α in the hidden units the derivative needs to be weighted by the posterior probability of α given $\mathbf{d}$ and θ:

$$P(\alpha|\mathbf{d},\theta) = \frac{P(\alpha|\theta)P(\mathbf{d}|\alpha,\theta)}{\sum_\beta P(\beta|\theta)P(\mathbf{d}|\beta,\theta)} \tag{1}$$

It is intractable to compute every $P(\alpha|\mathbf{d},\theta)$, so instead of minimizing $-\log P(\mathbf{d}|\theta)$, we minimize an easily computed upper bound on this quantity that depends on some additional parameters, ϕ:

$$F(\mathbf{d}|\theta,\phi) = -\sum_\alpha Q(\alpha|\mathbf{d},\phi)\log P(\alpha,\mathbf{d}|\theta) + \sum_\alpha Q(\alpha|\mathbf{d},\phi)\log Q(\alpha|\mathbf{d},\phi) \tag{2}$$

If we view $-\log P(\alpha,\mathbf{d}|\theta)$ as an energy, $F(\mathbf{d}|\theta,\phi)$ is a Helmholtz free energy and is equal to $-\log P(\mathbf{d}|\theta)$ when the distribution $Q(.|\mathbf{d},\phi)$ is the same as the posterior distribution $P(.|\mathbf{d},\theta)$. Otherwise, $F(\mathbf{d}|\theta,\phi)$ exceeds $-\log P(\mathbf{d}|\theta)$ by the asymmetric divergence:

$$\sum_\alpha Q(\alpha|\mathbf{d},\phi)\log\frac{Q(\alpha|\mathbf{d},\phi)}{P(\alpha|\mathbf{d},\theta)} \tag{3}$$

We restrict $Q(.|\mathbf{d},\phi)$ to be a product distribution within each layer that is conditional on the binary states in the layer below. We can then compute the distribution efficiently using a bottom-up recognition network (Dayan, Hinton, Neal and Zemel, 1995). The recognition weights, ϕ, take the binary activities in one layer and stochastically produce binary activities in the layer above using probabilities given by a logistic function. So for a given visible vector, the recognition weights may produce many different representations in the hidden layers, but we can get an unbiased sample from the distribution $Q(.|\mathbf{d},\phi)$ in a single bottom-up pass through the recognition net.

The highly restricted form of $Q(.|\mathbf{d},\phi)$ means that even if we use the optimal recognition weights, the gap between $F(\mathbf{d}|\theta,\phi)$ and $-\log P(\mathbf{d}|\theta)$ is large for some generative models. However, when $F(\mathbf{d}|\theta,\phi)$ is minimized with respect to the generative weights, these models can be avoided.

$F(\mathbf{d}|\theta,\phi)$ can be viewed as the expected number of bits required to communicate a visible vector to a receiver. First we use the recognition model to get a sample from the distribution $Q(.|\mathbf{d},\phi)$. Then, starting at the top layer, we communicate the activities in each layer using the top-down expectations generated from the already communicated activities in the layer above. Using an argument described in Hinton and Zemel (1994), it can be shown that the effective number of bits required for communicating the state of each binary unit is:

$$s_k \log \frac{q_k}{p_k} + (1 - s_k) \log \frac{1 - q_k}{1 - p_k} \tag{4}$$

where p_k is the top-down probability that s_k is on and q_k is the bottom-up probability that s_k is on.

There is a very simple online algorithm that minimizes $F(\mathbf{d}|\theta, \phi)$ with respect to the generative weights. We simply use the recognition network to generate a sample from the distribution $Q(.|\mathbf{d}, \phi)$ and then we adjust each top-down weight using the delta rule:

$$\Delta\theta_{kj} = \epsilon s_k (s_j - p_j) \tag{5}$$

where θ_{kj} connects unit k to unit j and ϵ is a learning rate.

It is much more difficult to exactly follow the gradient of $F(\mathbf{d}|\theta, \phi)$ with respect to the recognition weights, but there is a simple approximate method called the "wake-sleep" algorithm (Hinton, Dayan, Frey and Neal, 1995). We generate a stochastic sample from the generative model and then we apply the delta rule to increase the log probability that the recognition weights would produce the correct activities in the layer above:

$$\Delta\phi_{ij} = \epsilon s_i (s_j - q_j) \tag{6}$$

2 The Helmholtz Machine Through Time

The major disadvantage of conventional Hidden Markov Models (HMM's) is the impoverished nature of their representation of states. A Markov model can only be in one state at a time, and it is only by being in different states that it can preserve information over time (by the Markov property). So if the states use local representations in which a single hidden unit is active, the number of units required is exponential in the number of bits of information that need to be held (Williams & Hinton, 1991). If the states correspond to distributed representations in the hidden units, the number of hidden units required can scale linearly with the amount of information held in a state, but unfortunately learning and inference still seem to involve the exponential sums of equation 1.

