Varieties of Helmholtz Machine

Peter Dayan
Department of Brain and Cognitive Sciences
E25-201, MIT
Cambridge,
Massachusetts 02139
US
617 252 1693
fax: 617 253 2964
dayan@psyche.mit.edu

Geoffrey E Hinton
Department of Computer Science
6 King's College Road
Toronto
Ontario M5S 1A4
Canada
416 978 3707
fax: 416 978 1455
hinton@cs.toronto.edu

Running head: *Varieties of Helmholtz Machine*
Keywords: Expectation-Maximisation; Unsupervised Learning; Feedback Connections

Abstract

The Helmholtz machine is a new unsupervised learning architecture that uses top-down connections to build probability density models of input and and bottom up connections to build inverses to those models. The wake-sleep learning algorithm for the machine involves just the purely local delta rule. This paper suggests a number of different varieties of Helmholtz machines, each with its own strengths and weaknesses, and relates them to cortical information processing.

1 Introduction

This special issue focuses on four important questions about neural computation: 1) How is the world represented in the firing of neurons? 2) What are the functional roles of bottom-up and top-down connections between cortical areas? 3) Does the brain use internal models of the external world? 4) What are the basic synaptic plasticity rules? We believe that answers to these questions are indicated by the new theory of unsupervised learning that we present here.

Supervised learning algorithms for neural networks usually have the clear and easily justified goal of minimizing some loss function (*eg* the expected squared error or misclassification rate) on test data that is drawn from the same distribution as the training data. For unsupervised learning algorithms the goals are often less clear and less justifiable. Those that have been proposed in the literature can be roughly divided into four broad classes:

- Revealing structure in the ensemble of input vectors. This can be done by assigning each input vector to a cluster or to a location in a low-dimensional map. It can also be done by redescribing the input vectors in terms of multiple hidden causes that are not mutually exclusive, or by directly extracting informationally rich relationships between different parts of the input (Becker & Hinton, 1992).

- Preprocessing the data to facilitate a subsequent stage of supervised or reinforcement learning. The preprocessing may reduce the dimensionality of the data (*eg* principal components analysis, PCA, or self-organising feature maps, Kohonen, 1982). It may also produce a representation in terms of approximately independent features, as in PCA or factorial codes (Barlow, 1989) which typically reduces interference and increases the convergence speed of subsequent supervised learning procedures. It may even allow the subsequent supervised stage to be linear (as in radial basis functions).

- Assigning a probability density to input vectors so as to maximize the likelihood of the observed data or the expected likelihood of more data from the same distribution. Density estimation allows novel input vectors to be detected by their low probability density. It also allows input vectors to be assigned to classes probabilistically if a separate density model is created for each class.

- Compressing the data, with or without loss, to facilitate storage or communication.

Obviously, these goals are not independent. For example, clustering the data using a mixture of Gaussians model also assigns a probability density to each input vector. Using a hierarchical coding scheme to compress input vectors can be a good way of discovering true hidden causes because, in the large data limit, the true way in which the data was generated is almost certainly the most efficient way to code it. Discovering the true hidden causes of the input vectors is an excellent way to facilitate subsequent supervised learning because desired responses are almost always more simply related to the underlying causes of the sensory data than to the raw sensory data itself. Schemes for losslessly compressing input vectors can be used as probability density estimators because the number of bits required to code an input vector losslessly can always be viewed as an upper bound on its negative log probability.

Helmholtz machines, the new model of unsupervised learning which we present here, were originally motivated by the idea that hierarchical compression schemes would reveal the true hidden causes of the sensory data and that this would greatly facilitate subsequent supervised learning. An obvious practical advantage of this two stage approach is that it is typically easier to get the unlabelled data needed for unsupervised learning than the labelled data required for supervised learning, so the first stage can extract useful information from data that is useless for pure supervised learning. It is interesting, but far less obvious, that under very reasonable assumptions, the two stage approach can be superior even if all of the data is labelled. The minimum description length framework

(MDL, Rissanen, 1989) can be used to clarify this point. For a neural network to be of any value at all, the information required to specify the parameters of the neural network must be less than the information saved when communicating the data using the network. When MDL is applied to supervised learning, it is the labels that need to be communicated, so the information that each training case provides about the parameters is at most the number of predictable bits in the label. In unsupervised learning it is the input vectors that must be communicated so the useful information is at most the redundancy in the input vector which is typically much greater. Unsupervised learning therefore extracts far more information per training case. Of course, this information may be irrelevant to the labels. In the worst case, a complicated generative process involving hidden states could be used to model the input vectors, but the label associated with each input vector could have nothing to do with the hidden states of the generative model. Luckily, the world is not like this. It is almost always true that the desired responses to raw sensory data have a simpler relationship to the way the data was generated than to the raw data itself.

The next section develops the theory behind the two existing varieties of Helmholtz machine; section 3 discusses alternative versions of the machine that we have explored, changing the internal architecture, the unit activation functions, and the learning rules; section 4 draws together the answers to the four questions about the cortex that are suggested by these Helmholtz machines.

2 Density estimation with hidden states

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. However, networks can also be used as top-down generative models that produce vectors. Learning in such networks consists of adjusting parameters so as to maximise the log-likelihood of observed data vectors d:

$$\log p(d|\theta) = \log \left[\sum_\alpha p(d, \alpha|\theta) \right] \tag{1}$$

where $\alpha \in \mathcal{A}$ are the *hidden states* that can underly datapoint d. Datapoints are treated as being independent and therefore (adopting a non-Bayesian perspective) learning the parameters amounts to maximum likelihood estimation, finding

$$\theta^* = \text{argmax} \sum_d \log p(d|\theta).$$

or at least local maxima of this. For the case of mixtures of Gaussians, α indexes the different members in the mixture, $p(d, \alpha|\theta) = p(\alpha|\theta)p(d|\alpha, \theta)$ where $p(\alpha|\theta)$ are the mixing proportions, and $p(d|\alpha, \theta)$ is the probability density function associated with Gaussian α. For the case of binary Boltzmann machines (Hinton & Sejnowski, 1986) or binary Bayesian networks (Pearl, 1988) for density estimation or unsupervised learning, the inputs d are clamped on some of the units, and α indexes the 2^h possible states of the h remaining units in the machine.

These last two cases point up the difficulty that Helmholtz machines are intended to address. Changing the parameters θ to increase the probability (or probability density) assigned to data d requires performing sums such as those in equation 1 over an exponentially large set of hidden states. This is intractable in many cases of interest, and so other methods are necessary. One can hope that many of the states α will have very low net probabilities $p(d, \alpha|\theta) \ll 1$ and use Markov chain Monte Carlo sampling methods to sample the higher probability states (as in the Boltzmann machine). Alternatively, one can use approximation methods such as mean field methods or the Helmholtz machine which make the task of optimising θ in the light of the data computationally tractable.

Tractable Cases

There are two important cases in which the hidden states are such that it is computationally tractable to calculate the sum in equation 1 exactly. First, if all the units are linear and are subject to Gaussian noise, the resulting Helmholtz machine performs a standard form of factor analysis (Everitt, 1984; Neal *et al*, in preparation).

The second case is mixtures of Gaussians, in which the hidden units are non-linear, but interact in a simple way. In the generative model for mixtures of Gaussians, $p(\alpha|\theta) = \pi_\alpha$ where

$$\sum_\alpha \pi_\alpha = 1 \tag{2}$$

and $p(d|\alpha, \theta) = \mathcal{N}[\mu_\alpha, \Sigma_\alpha]$. Tractability comes from equation 2, since it implies that only one of the Gaussians can be responsible for each d. Therefore the sum in equation 1 grows linearly rather than exponentially in the number of Gaussians. Although one could use gradient ascent to optimise $\theta = \{\pi_\alpha, \mu_\alpha, \Sigma_\alpha\}$, just as for factor analysis (Rubin & Thayer, 1982), there is a more efficient technique called the expectation maximisation (EM) algorithm (Baum *et al*, 1970; Dempster *et al*, 1977; Neal & Hinton, 1994; Amari, 1995) which is another of the bases of the Helmholtz machine.

EM makes use of an equivalent form of equation 1:

$$\begin{aligned}
\log p(d|\theta) &= \log\left[\sum_\alpha p(d, \alpha|\theta)\right] && (3)\\
&= \sum_\alpha \mathcal{P}_{\alpha|d} \log\left[p(d, \alpha|\theta)\right] - \sum_\alpha \mathcal{P}_{\alpha|d} \log\left[\mathcal{P}_{\alpha|d}\right] && (4)\\
&= \sum_\alpha \mathcal{Q}_{\alpha|d} \log\left[p(d, \alpha|\theta)\right] - \sum_\alpha \mathcal{Q}_{\alpha|d} \log\left[\mathcal{Q}_{\alpha|d}\right] + KL\left[\mathcal{Q}_{\cdot|d}, \mathcal{P}_{\cdot|d}\right] && (5)
\end{aligned}$$

where $\mathcal{P}_{\alpha|d} = p(\alpha|d,\theta)$ is the *posterior* probability of α given d, $\mathcal{Q}_{\alpha|d}$ is the probability accorded to hidden state α by an arbitrary probability distribution $\mathcal{Q}_{\cdot|d}$ that depends on d but need not depend on θ, and

$$KL\left[\mathcal{Q}_{\cdot|d}, \mathcal{P}_{\cdot|d}\right] = \sum_{\alpha} \mathcal{Q}_{\alpha|d} \log\left[\mathcal{Q}_{\alpha|d}/\mathcal{P}_{\alpha|d}\right]$$

is the Kullback-Leibler divergence from distribution $\mathcal{Q}_{\alpha|d}$ to distribution $\mathcal{P}_{\alpha|d}$. Define $\mathcal{F}(d;\theta,\mathcal{Q})$ as:

$$-\mathcal{F}(d;\theta,\mathcal{Q}) = \sum_{\alpha} \mathcal{Q}_{\alpha|d} \log\left[p(\alpha|\theta)p(d|\alpha,\theta)\right] - \sum \mathcal{Q}_{\alpha|d} \log\left[\mathcal{Q}_{\alpha|d}\right]. \tag{6}$$

It turns out (Neal & Hinton, 1994) that $\mathcal{F}(d;\theta,\mathcal{Q})$ is minimised in $\mathcal{Q}$ for fixed d and θ at $\mathcal{Q}_{\alpha|d} = \mathcal{P}_{\alpha|d}$, at which point $KL\left[\mathcal{Q}_{\alpha|d}, \mathcal{P}_{\alpha|d}\right] = 0$ and so $-\log p(d|\theta) = \mathcal{F}(d;\theta,\mathcal{P})$.[1] Neal & Hinton (1994) showed that EM consists of alternating minimisation of $\mathcal{F}(d;\theta,\mathcal{Q})$ for all the data d. During the E-step, $\mathcal{F}$ is minimised with respect to $\mathcal{Q}$, leading to $\mathcal{Q}_{\alpha|d} = \mathcal{P}_{\alpha|d} = p(\alpha|d,\theta)$; during the M-step, $\mathcal{F}$ is minimised with respect to θ assuming that $\mathcal{Q}$ is fixed. The log likelihood of the data is guaranteed not to decrease during each complete step. The advantages of EM over simple gradient methods are that θ can change by a large amount during each step, and that no learning rate is required. Jordan & Xu (1993) and Xu & Jordan (1995) show that for certain mixture models, EM adjusts the parameters in a direction that has a positive projection onto the derivative of the log likelihood with respect to those parameters, through a projection matrix that is a function of the parameters.

For mixtures of Gaussians, it is computationally simple to calculate the optimal $\mathcal{Q}_{\alpha|d} = \mathcal{P}_{\alpha|d}$, because the number of hidden states that must be considered is equal to the number of Gaussians. Given $\mathcal{Q}_{\alpha|d}$,

$$\sum_{\alpha} \mathcal{Q}_{\alpha|d} \log\left[p(\alpha|\theta)p(d|\alpha,\theta)\right] = \sum_{\alpha} \mathcal{Q}_{\alpha|d} \left\{\log \pi_{\alpha} - \frac{1}{2}(d-\mu_{\alpha})^{T}\Sigma_{\alpha}^{-1}(d-\mu_{\alpha}) - \frac{1}{2}\log|\Sigma_{\alpha}|\right\} + \mathcal{K}$$

where $\mathcal{K}$ is a constant. $\mathcal{Q}$ separates out the optimisation problems for each α making it simple to adjust the parameters θ.

Sampling Methods

Unfortunately, examples in which the sum in equation 1 can be performed exactly are rare. We are interested in cases with large numbers of hidden units where it is unreasonable for them all to be linear (which would nullify the point of having many layers in a hierarchy) or for only one or a few to fire in response to a single input. Other probability density models

[1] For later convenience, $\mathcal{F}$ is defined in terms of negative log probabilities, which is why it is minimised rather than maximised.

are appropriate here such as the Boltzmann machine or Bayesian networks, but optimising their parameters is difficult. The Boltzmann machine (BM) is a case in point. For this, there are n observable binary units, h hidden binary units, and probabilities are defined in terms of energies

$$p(d, \alpha|\theta) = \frac{e^{-E(d,\alpha)/T}}{\mathcal{Z}(T)} \tag{7}$$

where

$$E(d, \alpha) = -\sum_{i<j} w_{ij} s_i s_j - \sum_i b_i s_i$$

where s_i is the binary state of the ith units, T is a temperature parameter,

$$\mathcal{Z}(T) = \sum_{e,\beta} e^{-E(e,\beta)/T} \tag{8}$$

is called the partition function, and $\theta = \{w_{ij}, b_i\}$ is the collection of parameters.

Optimisation of the parameters θ in the Boltzmann machine can also be seen in terms of EM and equation 4. Since there are h hidden units, the sums are over 2^h possibilities. Worse, the sum in equation 8 to calculate the partition function is over 2^{n+h} possibilities. Hinton and Sejnowski (1986) therefore used a Markov chain Monte-Carlo sampling technique called Gibbs sampling (see Neal, 1993 for an excellent description of Gibbs sampling and other Markov chain methods) to approximate averages such as those in equation 4 over the posterior probabilities $p(\alpha|d, \theta)$. Gibbs sampling with clamped data d involves starting with all the hidden units in some state, and considering each hidden unit in turn to decide whether to change its state. The decision for unit i is stochastic and is based on the sigmoid of the net input $\sum_i w_{ij} s_j + b_i$ including contributions from the states of all the hidden and the clamped units. With d clamped, the Gibbs sampling Markov chain has $\mathcal{P}_{\cdot|d}$ as its equilibrium distribution, and so sample averages over states observed from the chain can be used to estimate equation 4 and its derivatives.

It turns out that optimising θ in the BM requires two applications of Gibbs sampling. During the "wake" phase, the observed data d are clamped on the n observable units, and Gibbs sampling on just the hidden units is used to estimate the average $\langle s_i s_j \rangle^+$ of the correlations between the activities of both hidden and observable units. During the "sleep" phase, no units are clamped, and Gibbs sampling over all the units is used to estimate the average $\langle s_i s_j \rangle^-$ of the correlations between the activities of all the units when they are 'running free'. Gradient ascent in the log likelihood then corresponds to:

$$\Delta w_{ij} \propto \langle s_i s_j \rangle^+ - \langle s_i s_j \rangle^- \tag{9}$$

There are two problems with the Boltzmann machine approach to density estimation in the face of hidden states. The first is the application of Gibbs sampling at all, since it is tricky to know when enough sampling has been done to get sufficiently close to thermal equilibrium.

The second is the subtraction of the two noisy terms in the update in equation 9. During learning, these quantities get to be close to each other, and the signal in the difference gets swamped by the noise, which adds across wake and sleep phases.

The sleep phase is required in the BM because equation 7 relates the probabilities of states in terms of a partition function which it is intractable to compute. Bayesian networks (*eg* Pearl, 1988) offer a hierarchical probability model that is unidirectional. In this case, there is an ordering of the units such that one can write:

$$p(\{s_i\}|\theta) = p(s_1|\theta)p(s_2|s_1, \theta) \ldots p(s_N|\{s_1, s_2, \ldots, s_{N-1}\}, \theta)$$

where N is the total number of units. In typical Bayesian networks, the number of parents that directly affect the conditional probability of a unit is fairly small.

Learning in Bayesian networks follows the pattern already established. Given observed data d, the posterior probability distribution $p(\alpha|d, \theta)$ over the hidden units is calculated, and gradient or analytical methods can be used to increase the log likelihood. In cases where units have very few parents, it is computationally tractable to calculate the posterior probabilities exactly, and also to represent the conditional probabilities $p(s_k|\{s_1, s_2, \ldots, s_{k-1}\}, \theta)$ explicitly in tables. This makes adaptation very quick. However, many interesting models, and particularly learned rather than intuited models, require more dependencies than can be accommodated in this way.

Neal (1992) suggests using parameterised conditional densities for $p(s_k|\{s_1, s_2, \ldots, s_{k-1}\}, \theta)$, specifically making this a sigmoid function of net input expressed through a set of variable weights. He also describes a method for doing estimated gradient ascent in the log likelihood using Gibbs sampling to estimate the posterior distribution. Unlike the update rule in equation 9, there is no need for the sleep phase of learning, since the partition function is 1, and Neal (1992) showed that this speeded learning.

Approximate Methods

Experience with Gibbs sampling suggests that existing sampling methods are unlikely to be adequately fast for complicated models, either for learning or for inferring hidden causes from data once learning is complete. There is therefore reason to search for approximate methods whose error of approximation can be bounded in some way. $\mathcal{F}(d; \theta, \mathcal{Q})$ defined in equation 6 is a natural place to turn, for two reasons. First, since it is always true that $KL\left[\mathcal{Q}_{\cdot|d}, \mathcal{P}_{\cdot|d}\right] \geq 0$, we know that

$$-\log p(d|\theta) \leq \mathcal{F}(d; \theta, \mathcal{Q}) \tag{10}$$

with equality if $\mathcal{Q}_{\cdot|d} = \mathcal{P}_{\cdot|d}$. Second, the sums over the exponentially many α in equation 6 involve a probability distribution $\mathcal{Q}_{\cdot|d}$ that can be determined to make minimisation of $\mathcal{F}$

tractable, instead of the distribution $\mathcal{P}_{\cdot|d}$ that in many cases requires an exponential sum simply to be calculated.

Mean field methods are the paradigm in these cases. For either the BM (Peterson & Anderson, 1987; Hinton, 1989) or Bayesian nets (Saul, Jaakkola & Jordan, 1995; Jaakkola, Saul & Jordan, 1995), mean field methods take $\mathcal{Q}_{\cdot|d}$ to be factorial, *ie* they take all the hidden units to be mutually independent given the data. Solving for the optimal factorial distribution for the hidden units typically requires solving a set of consistency equations and can be performed using a few iterations. Depending on the form of $p(d, \alpha|\theta)$ and the mean field distributions, one can then use gradient descent or exact methods to update the parameters θ to minimise $\mathcal{F}(d; \theta, \mathcal{Q})$ using the factorial form of $\mathcal{Q}_{\cdot|d}$ to make this tractable. The whole procedure amounts to minimising an upper bound $\mathcal{F}(d; \theta, \mathcal{Q})$ to the negative log likelihood using the same alternating method as EM (estimate the best $\mathcal{Q}$ as a function of θ and then improve θ using this $\mathcal{Q}$) except that a bound is being minimised rather than the real negative log likelihood, and there are restrictions on the form of $\mathcal{Q}$ (that it be factorial) which prevent the E-step from being complete (the calculated $\mathcal{Q}_{\cdot|d}$ may not be $\mathcal{P}_{\cdot|d}$).

The Helmholtz machine (Hinton *et al*, 1995; Dayan *et al*, 1995) is an instance of this technique, except that rather than use mean field methods and mean field iteration to get a convenient $\mathcal{Q}_{\cdot|d}$, it devotes a separate set of parameters ϕ to model $\mathcal{Q}_{\cdot|d}$ and optimises $\mathcal{F}(d; \theta, \phi)$ over these parameters too.

The Helmholtz Machine

Figure 1 shows an example of the Helmholtz machine. The top-down weights are the parameters θ of the *generative* model – a unidirectional Bayesian network in which units are divided into layers with complete connections from units in layer $\mathcal{Z}$ to those in layer $\mathcal{Y}$ and so forth (one could clearly use partial connectivity, or also connections that skip layers, provided that the directed connection graph remains unidirectional). The generative model is factorial within each layer – each unit in layer $\mathcal{Y}$ is independent of the others within the same layer given the activities in the layer $\mathcal{Z}$ and the weights. Note that optimising the parameters of this Bayesian network is intractable in the sense discussed above, since each unit has many parents.

The bottom-up weights are the parameters ϕ of the *recognition* model – another unidirectional Bayesian network based on the same architecture as the generative model. The recognition model is also factorial within each layer – but now the units in layer $\mathcal{Z}$ are mutually independent given the activities in layer $\mathcal{Y}$ rather than *vice-versa.* There is no reason to believe that the true posterior distribution $p(\alpha|d, \theta)$ is really factorial in this manner, although the way the network is trained provides it with strong encouragement to choose generative parameters θ for which forcing this approximation is not disastrous.

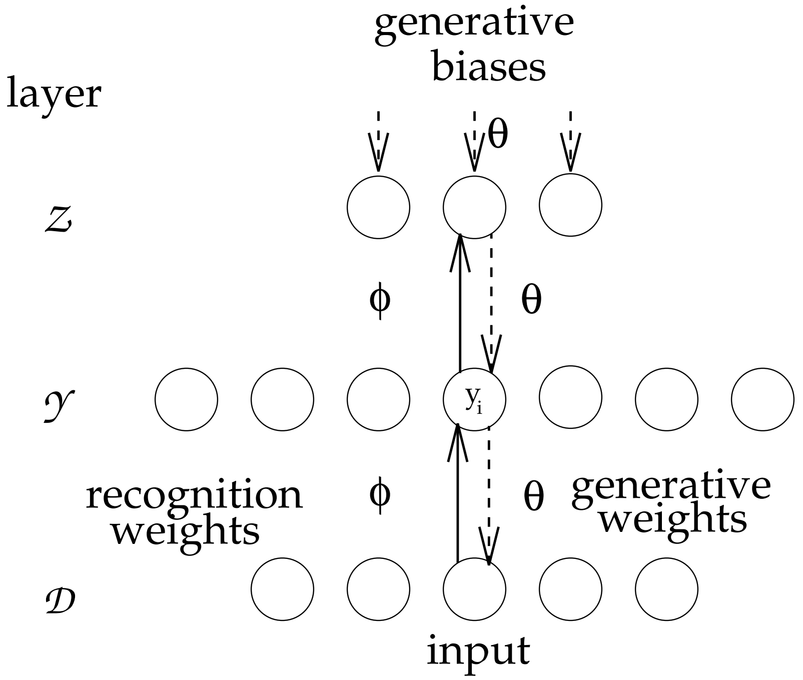

Figure 1: A three-layer Helmholtz machine. Weights θ define a top-down generative model in layers $\mathcal{Z}$, $\mathcal{Y}$ and $\mathcal{D}$ of stochastic binary units. $\mathcal{D}$ are the observables, and the task for θ is maximising the probability $p(\mathbf{d}|\theta)$. Weights ϕ define a bottom-up recognition model that attempts to invert the generative model.

An alternative way to look at the HM is in terms of autoencoders. A HM with just one hidden layer is exactly (a folded-over version of) the sort of autoencoder that Zemel (1994) and Hinton & Zemel (1994) treated. The recognition model performs the *coding* operation of turning inputs d into stochastic codes in the hidden layer; the generative model reconstructs its best guess of the input on the basis of the code that it sees. Maximising the likelihood of the data can be interpreted in terms of minimising the total number of bits (which is just $\mathcal{F}$) it takes to send the data from a sender to a receiver who knows the generative model but not the inputs themselves.

There are presently two main variants of the Helmholtz machine. Both use gradient methods to optimise $\mathcal{F}$, but they use different types of bottom-up recognition model to avoid the remaining exponential sums in (the derivatives of) equation 6.

The *deterministic* HM (Dayan *et al*, 1995) makes approximations inspired by mean-field methods, replacing stochastic firing probabilities in the recognition model by their deterministic mean values. The advantage of this is that one can use powerful optimisation methods to update the parameters θ, ϕ. However, there are two main disadvantages. One is that this replacement of random quantities by their means entails that the resulting recognition distribution $\mathcal{Q}_{\cdot|d}$ is trivial. Even though we are assuming that the hidden activities within a layer are conditionally independent, there are correlations between the activities in different hidden layers. These correlations are not captured in the deterministic HM.

The second disadvantage is that under the model we actually used, we did not have a guaranteed bound on the likelihood – the term $\sum_\alpha \mathcal{Q}_{\alpha|d} \log\left[p(\alpha, d|\theta)\right]$ in equation 6 was not calculated tractably in a way that ensured the bound would be preserved. More sophisticated methods (Saul *et al*, 1995; Jaakkola *et al*, 1995) could be used to repair this problem. Even with these disadvantages, though, we showed this machine working on a number of difficult unsupervised learning problems.

The *stochastic* HM (Hinton *et al*, 1995) captures the correlations between the activities in different hidden layers, but at the expense of requiring sampling. However, because the recognition model is unidirectional, an unbiased sample can be obtained in a single forward pass. The learning method for the stochastic HM is called the wake-sleep algorithm.[2] During the wake phase, samples are taken from the recognition model, and the parameters θ of the generative model are updated according to the sampled gradient of $\mathcal{F}$. The weight updates require nothing more than the completely local delta rule.