The HMTT is shown in figure 1. As in backprop through time (Rumelhart, Williams and Hinton, 1986), weights are shared across timesteps, so that the amount of hardware required is fixed. There are two directions of generative influence; top-down generation within a timestep, just as in the static HM, and sideways generation between timesteps, in which the units in one layer at one timestep affect units in the same layer at the subsequent timestep.[1] In the most straightforward implementation of the HMTT, these two influences are

[1] It would be equally easy to have generative connections across time such that units in one layer at one time influence units in lower or higher layers at subsequent times.

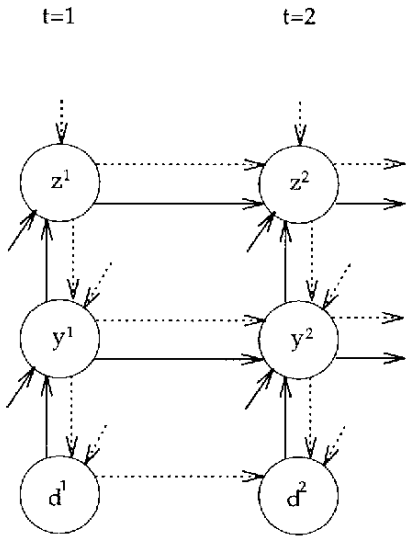

Figure 1: The Helmholtz machine Through Time. This shows the first 2 timesteps for a three-layer HMTT. Only one unit per layer is shown for convenience. Note that recognition (solid) and generative (dotted) connections both point in the direction of increasing time. More complicated architectures are also possible with direct generative connections from the input units at one time to the inputs at the next, or links that cross layers over time. The recognition and generative biases for the first timestep can be different from those for subsequent timesteps.

---

just added together before imposing the non-linearity. A complete generative sample from the network therefore consists of:

1. For the first time-step, generate activities $\mathbf{z}^1$, $\mathbf{y}^1$ for units starting at the top using only top-down generative weights;
2. For the second and subsequent time-steps, combine top-down activities in the current timestep with sideways activities from the previous timestep to determine the new generative probability for a unit, and sample from this generative probability to determine the new activities.

Since the first time-step lacks sideways influences, it is reasonable for it to use different generative biases (Frey, personal communication) from those used for subsequent timesteps.

If these are the only generative connections, then the overall generative model is clearly an HMM, where the output model is captured by the connections to the visible layer. Recognition in this context requires taking a sequence $\mathbf{d}^1, \mathbf{d}^2, \ldots$

of observed outputs and determining the posterior probabilities:

$$P(\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{z}^1, \mathbf{z}^2, \ldots | \mathbf{d}^1, \mathbf{d}^2, \ldots, \theta)$$

over the states of the units in the hidden layers. Note that the posterior probabilities are not causal, *ie* it is not necessarily true that $p(\mathbf{y}^1, \mathbf{z}^1 | \mathbf{d}^1, \mathbf{d}^2, \ldots, \theta)$ can be expressed as $p(\mathbf{y}^1, \mathbf{z}^1 | \mathbf{d}^1, \theta)$. Given their localist representations of states, standard HMMs can use the computationally efficient forward-backward algorithm (Baum and Eagon, 1967) to incorporate information from the future into state occupancy probabilities.

The HMTT uses a set of adjustable parameters ϕ which specify a simpler recognition model $Q(\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{z}^1, \mathbf{z}^2, \ldots | \mathbf{d}^1, \mathbf{d}^2, \ldots, \phi)$. In the simplest case this is a causal model, though it need not be so. As for generation, there are influences on recognition states from both the current and the previous timesteps, and these are additive. Recognition therefore proceeds as:

1. For the first time-step, sample activities for units bottom-up as in a standard wake phase recognition pass using the first observed data $\mathbf{d}^1$.
2. For the second and subsequent time-steps, combine bottom-up influences based on the observed data $\mathbf{d}^t$ with sideways influences from the previous hidden states $\mathbf{y}^{t-1}$ and $\mathbf{z}^{t-1}$ to determine the new recognition probabilities for the units. Sample from these to produce the states of the hidden units.