During the sleep phase, samples (fantasies) are generated top-down by the generative model, and the parameters ϕ of the recognition model are updated to make it better approximate the inverse of the generative model. The weight updates again require nothing more than the completely local delta rule. Unfortunately, as discussed in Hinton *et al* (1995), the recognition model is not being adjusted according to the gradient of $\mathcal{F}$ as desired, but rather according to the gradient of $KL\left[\mathcal{P}_{\cdot|d}, \mathcal{Q}_{\cdot|d}\right]$ where the data d are drawn according to the generative distribution. This is the Kullback-Leibler divergence from the true posteriors to those imputed by the recognition model rather than *vice-versa.* The Kullback-Leibler divergence is not a metric because it is not symmetric, and the asymmetry can be important. Nevertheless, the wake-sleep algorithm also worked well in practice, including on a task which required it to build hierarchical generative models of 8×8 binary images of handwritten digits and use the log likelihoods under each model to perform digit recognition.

The stochastic HM has two important advantages over the BM. First, no iteration is required to extract unbiased samples. Second, although there is random sampling error when computing the weight updates, the fact that the top-down generative model is a Bayesian net rather than a Markov random field implies that it is at least not necessary to learn based on the difference between two noisy samples (Neal, 1992).

Learning a model of the world and using samples from the model during an offline phase to learn to invert that model has its parallels in DYNA, Sutton's (1991) model-based reinforcement learning system. The equivalent of the recognition model in DYNA reports

[2] Note that wake and sleep are used in a somewhat different manner from the BM.

the long run values of states in a rewarded Markov decision problem (MDP). Just like the exponential sums in equation 1, it is intractable to calculate these values online even having a full model of the MDP (which is like knowing θ). DYNA uses sleep samples from the model and temporal difference learning (Sutton, 1988; Barto, Sutton & Watkins, 1989) to acquire the inverse.

3 Variants of the Helmholtz Machine

The main purpose of this paper is to describe a number of the variants of the Helmholtz machine (HM) which we have explored. We have not attempted to provide an exhaustive survey – there are also many other varieties. Few of the machines have been fully tested and some are only really intended to provide baselines for the performance of other, more sophisticated methods. Mean field methods that do not use separate sets of recognition parameters (Saul *et al*, 1995; Jaakkola *et al*, 1995) are also under investigation.

Our explorations can be summarized as follows:

unit activation functions The original HM involved networks of binary stochastic sigmoid, noisy-or (Pearl, 1988; Saund, 1995) or competitive (Dayan & Zemel, 1995) units. Softmax, linear, and essentially all other unit types can also be used, and different unit types can be intermixed in the same network. The wake-sleep algorithm is particularly convenient for this since it is not necessary to concoct different mean field bounds for each new activation function. The case with just one hidden layer of linear units and added Gaussian noise turns out to be equivalent to factor analysis.

reinforcement learning We have explored a reinforcement learning method (based on REINFORCE, Williams, 1992) which optimises both ϕ and θ during the wake phase (and therefore does without a sleep phase). Although it changes the recognition model correctly according to the sampled gradient of $\mathcal{F}$ with respect to ϕ, one can expect it to scale poorly with network size in terms of convergence time.

alternative recognition models As a more sophisticated version of the deterministic HM, we tried a form of backpropagation through time (Rumelhart, Hinton & Williams, 1986) to implement a recurrent recognition model. We also tried including "dangling" units that have no outgoing generative connections. Such units can be useful for recognition and are theoretically attractive for modeling 'explaining away' effects (Pearl, 1988). They can be trained using the conventional sleep-wake algorithm. Finally, we considered integrating the HM with the BM, using a strictly top-down generative model, as in the HM, but allowing recurrent connections within layers in the recognition model.

supervised HMs There are at least three different ways of using the Helmholtz machine for supervised learning. Each builds a different form of conditional probability model for the outputs given the inputs.

modeling temporal structure We have implemented a version of the HM designed for the kinds of temporally sensitive tasks to which HMMs are applied. The advantage of the HM over HMMs is that it allows hidden states to have distributed representations, and this can allow exponential savings (Williams & Hinton, 1991; Ghahramani & Jordan, 1995). However, the HM is more difficult to train, since wake-sleep lacks an equivalent of the forward-backward algorithm which permits future observations to affect the posterior distributions over present states.

3.1 Unit activation functions

In one of its abstract forms, a Helmholtz machine consists of two uni-directional Bayesian networks, one of which is the underlying generative model for data; the other of which is responsible for producing an estimate of the posterior probabilities of the states of the generative network given a datapoint. In most cases, there is no principled reason to use exactly the same kind of model in the top-down pathway as in the bottom-up pathway. Indeed, in the deterministic version of the machine (Dayan *et al*, 1995) that we used for solving the 8-bit shifter problem, we used a particular competitive imaging model (Saund, 1995; Dayan & Zemel, 1995) for the output units that was inspired by the Integrated Segmentation and Recognition architecture (Keeler *et al*, 1991).

We will describe some of the options in the case that unit i, whose activation is y_i, receives variable connections w_{ji} from units j whose stochastic activations are z_j. For the wake-sleep algorithm, one needs to specify the activation rule, namely how w_{ji} and z_j interact to generate a probability distribution over y_i (for use in terms such as $p(\alpha|\theta)$ and $p(d|\alpha,\theta)$), and the derivative of the $\log$ probability of y_i under this rule with respect to the variable parameters. The deterministic version is slightly more demanding. Typically z_j will be stochastic, and will therefore generate a variety of probability distributions over y_i depending on their values. One should really sum over these distributions, weighted according to the probabilities of the z_j that generated them. However, this is usually intractable, and one therefore needs some approximation to the net $\log$ probability of y_i and its derivatives. Arbitrary approximations can be made in the recognition distribution without harming the fact that $-\mathcal{F}(\theta,\phi)$ is a lower bound on the $\log$ likelihood (although they may have an adverse impact on its tightness). However, this is not true for using these activation functions for the generative distribution – preserving the bound restricts the allowable approximations. For most of the activation functions, there seems not to be an obvious best approximation.

We have generally used standard exponential family units (Rumelhart *et al*, 1995). Table 1 collects together the rules, and the derivatives of the $\log$ probabilities as a function of the weights. Using the exponential family ensures the simple form for the derivatives; the underlying factorial nature of the units in the networks ensures that only local information need be used in updating the weights.

3.2 The Reinforcement Learning HM

The reinforcement learning framework REINFORCE (Barto & Anandan, 1985; Williams, 1992) can be used to adapt the parameters of both recognition and generative models according to unbiased sample derivatives of $\mathcal{F}$. For an HM with two hidden layers, $\mathcal{Y}$ and $\mathcal{Z}$, one can write $\mathcal{F}$ as a sum over the exponential number of settings for the states of the hidden units ($\alpha = \{\mathbf{y}, \mathbf{z}\}$) weighted by the recognition probabilities of those units.

$$\begin{aligned} \mathcal{F}(\mathbf{d}; \theta, \phi) \quad = \quad -\sum_{\mathbf{y}} \mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi} \sum_{\mathbf{z}} \mathcal{Q}_{\mathbf{z}|\mathbf{y};\phi} \quad & \log\left[p(\mathbf{d}|\mathbf{y}, \theta) p(\mathbf{y}|\mathbf{z}, \theta) p(\mathbf{z}|\theta)\right] - \\ & \mathcal{H}\left[\mathcal{Q}_{\cdot|\mathbf{d};\phi}\right] - \mathcal{H}\left[\mathcal{Q}_{\cdot|\mathbf{y};\phi}\right] \end{aligned} \tag{11}$$

where $\mathcal{H}[\mathcal{Q}]$ is the entropy of distribution $\mathcal{Q}$ and we are explicitly using the conditional factorial form of $\mathcal{Q}$ to write $\mathcal{Q}_{\mathbf{y},\mathbf{z}|\mathbf{d};\phi} = \mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi}\mathcal{Q}_{\mathbf{z}|\mathbf{y};\phi}$. Gradient descent in $\mathcal{F}$ requires us to calculate the derivatives of $\mathcal{F}$ with respect to ϕ and θ.

The gradient with respect to θ is

$$\frac{\partial}{\partial \theta}\left[\mathcal{F}(\mathbf{d}; \theta, \phi)\right[= -\sum_{\mathbf{y}} \mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi} \sum_{\mathbf{z}} \mathcal{Q}_{\mathbf{z}|\mathbf{y};\phi} \frac{\partial \log\left[p(\mathbf{d}|\mathbf{y}, \theta) p(\mathbf{y}|\mathbf{z}, \theta) p(\mathbf{z}|\theta)\right]}{\partial \theta}$$

which is in the form of an expectation over the recognition distribution. Using one or more stochastic samples produced by the recognition model and averaging

$$-\frac{\partial}{\partial \theta} \log\left[p(\mathbf{d}|\mathbf{y}, \theta) p(\mathbf{y}|\mathbf{z}, \theta) p(\mathbf{z}|\theta)\right]$$

for those particular samples provides an unbiased estimate of the gradient of $\mathcal{F}(\mathbf{d}; \theta, \phi)$. This is exactly the analysis underlying the wake phase of the wake-sleep algorithm.

The gradient of $\mathcal{F}(\mathbf{d}; \theta, \phi)$ with respect to ϕ cannot apparently be calculated in quite the same manner because terms in the recognition distribution such as $\mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi}$ depend on ϕ. However, the recognition model is *factorial* and this restores sampling as a credible option. For instance, if there are n units in layer $\mathcal{Y}$ and $q_j \equiv p(y_j = 1|\mathbf{d}, \phi) = \sigma(\sum_j \phi^y_{ij} d_i)$ where ϕ^y_{ij} are the recognition weights into layer $\mathcal{Y}$ and d_i are the activities of the input units, then

$$\mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi} = \prod_{k=1}^{n} q_k^{y_k} (1 - q_k)^{1-y_k}$$

Type	Range	p_i	$p(y_i)$	$\partial \log p(y_i)/\partial w_{ji}$
sigmoid	$\{0,1\}$	$\frac{1}{1+\exp(-\sum_j w_{ji}z_j)}$	$p_i^{y_i}(1-p_i)^{1-y_i}$	$(y_i - p_i)z_j$
softmax	$\{1,..,n\}$	$\frac{\exp(\sum_j w_{ji}^{y_i} z_j)}{\sum_k \exp(\sum_j w_{ji}^k z_j)}$	p_{y_i}	$(\delta_{ky_i} - p_k)z_j$
noisy-or	$\{0,1\}$	$1-\prod_j(1-w_{ji}z_j)$	$p_i^{y_i}(1-p_i)^{1-y_i}$	$\frac{1}{p_i}(y_i - p_i)z_j$
competitive	$\{0,1\}$	$1-\frac{1}{1+\sum_j w_{ji}z_j}$	$p_i^{y_i}(1-p_i)^{1-y_i}$	$\frac{1-p_i}{p_i}(y_i - p_i)z_j$
Gaussian	$\Re$	$\frac{\exp(-(y_i - \sum_j w_{ji}z_j)^2/2\sigma_i^2)}{\sqrt{2\pi\sigma_i^2}}$	p_i	$\frac{1}{\sigma_i^2}(y_i - \sum_j w_{ki}z_k)z_j$

sigmoid For the deterministic machine (Dayan *et al*, 1995) when the z_j are also Bernoulli, we calculated $\partial \log p(y_i)/\partial w_{ji}$ for the sigmoid under the assumption that $\sum_j w_{ji}z_j$ is approximately Gaussian and used a table (Hinton & van Camp, 1993) for the effect of composing a normal distribution and the sigmoid.

softmax This is for the case in which i is an $n-$valued unit, with input weights for the kth value of w_{ji}^k, and the derivative of the log probability is with respect to w_{ji}^k.

noisy-or For the noisy-or (Pearl, 1988; Saund, 1995), z_j are also $\{0,1\}$, and $0 \leq w_{ji} < 1$ are interpreted as the contributions to the probability that if z_j is 1 then y_i is 1. For the deterministic machine, if $q_j = p(z_j = 1)$, we made the approximation of using $\hat{p}_i = 1 - \prod_j(1 - q_j w_{ji})$ to calculate $\mathcal{F}$ and its derivatives.

competitive For the competitive rule (Dayan & Zemel, 1995), z_j are also $\{0,1\}$, and $0 \leq w_{ji}$ are interpreted as the contributions to the odds that if z_j is 1 then y_i is 1. For the deterministic machine, if $q_j = p(z_j = 1)$, we made the approximation of using to calculate $\mathcal{F}$:

$$\hat{p}_i = \left(1 - \frac{1}{1+\sum_j w_{ji}q_j}\right)\left(1 - \prod_j (1 - q_j \frac{w_{ji}}{1+w_{ji}}\right)$$

Gaussian See Neal *et al* (in preparation) for the way in which the linear and Gaussian HM implements factor analysis.

Table 1: Unit activation functions.

and so, just as in Williams' (1992) REINFORCE framework:

$$\frac{\partial}{\partial \phi_{ij}} \mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi} = \mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi} \frac{\partial}{\partial \phi_{ij}} \log q_j^{y_j} (1 - q_j)^{1-y_j}. \tag{12}$$

This has the leading term $\mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi}$ and so when it is substituted as part of the derivative, equation 11 takes the form:

$$\frac{\partial}{\partial \phi} \left[\mathcal{F}(\mathbf{d}; \theta, \phi)\right] = \sum_{\mathbf{y}} \mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi} \sum_{\mathbf{z}} \mathcal{Q}_{\mathbf{z}|\mathbf{y};\phi} \mathcal{D}(\mathbf{d}, \theta, \phi) \tag{13}$$

for a particular function $\mathcal{D}$, and therefore also involves just an expectation over the recognition model. This means that it can again be calculated by averaging over recognition samples. However, whereas the derivatives with respect to θ only involve two adjacent layers, those involving ϕ are not local. The derivative with respect to a ϕ_{ij} which helps determine $\mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi}$ depends on terms in the sum in equation 13 from layer $\mathcal{Z}$ and all higher layers (if there are any). Although this information is just a scalar (it is essentially the contribution to $\mathcal{F}$ from the layers above $\mathcal{Y}$), it can be expected to be very noisy, given stochasticity in the recognition model. This will make learning very slow. Another way of looking at this is that the bandwidth of the information from higher layers used to change ϕ_{ij} is very small, and as the network gets deeper, the problem gets worse. A_{RP} and other static reinforcement learning algorithms suffer the same problem. However, this and its variants are the only methods of which we are aware for correctly optimising recognition weights.

3.3 Alternative Recognition Models

Recognition models lie at the heart of the HM. Since there is no reason to believe that the best possible top-down generative model will have an inverse that can be represented in the bottom-up and conditionally factorial model that we assume, we have explored more complicated recognition models that have greater representational power.

Recurrent Recognition

In a deterministic context, an obvious approach is to abandon the recognition model and use sophisticated mean field methods (Saul *et al*, 1995; Jaakkola *et al*, 1995). These involve an iterative approximate E-step, followed by either a hill-climbing M-step or an exact M-step. They have the strong advantage that they only use the generative weights – they do not need two sets of parameters, are not hampered by the inaccuracies that can plague the recognition model, and can easily perform the E-step based on partial information about the input. However, the iterative E-step in mean field methods can cause problems. During the M-step, the generative weights are changed to reflect the probability of the input given the model. The E-step is merely an instrument of finding the posterior distribution of the parameters given the model. However, changing the generative parameters can also make the E-step more difficult – it can make the iteration more prone to local minima. The weight changes are completely insensitive to these difficulties. Problems arising from this have been reported for mean field methods in Boltzmann machines (Hinton, 1989; Williams & Hinton, 1991). Further, even if the mean field methods settle correctly to the factorial distribution that most closely matches the true posterior distribution, this distribution may still be inferior to the one produced by a stochastic recognition net. The mean field method assumes a *globally* factorial distribution rather than one in which the factorial distribution in one layer is conditional on the stochastic binary states in the previous layer.

An alternative is to use iterative (*ie* recurrent) recognition models in the deterministic HM. We address the concern about the difficulty of iteration in two ways. First, the recognition model has its own set of connections ϕ which change to minimise $\mathcal{F}$ as before. Changes to ϕ are consequent on changes to θ, but only through the medium of $\mathcal{F}$. Second, we allow only a *fixed* and small number of iterations and use the terminal distribution as $\mathcal{Q}$. Keeping the number of iterations fixed implies that the network is being specifically required to produce good recognition distributions quickly. Any of the recurrent backpropagation algorithms (Rumelhart *et al*, 1986; Williams & Zipser, 1989; Schmidhuber, 1992) can be employed. It is not clear how best to design the recurrent connections. One obvious choice is to make them also respect the layered structure of the network – making two top-down connections between adjacent layers, one for the generative model and one the recurrent half of the recognition model. Recurrent connections that skip layers are also possible.

Note that it would be unwise to employ recurrent generative connections. This would make the system back into something like a Boltzmann machine, and would reintroduce the need to calculate the partition function.

Dangling Units

One of the key attractions of Bayesian networks is that they capture an effect called *explaining away.* Figure 2a shows an example, where two binary hidden units, y_1 and y_2 are each unlikely under the generative model $p(y_1 = 1|\theta) = p(y_2 = 1|\theta) = 0.1$, but at least one of which is necessary to explain a case in which $d = 1$. In this case, the posterior probabilities for the four possibilities for y_1 and y_2 given that $d = 1$ are roughly:

y_1	y_2	$p(y_1, y_2 \mid d = 1, \theta)$
0	0	0.00
0	1	0.47
1	0	0.47
1	1	0.05

Therefore, given that $d = 1$, it is highly likely that just one of the $y_i = 1$, and very unlikely that they are both on. However, under the recognition models that we have so far discussed, there is no interaction between y_1 and y_2 and so it is impossible to capture the negative interactions between these units. This is a form of the ubiquitous XOR problem.

The Boltzmann machine would have no trouble doing this, using a large and negative weight between these two units, but taking advantage of this would require iteration or sampling in the recognition pathway. Figure 2b shows an alternative – using a 'dangling' unit $\bar{y}$ in the recognition pathway. This unit receives a recognition connection from the input and has recognition and generative connections to y_1 and y_2, but it *lacks* a generative connection into d.[3] Unit $\bar{y}$ therefore only plays a role in the recognition model. Using the weights shown in the figure, $\bar{y}$ is almost certainly off if d is off, and otherwise comes on with probability $1/2$. This stochastic choice in the recognition model between making $\bar{y} = 0$ or $\bar{y} = 1$ chooses between $y_1 = 1$ and $y_2 = 1$, as required to solve explaining away correctly.

The advantage of dangling units is that no modification is required to the wake-sleep algorithm to learn their weights. The disadvantage is that one may in general require an unreasonably large number of such units.

Other Sampling Methods

Instead of placing all of the responsibility for inverting the generative model on the bottom-up connections, these connections can be used to facilitate more powerful methods for inverting the generative model. Even though the bottom-up model alone is insufficiently powerful to invert the generative model, in conjunction with forms of stochastic sampling, the inversion can be made exact. The better the recognition model, the faster the learning that can result, but, with sufficient sampling, using an incorrect model would have no adverse consequences on the resulting generative model.

Consider the standard HM with two hidden layers. When an input $\mathbf{d}$ is presented, the Gibbs sampling algorithm is a Monte-Carlo method of flipping the states of the hidden units in layers $\mathcal{Y}$ and $\mathcal{Z}$ which defines a Markov chain whose equilibrium distribution is $p(\mathbf{y}, \mathbf{z} \mid \mathbf{d}, \theta)$. As discussed above, this can be used to perform an approximate E-step,

[3] Radford Neal (personal communication has suggested a similar architecture, except that $\bar{y}$ actually *receives* a generative connection from the input d.

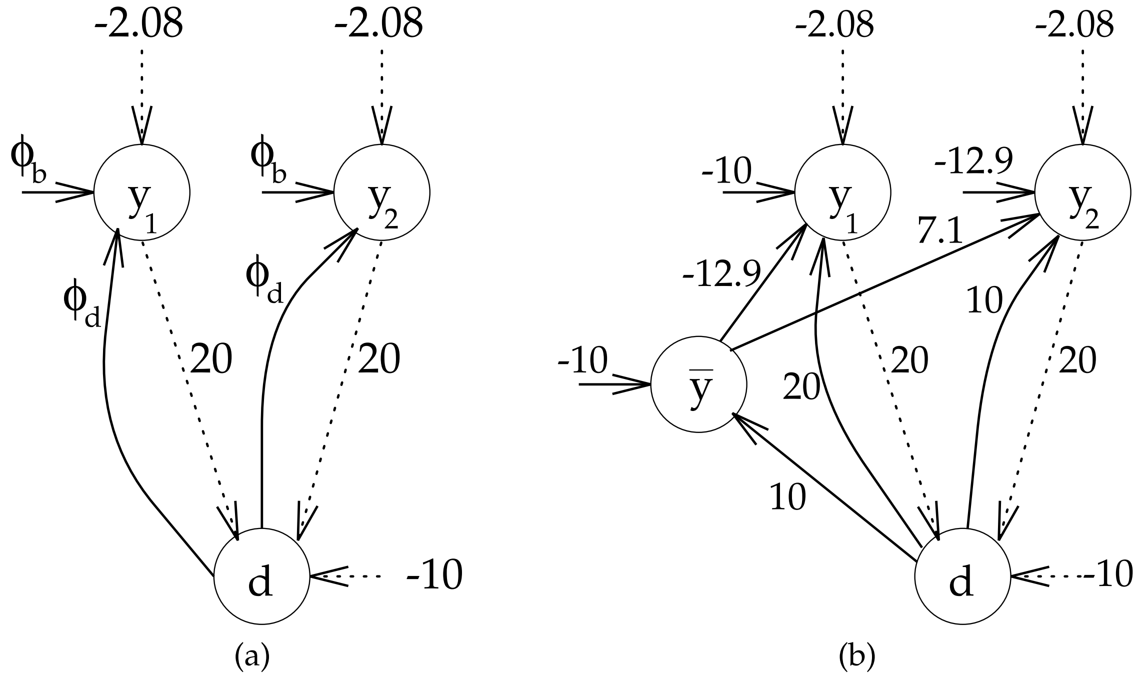

Figure 2: Dangling units. (a) The weights and architecture of a generative model that makes it improbable that both y_1 and y_2 come on together, but requires one of them to explain $d = 1$. There are no settings of the recognition weights ϕ that permit this. (b) Including a dangling unit $\bar{y}$ makes recognition possible using the weights shown. In practice, when the dangling unit learns using generative weights from y_1 and y_2, it prefers somewhat non-symmetrical solutions. The arrows without a starting unit are biases. The weights are learned, and are not quite at their final values.

and the M-step involves averaging appropriate weight changes over the resulting sample distribution. One way in which the recognition model can be used is to initialise the states of $\mathbf{y}$ and $\mathbf{z}$ by sampling successively from $\mathcal{Q}_{\mathbf{y}|\mathbf{d};\phi}$ and $\mathcal{Q}_{\mathbf{z}|\mathbf{y};\phi}$. If the recognition model awards high probability to hidden states that are likely to generate an input, then this will speed Gibbs sampling. Indeed, it turns out to work surprisingly well. Since under this scheme, the recognition model plays no further part in the sampling and the process of reaching equilibrium washes out biases due to the initial distribution, errors in the recognition model cannot jeopardise learning of the generative weights θ. However, equilibrium can be reached faster when starting from a good bottom-up distribution.

A more complicated alternative (de Sa, personal communication) is to use the recognition distribution in the context of a Metropolis sampling algorithm (Metropolis *et al*, 1953). This algorithm is also based on a Markov chain with a correct stationary distribution as defined by θ. However, instead of updating just one single unit at a time as in Gibbs sampling, a candidate update to the states of a large number of units is suggested according to a *proposal* distribution. This candidate update is then accepted or rejected according to an *acceptance* distribution. Proposal and acceptance distributions act in consort, in our case, to satisfy a condition called detailed balance which ensures that the chain's stationary distribution is $p(\mathbf{y}, \mathbf{z}|\mathbf{d}, \theta)$, as required. The speed of the Metropolis algorithm is governed by the probability that large jumps in the states of many units (to ensure rapid mixing) will frequently be proposed and then accepted, and there is a large canon of knowledge about good distributions to use. In our case, the recognition model ϕ or the generative model θ could be used to propose updates to states one whole layer at a time, provided that the way that each fails to account for the other is taken into proper account.

Unfortunately, it is not clear how much computational effort can be saved over Gibbs sampling through the use of such Metropolis algorithms. The computational effort in calculating a proposal for a change to all the units in a layer and the associated acceptance probability is, to within a constant factor, equivalent to the amount of effort involved in considering each unit in the layer in turn as in Gibbs sampling. Although there are cases such as 'explaining away' (the xor-like problem addressed by dangling units in which just one of two units in a layer needs to come on) in which Gibbs sampling faces large energy barriers to appropriate sampling, the simplistic form of the recognition distribution into a layer (which is factorial and has no hidden units from the layer below) makes the recognition model equally incapable of correctly proposing appropriate candidate states. More powerful recognition models need not, of course, be so hampered.