Although generative and recognition connections within a time-step are in opposed directions, recognition and generative connections from the previous timesteps are in the same direction – they both point forward. They will *not* in general have the same weights.

Not only is this making the standard assumptions of the static HM that the recognition model is conditionally factorial, but also, by being causal, it is ignoring errors in setting states that could be fixed by information from future observables. Consider the case in which there are two hidden states α and β that at time t are equally probable under the true posterior, given only the information $\mathbf{d}^1, \ldots \mathbf{d}^t$. At time t, the HMTT chooses just one of α and β with probability 0.5, and, unlike forward-backward based schemes, does not go back and revise its choice in the light of subsequent observations $\mathbf{d}^{t+1}, \ldots$. Causal or on-line recognition is computationally and neurally very attractive, however, since it is inconvenient to have repeatedly to backtrack and revise ones estimates. Limited temporal dependencies in the recognition and generative models can be incorporated through connections from further back in time than just the last timestep. Recognition connections from the observed data from further into the future can also be used to attempt to provide the recognition model with the non-casual information really required for correct inference. In the latter case, the state of the system at time t cannot be picked until time $t+r$, where r is the farthest future time that is allowed to influence the posterior distribution at the present.

The wake-sleep algorithm can be used to train the HMTT. As before, the wake phase consists of presenting complete sequences of observed inputs, picking

hidden states according to the recognition model, and training all the generative weights using the delta rule. The sleep phases consists of generating sequences of hidden and observable states from the generative model, and training the recognition model using the delta rule to be its inverse. The fact that there are both recognition and generative connections in the same direction between hidden units at adjacent timesteps causes no problems. All that is essential is that neither the recognition model nor the generative model contain directed cycles.

3 Experiment and results

3.1 The Learning Task

We modelled a time series of the activities of 9 binary visible units. These activities are generated by the interaction of 3 independent 3-state first-order Markov chains: a random walk process; a deterministic cyclic process; and an absorbing process. The transition structures are shown in figure 2a and were adapted from the factorial HMM used by Ghahramani & Jordan (1995). The entropies of the Markov chains over the length of the sequence of 100 patterns are approximately 87 nats, 0 nats, and 5 nats respectively (a nat is $\log_2 e$ bits). Within each group of three visible units, only one is on at a time depending on which state each Markov process is in. The initial observable pattern is 100100100.

3.2 Network Architecture

Given the factorial nature of the generative model for the time series we want to model, we adopt an HMTT that has a hidden layer of 3 soft-max groups of 3 units each[2] and a visible layer of 9 logistic units. Each unit within a softmax group has a probability of being on that is determined by:

$$p(y_j = 1) = \frac{e^{x_j}}{\sum_k e^{x_k}} \tag{7}$$

where k sums over units in the same softmax group as j and x_j is the total input (in either the generative or the recognition phase) for unit j. Only one unit within a soft-max group can be on at any one time.

Within a time-step, the network has 81 top-down generative weights and 81 bottom-up recognition weights. It also has 81 generative and 81 recognition weights from all 9 units in the hidden layer at time t to the same 9 units at time $t+1$. In addition, there are 18 generative biases and 9 recognition biases. On the very first time step, the 9 generative and 9 recognition biases of the

[2] Sigmoid hidden units do not perform as well.

hidden units have different values to compensate for the fact that there are no inputs across time from the previous states of the hidden units.

The point of the experiment is to see if the HMTT can learn to to use the 3 softmax groups to model the 3 Markov chains that produced the data.