The Lateral HM

Although it is computationally convenient to force the hidden activities within a layer to be conditionally independent given the activities in the layer below (including the activities of any intermediate dangling units), this is far from true in the cortex. Cortical areas are replete with short- and long-range excitatory connectivity and at least short-range inhibitory connections, and recent results suggest that these are important for generating even the simplest properties of receptive fields, such as orientation selectivity (Douglas, Martin & Whitteridge, 1989; Somers, Nelson & Sur, 1995; Ben Yishai *et al*, 1995).

If recurrent connections were incorporated into the generative pathway, then the Helmholtz machine would become exactly a Boltzmann machine (BM), and would inherit the BM's problems of using the difference between two noisy estimates of the correlations between units for learning. However, if the recurrence is introduced only into the *recognition* model, and the generative model is left purely top-down, then at least some of the difficulties are absent. Sampling will still be necessary during recognition, but the approximate M-step in the generative parameters will continue to use just the delta rule based on these samples. If, furthermore, the recognition model is not made fully recurrent, but is only allowed to have links between units within single layers, then more of the problems vanish.

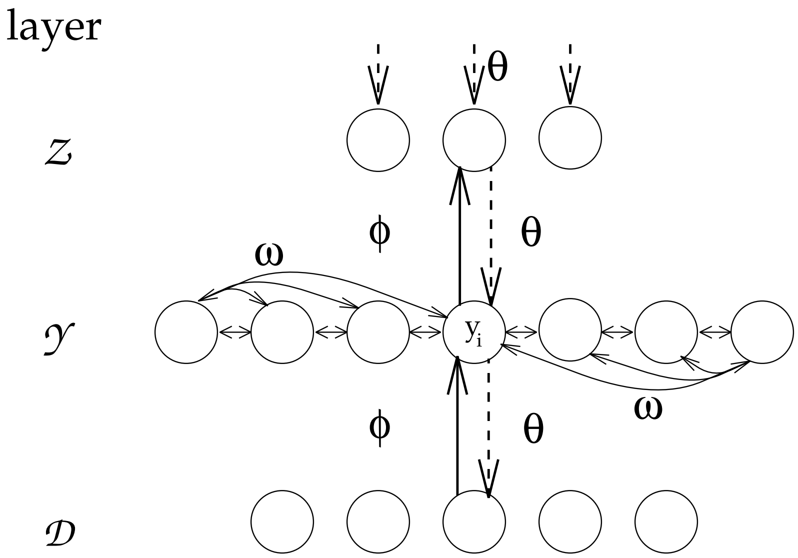

Figure 3: The Lateral HM. θ is a standard generative model; ϕ an altered recognition model; and ω are recurrent weights within layer $\mathcal{Y}$. Only some of the recurrent weights are shown.

Figure 3 shows an example of a lateral HM. θ parameterises a standard top-down generative model; ϕ a slightly altered form of bottom-up recognition model, and ω are recurrent weights within layer $\mathcal{Y}$. Units in $\mathcal{Y}$ need not be fully connected with each other – it is known that there is a tendency for lateral connectivity in the cortex to be mostly local.

This way of incorporating lateral recognition connections makes the recognition model a series of simple, supervised Boltzmann machines, one for each hidden layer. In one scheme for using these Boltzmann machines, during the wake phase, each hidden layer is allowed to settle for a while using stochastic Gibbs sampling and then a sample is taken from the equilibrium distribution. This sample provides the bottom-up input to the next layer. To reduce the delays it would probably be sensible for each hidden layer to start settling to equilibium before the layer below has settled on its final sample. After completing a bottom-up pass in this way, the generative weights can be learned in the standard way.

In the sleep phase, the top-down generative weights are used to produce a sample $\mathbf{y}^+$ from the generative model and the lateral weight between unit i and unit j in layer $\mathcal{Y}$ is incremented by $\epsilon y_i^+ y_j^+$. This corresponds to the positive phase of the Boltzmann machine learning algorithm. Then, holding the activities in the layer below fixed, each layer is allowed to settle in turn (with activities $\mathbf{y}^-$) and the weights are decremented by $\epsilon y_i^- y_j^-$, which implements the negative phase of Boltzmann machine learning. The bottom-up recognition weights are learned exactly the same way. As with the standard wake-sleep algorithm, the sleep phase minimizes the wrong Kullback-Leibler divergence, but this problem gets less serious as the recognition model gets more powerful.

Compared with a multilayer Boltzmann machine, this algorithm has distinct advantages. In the positive phase of the Boltzmann machine learning, no settling is required because the generative model is acyclic. In the negative phase, each layer settles separately and under the influence of bottom-up input. This is much more constrained, and hence much easier, than the usual unconstrained settling in the negative phase of an unsupervised Boltzmann machine. In a sense, the wake-sleep algorithm allows us to piece together many small Boltzmann machines without having to do any global settling.

It is obviously possible to apply this algorithm with *fixed* lateral recognition connections. The generative connections above a layer then have to adapt to model the effects of the lateral interactions within the layer acting on the input coming from the layer below. We are currently investigating this approach to see if sensible fixed patterns of lateral interactions make it easier for the network to extract hidden properties that vary smoothly across space.

There are also other ways of using Boltzmann machine learning within a Helmholtz machine. We can motivate one by considering the problem of Gibbs sampling in the generative model. Given the input $\mathbf{d}$ and the states of the top level units $\mathbf{z}$ and the other units in layer $\mathcal{Y}$, the probability distribution used to decide on the state of unit y_i is:

$$\frac{p(y_i = 1|\mathbf{z}, \mathbf{d}, \{y_j\}_{j\neq i}, \theta)}{p(y_i = 0|\mathbf{z}, \mathbf{d}, \{y_j\}_{j\neq i}, \theta)} = \frac{p(\mathbf{d}|y_i = 1, \{y_j\}_{j\neq i}, \theta)}{p(\mathbf{d}|y_i = 0, \{y_j\}_{j\neq i}, \theta)} \frac{p(y_i = 1|\mathbf{z}, \theta)}{p(y_i = 0|\mathbf{z}, \theta)}. \tag{14}$$

This representation in terms of odds rather than probabilities will turn out to be very convenient later. y_i is a sigmoidal unit and so its weights can be interpreted in terms of $\log$ odds. Such contributions can just be added. Given a particular state $\mathbf{z}$ of the top level units, the second fraction in the product is directly available from the top-down generative connections into y_i. Unfortunately, even given a particular state of the input units $\mathbf{d}$ and a correct recognition model, the first term in the product is not conveniently available in the bottom-up recognition connections, since it depends not only on the input, but also on the states of the other hidden units $\{y_j\}_{j\neq i}$. One can imagine ways that this could be calculated – activity in layer $\mathcal{Y}$ could inhibit those portions of the activity in layer $\mathbf{d}$ that it can predict (MacKay, 1956; Pece, 1992; Miller, Li & Desimone, 1991), making the net input to layer $\mathcal{Y}$ a function of the unpredictable components of $\mathbf{d}$. Unit i could sample to determine how changing its state affects these residuals, averaging out the effects of the simultaneous sampling of the other cells in layer $\mathcal{Y}$. However, this is not computationally very reasonable.

Rather than do this, an alternative lateral Helmholtz machine would use the bottom-up recognition connections to try to calculate just:

$$\frac{\rho_i}{1-\rho_i} = \frac{p(\mathbf{d}|y_i = 1, \theta)}{p(\mathbf{d}|y_i = 0, \theta)},$$

and use the Boltzmann machine recurrent connections within the layer and/or top-down influences to capture the other terms in equation 14. Note that ρ_i is *not* what the recognition model learns using the wake-sleep algorithm, this is rather:

$$\frac{q_i}{1-q_i} = \frac{p(y_i = 1|\mathbf{d}, \theta)}{p(y_i = 0|\mathbf{d}, \theta)}$$

which is equivalently incorrect in terms of ignoring the states of the other $\{y_j\}_{j \neq i}$ (this is the familiar approximation of a conditionally factorial recognition model). The advantage of learning ρ_i rather than q_i as the bottom-up contribution to deciding whether unit y_i should be on or not is that it is easier to combine it with other top-down or lateral influences on y_i.[4] Learning this different ϕ turns out to be simple, in the sleep phase, instead of using the straight delta rule:

$$\Delta\phi_{ij} = \epsilon(y_j - \sigma(\sum_k \phi_{kj} d_k))d_i$$

the learning rate ϵ is modified according to the probability p_j that y_j came on in the sample in layer $\mathcal{Y}$ used for learning:

$$\Delta\phi_{ij} = \epsilon(y_j - \sigma(\sum_k \phi_{kj} d_k))d_i(y_j(1-p_j) + (1-y_j)p_j). \tag{15}$$

If y_i is a sigmoid unit, then the bottom up input determining ρ_i acts like an input-dependent bias $\log[\rho_i/(1-\rho_i)]$ that can be combined with:

1. another simple constant term $\log[p(y_i = 0|\theta)/p(y_i = 1|\theta)]$ that modulates the activity of y_i by subtracting out the statistical bias in ρ_i arising since ρ_i depends only on the generative weights between two layers. This would be good for fast bottom-up recognition.

2. top down information from $\mathbf{z}$ in the form of $\log[p(y_i = 1|\mathbf{z}, \theta)/p(y_i = 0|\mathbf{z}, \theta)]$. This would be good for Gibbs or Metropolis sampling in cases where there are prior expectations or missing input data.

3. Gibbs sampling in the intra-layer Boltzmann machine defined by weights ω_{jk}. In this case, the input bias is constant, and the intra-layer weights can capture two sorts of correlations between y_i and y_j. One comes from the contributions to $p(\mathbf{d}|y_i = 1, \{y_j\}_{j \neq i}, \theta)$ from $\{y_j\}_{j \neq i}$, the other comes from the possible influences from $\mathbf{z}$, which, at least in the recognition model, are clearly just a function of $\mathbf{y}$.

[4] Although $\frac{q_i}{1-q_i} = \frac{\rho_i}{1-\rho_i} \frac{p(y_i=0|\theta)}{p(y_i=1|\theta)}$, and the last term is just a bias that does not depend on the input $\mathbf{d}$ and can easily be learned during the sleep phase, the inability of the recognition model exactly to invert the generative model means that best settings for the actual weights ϕ may differ. Also, learning ρ_i may be easier than learning q_i since it only depends on the generative weights between two layers and not the whole generative distribution above the upper layer.

The first two options do not require substantive changes to wake-sleep or the way that recognition is performed after learning. For the Boltzmann machine, we need to specify how all the weights will be modified. The sleep phase is as for the standard HM, with *no* iterative processing in layer $\mathcal{Y}$. Equation 15 is used to change the recognition weights, and the lateral weight ϕ_{jk} between units y_j and y_k is *increased* according to the observed correlations in the activities of units $\langle y_j y_k \rangle_S$. During the wake phase, the bottom-up weights are used to calculate the input-dependent bias for y_i, and then standard Boltzmann machine Gibbs sampling is performed (using some annealing schedule) to reach the equilibrium distribution. Once at equilibrium, further sampling is performed, the generative weights θ_{ji} out of layer $\mathcal{Y}$ are altered using the conventional wake-phase delta rule, and the lateral weight ϕ_{jk} is *decreased* according to the correlations in the activities of units $\langle y_j y_k \rangle_W$. This particular form of contrastive Hebbian learning (Hinton & Sejnowski, 1986) for the lateral weights is intended to reduce the Kullback-Leibler divergence between the distribution of $\mathbf{y}$ under the generative model $p(\mathbf{y}|\theta)$, and the Gibbs sampled distribution under the recognition model $\mathcal{E}_{\mathbf{d}}\left[p(\mathbf{y}|\phi, \mathbf{d}, \omega)\right]$ where the expectation is taken over the environmental distribution over the inputs $\mathbf{d}$.

3.4 Supervised HMs

We have described the HM in the general setting of density estimation which is characteristically thought of as unsupervised learning. However, supervised learning can be seen in similar terms where the task is to learn the conditional densities $p(\mathbf{d}|\mathbf{e})$. Here, $\mathbf{e}$ is the classical 'input' and $\mathbf{d}$ the classical 'output'. Ghahramani & Jordan (1994), Tresp *et al* (1994) and others have adopted various versions of this framework, using mixtures of Gaussians or hidden Markov models as the underlying density estimation devices. There are equivalently various ways to use the HM for supervised learning. For classification tasks, we reported the results of the most trivial way, in which a different density model is built for each class using the labelled training data. Subsequent classification is based on presenting the test data to each of the density models and classifying it probabilistically according to the free energy of each.

Figure 4 shows the icons which we will use to describe the various novel versions. The rectangular boxes are layers of units; the solid lines are recognition connections, the dotted lines are generative connections. The black filled-in box is $\mathbf{e}$, which is always known; the grey filled-in box is $\mathbf{d}$, the desired output, and the empty boxes are hidden layers. We show just restricted numbers of hidden layers, but, in principle, arbitrary numbers could be used.

The architecture in figure 4 is *inappropriate* for supervised learning for HM. It is based on the technique used by Ghahramani & Jordan (1994) for fitting a density model to $p(\mathbf{d}, \mathbf{e}|\theta)$. In their case, it was straightforward to integrate this density over the unknown $\mathbf{d}$ to generate $p(\mathbf{d}|\mathbf{e}, \theta) = p(\mathbf{d}, \mathbf{e}|\theta)/p(\mathbf{e}|\theta)$. In our case, not only is it unclear how to do this integration, but

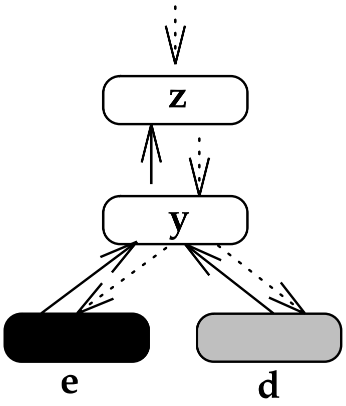

Figure 4: A joint density HM. Solid lines are recognition weights, dotted lines are generative weights, the boxes are layers of units, $\mathbf{y}$ and $\mathbf{z}$ are hidden units, $\mathbf{e}$ (black box) is the input, $\mathbf{d}$ (grey box) the output. This machine would not work well.

also the recognition model will have learned to generate the correct posteriors as a function of both $\mathbf{d}$ and $\mathbf{e}$. If $\mathbf{d}$ is not present after training, then its estimates of $\mathcal{Q}$ will be incorrect. Gibbs sampling or the Metropolis HM could be used for this missing data problem, but this is unlikely to be as good as the alternatives below. Supervised learning is not a conceptual problem for mean-field methods since they generate a new recognition distribution for each case, and treat missing inputs and true hidden units uniformly.

The Side-information HM Figure 5a shows a supervised learning machine that does work. The task for the machine is to be able to generate the distribution $p(\mathbf{d}|\mathbf{e})$. We can therefore treat $\mathbf{e}$ as providing extra input (equivalent to side information in terms of communication theory) to *both* recognition and generative pathways during learning and subsequent use. In the version shown, there are no additional hidden units between $\mathbf{e}$ and $\mathbf{y}$ or $\mathbf{z}$, and therefore completely standard wake-sleep learning can be used. If the deterministic HM is used, then there can also be hidden units on the two sets of outputs of $\mathbf{e}$ whose roles are determined during optimisation.

The Clipped HM: Figure 5b shows an alternative supervised model that also works. If $\mathbf{e}$ is substantially informationally richer than $\mathbf{d}$, it may be best to generate hidden representations by fitting a density model to $\mathbf{e}$, and use those hidden representations to generate samples over $\mathbf{d}$. This may also be the method of choice if there are substantial unlabelled data – hidden representations for $\mathbf{e}$ can be learned without needing to know $\mathbf{d}$. Standard wake-sleep is used to train the connections on the $\mathbf{e}$ pathway; the extra generative connections to $\mathbf{d}$ are trained during wake phases once the weights for $\mathbf{e}$ have converged.

The Inverse HM: Figure 5c shows a third possibility for a supervised machine which takes direct advantage of the capacity of the recognition model in the HM to learn inverse distributions. For supervised training, during wake, $\mathbf{e}$ and $\mathbf{d}$ are clamped and the generative weights are trained as normal. During sleep, the system uses its model of $p(\mathbf{d}|\theta)$ and $p(\mathbf{e}|\mathbf{d},\theta)$ to learn the inverse models $\mathcal{Q}_{\mathbf{d}|\mathbf{e}}$ and $\mathcal{Q}_{\mathbf{z}|\mathbf{d}}$. After training, the units above $\mathbf{d}$ can be discarded, since the recognition distribution $\mathcal{Q}_{\mathbf{d}|\mathbf{e}}$ itself is the desired output. In some cases (such as recognising handwritten images of digits) the distribution over $\mathbf{d}$ will be known ($0 \ldots 9$, each equally probable), in which case the layers above $\mathbf{d}$ are unnecessary, and during sleep, $\mathbf{d}$ should be sampled from this prior distribution.

The inverse machine is quite similar to the clipped machine. It differs in the way that $\mathbf{d}$ influences the sleep phase. The inverse machine is likely to be appropriate in cases such as digit recognition in which it is reasonable to regard the dependent variable $\mathbf{d}$ as being a high level 'cause' of the independent variable $\mathbf{e}$.

The inverse machine is also close in spirit to the neural heat exchanger, an algorithm proposed by Jürgen Schmidhuber (personal communication). The neural heat exchanger employs the portion of the network between $\mathbf{e}$ and $\mathbf{d}$ and has two complete sets of units, one for the top-down pathway and one for the bottom-up pathway. The heat exchanger only uses the wake phase of learning, and, on presentation of $\mathbf{d}$ and $\mathbf{e}$, it samples from $\mathbf{d}$ in the top-down network and from $\mathbf{e}$ in the bottom up network. The idea is that the activities of the units at the same layer in the two networks should be the same – and so simple delta rule learning, as in wake sleep, can be used, with the activities in each network being the target for learning the weights in the other. The trouble with the heat exchanger is that in non-deterministic domains, there is no reason why the particular $\mathbf{e}$ generated from a particular $\mathbf{d}$ should be the one that is observed – in terms of the digits, there is no reason why the top-down network should generate exactly the same image of a 0 that the bottom-up net sees for a particular training case. Wake-sleep learning overcomes this problem by using completely separate learning phases for the two classes of weights.

3.5 The Helmholtz Machine through Time

The major disadvantage of conventional HMMs 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 amount of information that needs to be held (Williams & Hinton, 1991; Ghahramani & Jordan, 1995). If the states are represented in a distributed manner, 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.

HMs (and similarly mean field methods, Saul & Jordan, 1995) offer the same way out of the impasse that they did for complicated static generative models. For the E-step, they use recognition distributions for the hidden states that are not the exact posteriors

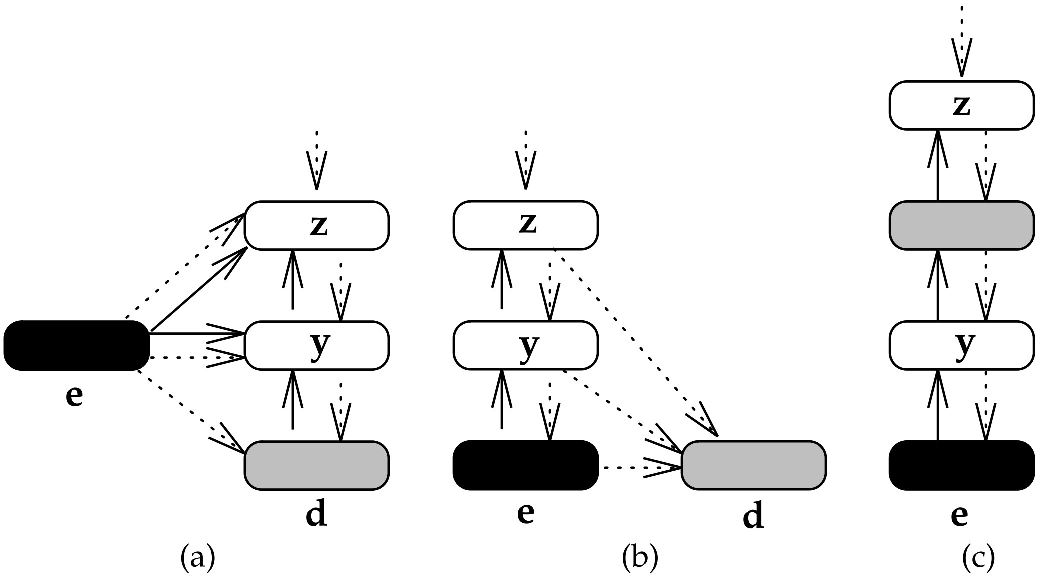

Figure 5: Supervised Helmholtz machines. a) The side-information HM explicitly builds a conditional probability density function using the known $\mathbf{e}$ like side information in coding. b) The clipped HM builds a density model of the known $\mathbf{e}$ and uses its hidden states to build a generative model of the unknown $\mathbf{d}$. c) The inverse HM builds generative models of the unknown $\mathbf{d}$ and a conditional model of the knowns $\mathbf{e}$ given the unknown and uses the sleep phase to learn to invert this model.

under the generative model, and use gradient descent for the M-step on the basis of these distributions.

The HMTT is shown in figure 6. As in backpropagation through time (Rumelhart *et al*, 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.[5] In the most straightforward implementation of the HMTT, these two influences are 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

[5] 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.

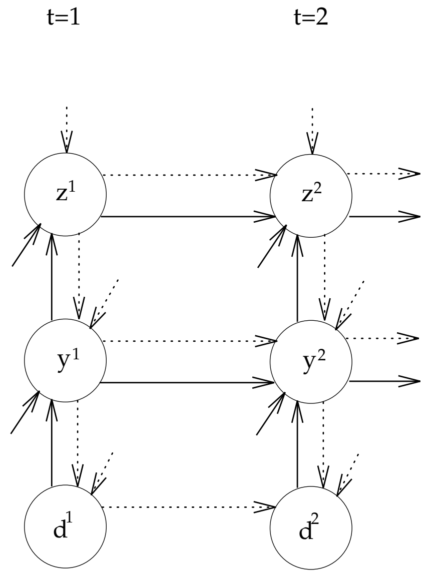

Figure 6: The Helmholtz machine Through Time. This shows the first two timesteps for a three-layer HMTT. Only one unit per layer is shown for convenience. Note that recognition (solid) and generative (dotted) connections both point in the direction of increasing time. More complicated architectures are also possible with direct 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.

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:

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

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 use the computationally efficient forward-backward algorithm (Baum & 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 $\mathcal{Q}_{\mathbf{y}^1,\mathbf{y}^2,\ldots,\mathbf{z}^1,\mathbf{z}^2,\ldots}(\mathbf{d}^1, \mathbf{d}^2, \ldots; \phi)$. 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 their conventional 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 numerical values.

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 $\mathbf{d}^{t+1}, \ldots$. Failing to do this in the static case is the problem that prevents Schmidhuber's neural heat-exchanger from working well. 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 observed data from further into the future can also be used (Neal, personal communication) 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 $t + \tau$, where τ 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 phase consists of generating sequences of hidden and observable states from the generative model, and training the recognition model to be its inverse, again using the delta rule. The fact that there are both recognition and generative connections in the same direction between hidden units at adjacent timesteps causes no problems. For wake-sleep, it is essential only that neither the recognition model nor the generative model contain directed cycles. Unfortunately, unlike for the temporally insensitive case, in the absence of information from the future (as would be provided by 'lookahead' recognition connections), no learning method can move down the gradient of the log-likelihood. It therefore becomes an empirical question as to how well it works (Hinton *et al*, 1995).

4 Discussion

We have described the original version of the Helmholtz machine and a number of closely related machines that have been designed with different tasks or mechanisms in mind, *eg* the supervised machines, the Helmholtz machine through time and the lateral HM. What relevance do any of these machines have to cortical processing and the four questions underlying this special issue?

In our models, the computational goal for unsupervised learning is probability density estimation. Along with Kawato, Hayakama & Inui (1993), we claim that the cortex builds a complex and hierarchical *generative* model[6] of its inputs, where the layers in the hierarchy roughly correspond to areas in the sense of Felleman & Van Essen (1991). The neurons in these areas are the hidden units, and they represent structure in the generative model. For instance, in a generative model of images of digits, hierarchical hidden structure might consist of such choices as which digit, what font, what curlicues, *etc.* The process of setting the weights of the connections between the neurons (and possibly determining the connections themselves, Friedlander & Martin, 1989; Antonini & Stryker, 1993) to improve the generative model gives the hidden units their functions. Although Barlow has not stressed so greatly the issue of learning probabilistic models of the inputs, the final roles accorded to the hidden units turn out to be closely analogous to his suggestions that neurons should respond to few patterns (be *sparse*), that neurons with common targets should generally fire in a somewhat independent manner (be *factorial* or *decorrelated*), and that they should respond to statistically surprising conjunctions of their inputs. However, all these desiderata are secondary to the main goal of density estimation.

As far as the coding scheme adopted by cortex is concerned, the HM is somewhat agnostic. Much of the work on unsupervised learning as activity dependent development has focused on bottom-up learning, and is driven by objective functions such as the sparseness or mutual independence of activities. Mutual independence is good, for instance, since later stages can combine information without having to build very complicated probability models. However, these metrics are somewhat unsatisfying since they are not couched in terms of a

[6] This model is sometimes called a *forward* model, but since we see it as being top-down, we have avoided use of this terminology.

whole goal for development. For instance, there are probably many sources of independent noise afflicting the inputs, but it would be unfortunate were cortex to arrange for mutual independence by filtering out the signal in favour of the noise. By contrast, the HM has optimal density estimation as its underlying motivation – and doing this correctly requires inducing the statistical structure of the inputs. There will be cases in which this will entail that the hidden activities it forms will be sparse, but sparseness itself plays no particular role. Equally, the HM can work with local softmax groups within a layer, which is a popular reading of the effect of short-range cortical connectivity, provided that they are combined in a factorial manner. The lateral HM points towards learning methods for such intra-cortical connections, but needs further exploration.