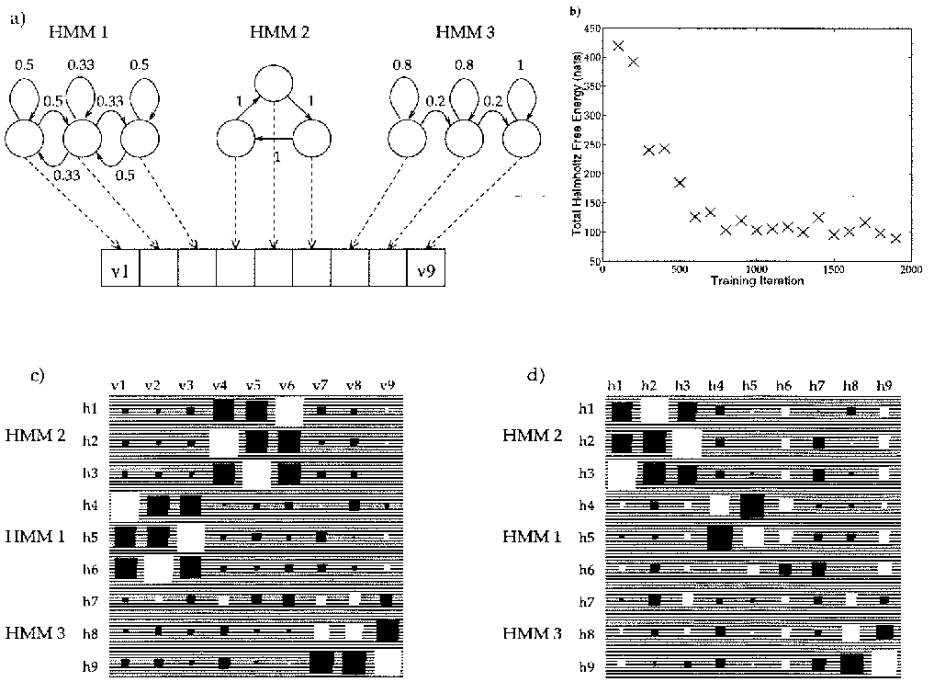

Figure 2: a) The generative model for the time series we modeled. The entropies of the Markov chains over a length of 100 are 87, 0 and 5 nats respectively. b) A plot of the total Helmholtz Free Energy (HFE) as a function of training iterations. At the end of the training, the HFE approaches the theoretical lower bound. The hidden-visible weights in c) show that the HMTT has learned to allocate one softmax group to each of the three different chains. d) shows the generative weights between the hidden units at consecutive times.

3.3 Results

Training sequences consisting of 100 patterns were generated according to the model described in section 3.1. The HMTT was trained using the wake-sleep algorithm with the learning rate for both recognition and generative connections set to 0.001. The performance was judged by the estimate of the total Helmholtz Free Energy (HFE) in Eq 2. Figure 2(b) shows how the estimated HFE changed as the network learned. It decreased rapidly at first, and after

800 training iterations it was close to the theoretical lower bound which is the entropy of the generating Markov process.

Figures 2(c) and 2(d) show all the generative weights learned by the HMTT. The hidden-visible weights in 2(c) show that the soft-max groups successfully discovered the factorial structure of the generative model. Figure2(d) shows the generative weights between hidden units that model the transition structure of the chains. Although the generative biases are not shown, it is clear that the first two chains – the random walk and the deterministic cycle – are learned almost perfectly. The absorbing process is learned less well – hidden unit $h8$ incorrectly generates 1s in both $v7$ and $v8$.

Acknowledgements

This research was funded by the Institute for Robotics and Intelligent Systems, the Information Technology Research Center, and NSERC. Hinton is the Noranda fellow of the Canadian Institute for Advanced Research.

References

Baum, L. E. and Eagon, J. A. (1967). An inequality with applications to statistical estimation for probabilistic functions of Markov processes and to a model for ecology. *Bulletin of the American Mathematical Society*, **73**, pp 360-363.

Dayan, P., Hinton, G. E., Neal, R., and Zemel, R. S. (1995) Helmholtz Machines. *Neural Computation, to appear.*

Ghahramani, Z. and Jordan, M.I. (1995). Factorial hidden Markov models. Computational Cognitive Science Technical Report 9502, MIT.

Hinton, G. E., Dayan, P., Frey, B. J. and Neal, R. (1995) The wake-sleep algorithm for self-organizing neural networks. *Science*, **268**, pp 1158-1161.

Hinton, G. E. and Zemel, R. S. (1994) Autoencoders, Minimum Description Length, and Helmholtz Free Energy. *Advances in Neural Information Processing Systems 6.* J. D. Cowan, G. Tesauro and J. Alspector (Eds.), Morgan Kaufmann: San Mateo, CA.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986) Learning representations by back-propagating errors. *Nature*, **323**, 533–536.

Williams, C. K. I. and Hinton, G. E. (1991) Mean field networks that learn to discriminate temporally distorted strings. In Touretzky, D. S., Elman, J. L., Sejnowski, T. J. and Hinton, G. E. (Eds.) Connectionist Models: Proceedings of the 1990 Connectionist Summer School. Morgan Kauffman: San Mateo, CA.