With Carpenter & Grossberg, (1987), Kawato *et al* (1993) and Mumford (1994), the HM is clearly not so agnostic about the existence of internal models of the world and the role of top-down and bottom-up cortical connections in learning those models and their inverses. In particular, the wake-sleep algorithm (Hinton *et al*, 1995) can be used to make precise hypotheses about the nature of the learning that occurs in both sorts of connections.

Model building is the main aim for the HM. There is something slightly unsatisfying about this which emerged in some of the simulations for Hinton *et al* (1995). We tried taking images of all ten digits and training one large and deep density model on the whole collection. We had expected that the most significant structure in the input would be the identity of a digit, and that units towards the top of the hierarchy would tend to respond to just one digit, or maybe even at a finer granularity than that – just one style of one digit. Structure in the world, in terms of the way we interact with it, should be matched by statistical structure that the HM or other density estimation devices can extract. However, the HM did not separate out the digits in this manner – top level units were not digit specific. Worse, it turned out not even to be fruitful to attempt to recognise digits from the activities of the hidden units – the HM is apparently capable of extracting the 'wrong' sort of structure. We are presently investigating why.

We have used two main rules for synaptic plasticity – the delta rule (Konorski, 1948; Widrow & Stearns, 1985; Rescorla & Wagner, 1972) in the wake-sleep algorithm and a correlational or Hebbian rule (Hebb, 1949) for the Boltzmann machine learning in the lateral HM. However, our main goal in designing learning rules was to make all the information required for learning be *local,* a minimal requirement for biological plausibility. Although most of the evidence on neural plasticity points to the implementation of Hebb rules, it is well known that delta-like rules can be implemented by Hebb-like mechanisms (Artola, Bröcher & Singer, 1990; Hancock, Smith and Phillips, 1991; Montague *et al*, 1994).

Unsupervised learning of representations is only part of the overall computational economy of the brain – reinforcement and associative learning based on these representations must also play major roles. The HM offers an hypothesis for how density estimation in complex hierarchical models can be implemented and suggests a role for puzzling anatomical motifs such as reciprocal connectivity. How closely the HM maps onto the microcircuitry of cortex is under investigation.

Acknowledgements

We are very grateful to our many collaborators on Helmholtz machines, including Radford Neal, Brendan Frey, Rich Zemel, Virginia de Sa, Ava To, Mike Revow and Drew van Camp, and to Chris Williams, Mike Jordan, Larry Saul, Tommi Jaakkola and the whole Neuron group at the University of Toronto for extensive ideas and discussion on Helmholtz machines. Support was from NSERC and start-up funds from MIT. GEH is a fellow of the Canadian Institute for Advanced Research.

References

Amari, S (1995). The EM algorithm and information geometry in neural network learning. *Neural Computation,* **7**, 13-18.

Antonini, A & Stryker, MP (1993). Development of individual geniculocortical arbors in cat striate cortex and effects of binocular impulse blockade. *Journal of Neuroscience,* **13**, 3549-3573.

Artola, A, Brocher, S & Singer, W (1990). Different voltage-dependent thresholds for inducing long-term depression and long-term potentiation in slices of rat visual cortex. *Nature,* **347**, 69-72.

Barlow, HB (1989). Unsupervised learning. *Neural Computation,* **1**, 295-311.

Barto, AG & Anandan, P (1985). Pattern recognizing stochastic learning automata. *IEEE Transactions on Systems, Man and Cybernetics,* **15**, 360-374.

Barto, AG, Sutton, RS & Watkins, CJCH (1989). Learning and sequential decision making. In M Gabriel & J Moore, editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks.* Cambridge, MA: MIT Press, Bradford Books.

Baum, LE & Eagon, JA (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**, 360-363.

Baum, LE, Petrie, E, Soules, G & Weiss, N (1970). A maximisation technique occurring in the statistical analysis of probabilistic functions of Markov Chains. *Ann. Math. Stat.* **41**, 1, pp 164-171.

Becker, S & Hinton, GE (1992). A self-organizing neural network that discovers surfaces in random-dot stereograms. *Nature,* **355**, 161-163.

Ben Yishai, R, Baror, RL & Sompolinsky, H (1995). Theory of orientation tuning in visual cortex. *Proceedings of the National Academy of Sciences,*, **92**, 3844-3848.

Carpenter, G & Grossberg, S (1987). A massively parallel architecture for a self-organizing neural pattern recognition machine. *Computer Vision, Graphics and Image Processing,* **37**, 54-115.

Dayan, P, Hinton, GE, Neal, RM & Zemel, RS (1995). The Helmholtz machine. *Neural Computation,* **7**, 889-904.

Dayan, P & Zemel, RS (1995). Competition and multiple cause models. *Neural Computation,* **7**, 565-579.

Dempster, AP, Laird, NM & Rubin, DB (1977). Maximum likelihood from incomplete data via the EM algorithm. *Proceedings of the Royal Statistical Society,* **B-39**, 1–38.

Douglas, RJ, Martin, KA & Whitteridge, D (1989). A canonical microcircuit for neocortex. *Neural Computation,* **1**, 480-488.

Everitt, BS (1984). *An Introduction to Latent Variable Models.* London: Chapman and Hall.

Felleman DJ & Van Essen DC (1991) Distributed hierarchical processing in the primate cerebral cortex. *Cerebral Cortex,* **1**, 1-47.

Friedlander, MJ & Martin, KAC (1989). Development of Y-axon innervation of cortical area 18 in the cat. *Journal of Physiology,* **416**, 183-213.

Ghahramani, Z & Jordan, MI (1994). Supervised learning from incomplete data via an EM approach. In JD Cowan, G Tesauro and J Alspector, editors, *Advances in Neural Information Processing Systems 6.* San Mateo, CA: Morgan Kaufmann, 120-127.

Ghahramani, Z & Jordan, MI (1995). Factorial hidden Markov models. Computational Cognitive Science Technical Report 9502, MIT.

Hancock, PJB, Smith, LS & Phillips, WA (1991). A biologically supported error-correcting learning rule. *Neural Computation,* **3**, 201-212.

Hebb, DO (1949) *The organization of behavior*. New York: Wiley.

Hinton, GE (1989). Deterministic Boltzmann learning performs steepest descent in weight-space. *Neural Computation,* **1**, 143-150.

Hinton, GE, Dayan, P, Frey, BJ & Neal, RM (1995). The wake-sleep algorithm for unsupervised neural networks. *Science,* **268**, 1158-1160.

Hinton, GE & Sejnowski, TJ (1986). Learning and relearning in Boltzmann machines. In DE Rumelhart, JL McClelland and the PDP research group, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations,* Cambridge, MA: MIT Press, 282-317.

Hinton, GE & van Camp, D (1993). Keeping neural networks simple by minimizing the description length of the weights. In *Proceedings of the the Sixth Annual Conference on Computational Learning Theory,* 5-13. Santa Cruz, CA.

Hinton, GE & Zemel, RS (1994). Autoencoders, minimum description length and Helmholtz free energy. In JD Cowan, G Tesauro and J Alspector, editors, *Advances in Neural Information Processing Systems 6.* San Mateo, CA: Morgan Kaufmann, 3-10.

Jaakkola, T, Saul, LK & Jordan, MI (1995). Fast learning by bounding likelihoods in sigmoid belief nets. *Advances in Neural Information Processing Systems, 8.*

Jordan, MI & Xu, L (1993). *Convergence Results for the EM Approach to Mixtures of Experts Architectures.* CBCL Memo 87, Centre for Biological and Computational Learning, MIT.

Kawato, M, Hayakama, H & Inui, T (1993). A forward-inverse optics model of reciprocal connections between visual cortical areas. *Network,* **4**, 415-422.

Keeler, JD, Rumelhart, DE & Leow, WK (1991). Integrated segmentation and recognition of hand-printed numerals. In RP Lippmann, J Moody & DS Touretzky, editors, *Advances in Neural Information Processing Systems 3.* San Mateo, CA: Morgan Kaufmann, 557-563.

Kohonen, T (1982). Self-organized formation of topologically correct feature maps. *Biological Cybernetics,* **43**, 59-69.

Konorski, J (1948) *Conditioned reflexes and neuron organization*. Cambridge: Cambridge University Press.

Luttrell, SP (1995). A componential self-organizing neural network. Submitted to *Advances in Neural Information Processing Systems, 8.*

MacKay, DM (1956). The epistemological problem for automata. In CE Shannon & J McCarthy, editors, *Automata Studies,* Princeton, NJ: Princeton University Press, 235-251.

Metropolis, N, Rosenbluth, AW, Rosenbluth, MN, Teller, AH & Teller, E (1953). Equation fo state calculations by fast computing machines. *Journal of Chemical Physics,* **21**, 1087-1092.

Miller, EK, Li, L & Desimone, R (1991). A neural mechanism for working and recognition memory in inferior temporal cortex. *Science,* **254**, 1377-9.

Montague, PR, Dayan, P, Nowlan, SJ, Pouget, A & Sejnowski, TJ (1993). Using aperiodic reinforcement for directed self-organization. In CL Giles, SJ Hanson & JD Cowan, editors, *Advances in Neural Information Processing Systems, 5,* 969-977. San Mateo: CA: Morgan Kaufmann.

Mumford, D (1994). Neuronal architectures for pattern-theoretic problems. In C Koch and J Davis, editors, *Large-Scale Theories of the Cortex.* Cambridge, MA: MIT Press, 125-152.

Neal, RM (1992). Connectionist learning of belief networks. *Artificial Intelligence,* **56**, 71-113.

Neal, RM (1993). *Probabilistic Inference using Markov Chain Monte Carlo Methods.* Technical Report CRG-TR-93-1. Department of Computer Science, University of Toronto.

Neal, RM & Hinton, GE (1994). A new view of the EM algorithm that justifies incremental and other variants. Submitted to *Biometrika.*

Pearl, J (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo, CA: Morgan Kaufmann.

Pece, AEC (1992). Redundancy reduction of a Gabor representation: a possible computational role for feedback from primary visual cortex to lateral geniculate nucleus. In I Aleksander & J Taylor, editors, *Artificial Neural Networks, 2.* Amsterdam: Elsevier, 865-868.

Peterson, C & Anderson, JR (1987). A mean field theory learning algorithm for neural networks. *Complex Systems,* **1**, 995-1019.

Rescorla, RA & Wagner, AR (1972). A theory of Pavlovian conditioning: The effectiveness of reinforcement and non-reinforcement. In AH Black & WF Prokasy, editors, *Classical Conditioning II: Current Research and Theory,* pp 64-69. New York, NY: Appleton-Century-Crofts.

Rissanen, J (1989). *Stochastic Complexity in Statistical Inquiry.* Singapore: World Scientific.

Rubin, DB & Thayer, DT (1982). EM algorithms for ML factor analysis. *Psychometrika,* **47**, 69-76.

Rumelhart, DE, Durbin, R, Golden, R & Chauvin, Y (1995). Backpropagation: The basic theory. In Y Chauvin & DE Rumelhart, editors, *Backpropagation: Theory, Architectures and Applications,* 1-34. Hillsdale, NJ: Erlbaum.

Rumelhart, DE, Hinton, GE & Williams, RJ (1986). Learning internal representations by back-propagating errors. *Nature,* **323**, 533-536.

Saul, LK, Jaakkola, T & Jordan, MI (1995). *Mean Field Theory for Sigmoid Belief Networks.* Computational Cognitive Science Technical Report 9501, MIT.

Saul, L & Jordan, MI (1995). Boltzmann chains and hidden Markov models. In G Tesauro, DS Touretzky & T Leen, editors, *Advances in Neural Information Processing Systems 7.* San

Mateo, CA: Morgan Kaufmann.

Saund, E (1995). A multiple cause mixture model for unsupervised learning. *Neural Computation,* **7**, 51-71.

Schmidhuber, J (1992). A fixed size storage $\mathcal{O}(n^3)$ time complexity learning algorithm for fully recurrent continually running networks. *Neural Computation,* **4**, 243-248.

Somers, DC, Nelson, SB & Sur, M (1995). An emergent model of orientation selectivity in cat visual cortical simple cells. *Journal of Neuroscience,* in press.

Sutton, RS (1988). Learning to predict by the methods of temporal difference. *Machine Learning,* **3**, pp 9-44.

Sutton, R (1991). Dyna, an integrated architecture for learning, planning and reacting. *Working Notes of the 1991 AAAI Spring Symposium on Integrated Intelligent Architectures* and *SIGART Bulletin*, **2**, 160-163.

Tresp, V, Ahmad, S & Neunier, R (1994). Training neural networks with deficient data. In JD Cowan, G Tesauro and J Alspector, editors, *Advances in Neural Information Processing Systems 6.* San Mateo, CA: Morgan Kaufmann, 128-135.

Widrow, B, Stearns, SD (1985) *Adaptive signal processing*. Englewood Cliffs, NJ: Prentice-Hall.

Williams, CKI & Hinton, G (1991). Mean field networks that learn to discriminate temporally distorted strings. In D Touretzky, J Elman & T Sejnowski, editors, *Connectionist Models: Proceedings of the 1990 Summer School,* 18-22. San Mateo, CA: Morgan Kaufmann.

Williams, RJ (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning,* **8**, 229-56.

Williams, RJ & Zipser, D (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation,* |bf 1, 270-280.

Xu, L & Jordan, MI (1995). *On Convergence Properties of the EM Algorithm for Gaussian Mixtures.* CBCL Paper 111, Centre for Biological and Computational Learning, MIT.

Zemel, RS (1994). *A Minimum Description Length Framework for Unsupervised Learning.* PhD Dissertation, Computer Science, University of Toronto, Canada.

Figure 1:
A three-layer Helmholtz machine. Weights θ define a top-down generative model in layers $\mathcal{Z}$, $\mathcal{Y}$ and $\mathcal{D}$ of stochastic binary units. $\mathcal{D}$ are the observables, and the task for θ is maximising the probability $p(\mathbf{d}|\theta)$. Weights ϕ define a bottom-up recognition model that attempts to invert the generative model.

Figure 2
Dangling units. (a) The weights an architecture of a generative model that makes it improbable that both y_1 and y_2 come on together, but requires one of them to explain $d = 1$. There are no settings of the recognition weights ϕ that permit this. (b) Including a dangling unit $\bar{y}$ makes recognition possible using the weights shown. In practice, when the dangling unit learns using generative weights from y_1 and y_2, it prefers somewhat non-symmetrical solutions. The arrows without a starting unit are biases. The weights are learned, and are not quite at their final values.

Figure 3
The Lateral HM. θ is a standard generative model; ϕ an altered recognition model; and ω are recurrent weights within layer $\mathcal{Y}$. Only some of the recurrent weights are shown.

Figure 4
A joint density HM. Solid lines are recognition weights, dotted lines are generative weights, the boxes are layers of units, $\mathbf{y}$ and $\mathbf{z}$ are hidden units, $\mathbf{e}$ (black box) is the input, $\mathbf{d}$ (grey box) the output. This machine would not work well.

Figure 5
Supervised Helmholtz machines. a) The side-information HM explicitly builds a conditional probability density function using the known $\mathbf{e}$ like side information in coding. b) The clipped HM builds a density model of the known $\mathbf{e}$ and uses its hidden states to build a generative model of the unknown $\mathbf{d}$. c) The inverse HM builds generative models of the unknown $\mathbf{d}$ and a conditional model of the knowns $\mathbf{e}$ given the unknown and uses the sleep phase to learn to invert this model.

Figure 6
The Helmholtz machine Through Time. This shows the first two 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 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.

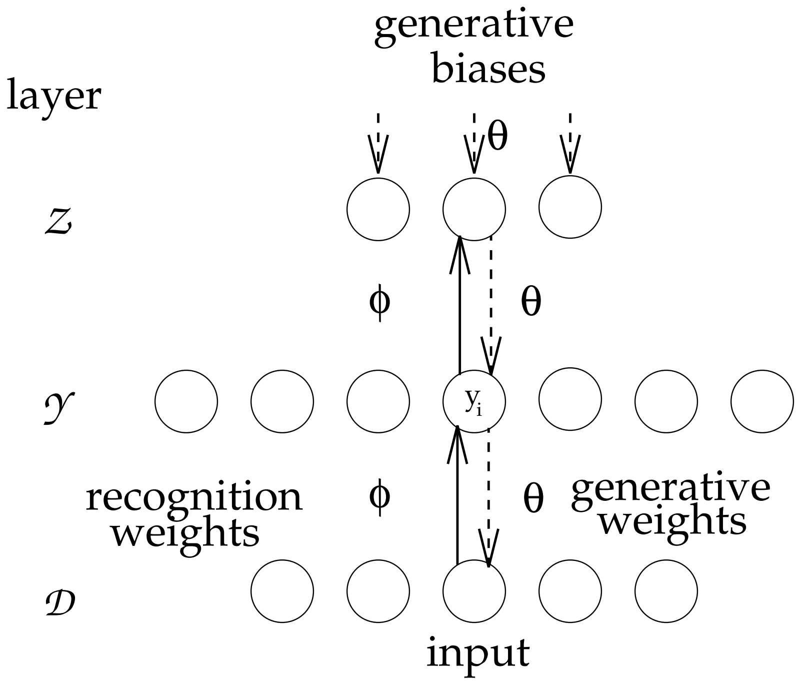
generative
biases
layer
θ
$\mathcal{Z}$
ϕ
θ
$\mathcal{Y}$
y_i
recognition
weights
ϕ
θ
generative
weights
$\mathcal{D}$
input

Figure 1

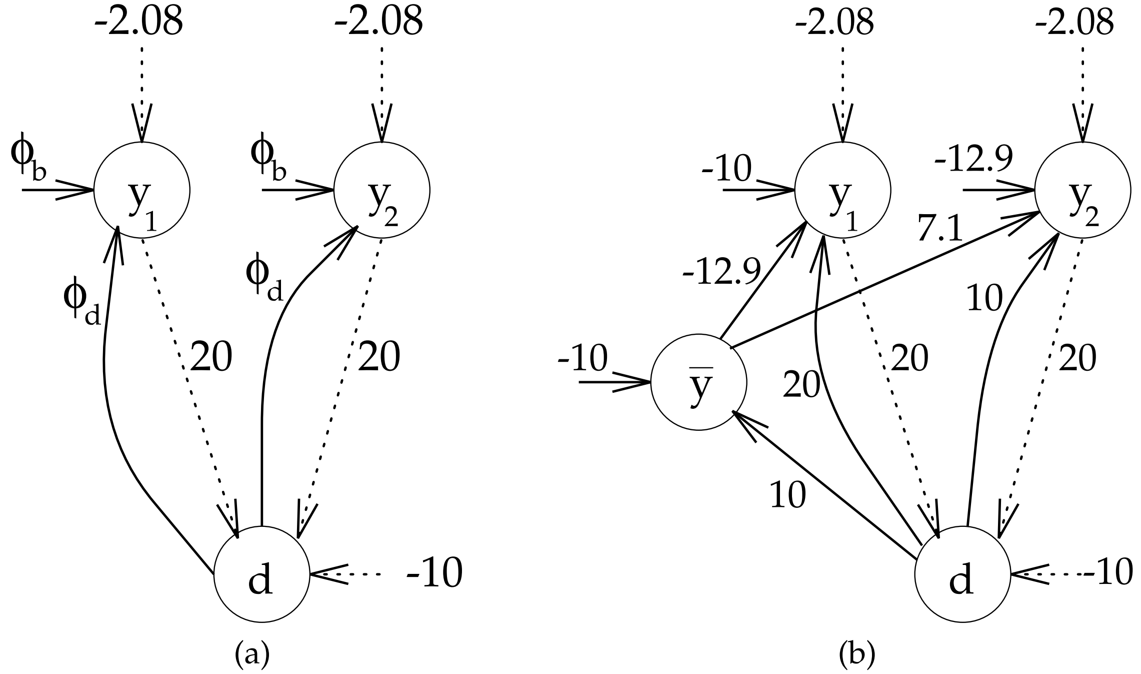
-2.08
-2.08
ϕb
y1
ϕb
y2
ϕd
ϕd
20
20
d
-10
(a)
-2.08
-2.08
-10
y1
-12.9
y2
7.1
-12.9
10
-10
ȳ
20
20
20
10
d
-10
(b)

Figure 2

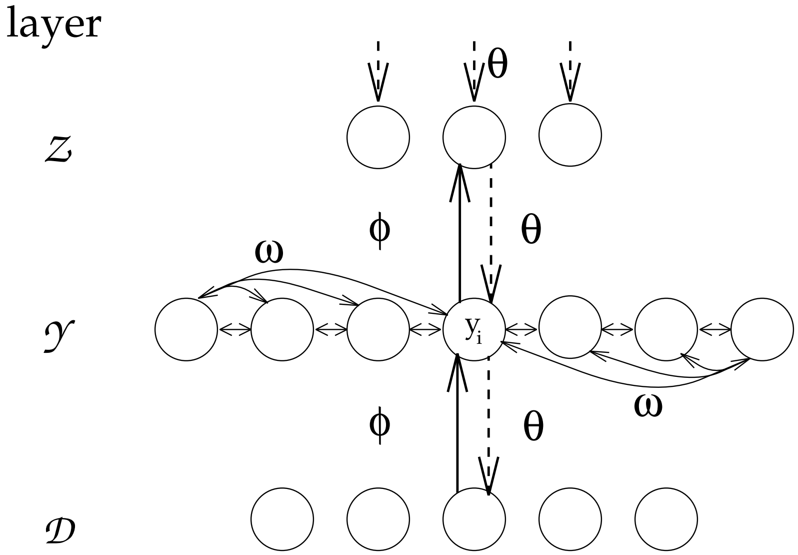

Figure 3

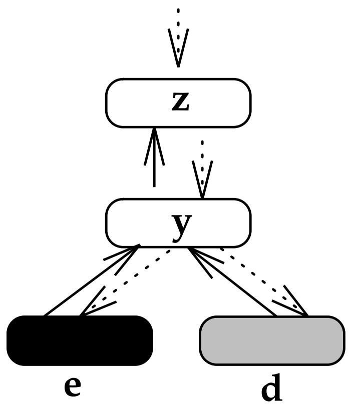

Figure 4

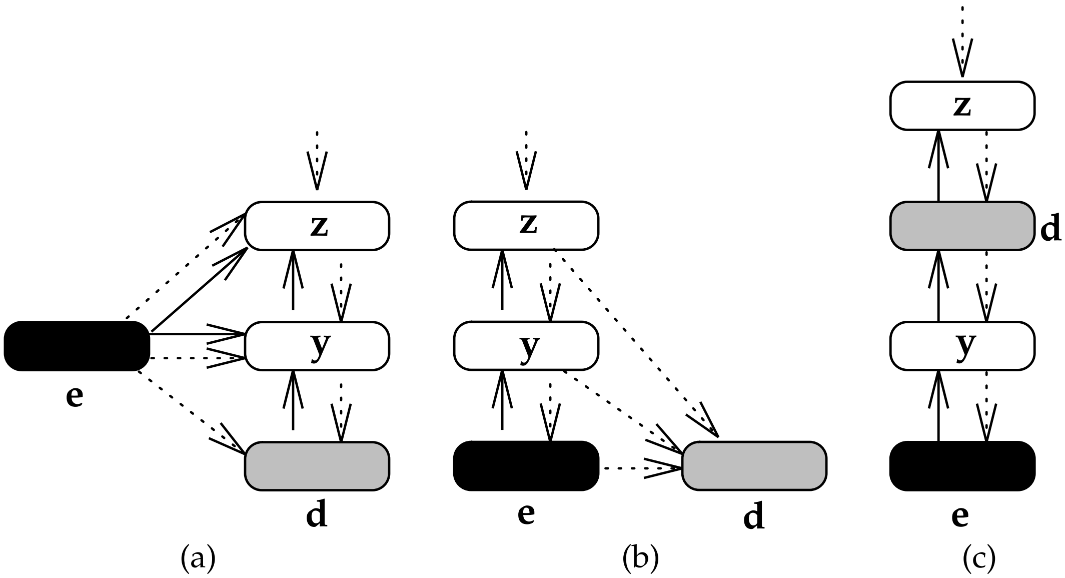

Figure 5

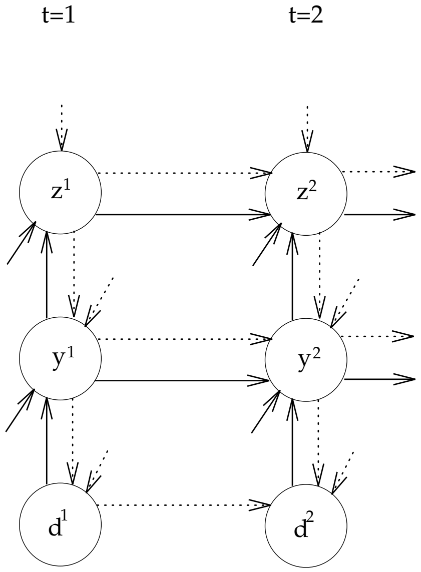

Figure 6