ARTICLE Communicated by Javier Movellan

Training Products of Experts by Minimizing Contrastive Divergence

Geoffrey E. Hinton
hinton@cs.toronto.edu
Gatsby Computational Neuroscience Unit, University College London, London WC1N 3AR, U.K.

It is possible to combine multiple latent-variable models of the same data by multiplying their probability distributions together and then renormalizing. This way of combining individual "expert" models makes it hard to generate samples from the combined model but easy to infer the values of the latent variables of each expert, because the combination rule ensures that the latent variables of different experts are conditionally independent when given the data. A product of experts (PoE) is therefore an interesting candidate for a perceptual system in which rapid inference is vital and generation is unnecessary. Training a PoE by maximizing the likelihood of the data is difficult because it is hard even to approximate the derivatives of the renormalization term in the combination rule. Fortunately, a PoE can be trained using a different objective function called "contrastive divergence" whose derivatives with regard to the parameters can be approximated accurately and efficiently. Examples are presented of contrastive divergence learning using several types of expert on several types of data.

1 Introduction

One way of modeling a complicated, high-dimensional data distribution is to use a large number of relatively simple probabilistic models and somehow combine the distributions specified by each model. A well-known example of this approach is a mixture of gaussians in which each simple model is a gaussian, and the combination rule consists of taking a weighted arithmetic mean of the individual distributions. This is equivalent to assuming an overall generative model in which each data vector is generated by first choosing one of the individual generative models and then allowing that individual model to generate the data vector. Combining models by forming a mixture is attractive for several reasons. It is easy to fit mixtures of tractable models to data using expectation-maximization (EM) or gradient ascent, and mixtures are usually considerably more powerful than their individual components. Indeed, if sufficiently many models are included in

the mixture, it is possible to approximate complicated smooth distributions arbitrarily accurately.

Unfortunately, mixture models are very inefficient in high-dimensional spaces. Consider, for example, the manifold of face images. It takes about 35 real numbers to specify the shape, pose, expression, and illumination of a face, and under good viewing conditions, our perceptual systems produce a sharp posterior distribution on this 35-dimensional manifold. This cannot be done using a mixture of models, each tuned in the 35-dimensional manifold, because the posterior distribution cannot be sharper than the individual models in the mixture and the individual models must be broadly tuned to allow them to cover the 35-dimensional manifold.

A very different way of combining distributions is to multiply them together and renormalize. High-dimensional distributions, for example, are often approximated as the product of one-dimensional distributions. If the individual distributions are uni- or multivariate gaussians, their product will also be a multivariate gaussian so, unlike mixtures of gaussians, products of gaussians cannot approximate arbitrary smooth distributions. If, however, the individual models are a bit more complicated and each contains one or more latent (i.e., hidden) variables, multiplying their distributions together (and renormalizing) can be very powerful. Individual models of this kind will be called "experts."

Products of experts (PoE) can produce much sharper distributions than the individual expert models. For example, each expert model can constrain a different subset of the dimensions in a high-dimensional space, and their product will then constrain all of the dimensions. For modeling handwritten digits, one low-resolution model can generate images that have the approximate overall shape of the digit, and other more local models can ensure that small image patches contain segments of stroke with the correct fine structure. For modeling sentences, each expert can enforce a nugget of linguistic knowledge. For example, one expert could ensure that the tenses agree, one could ensure that there is number agreement between the subject and verb, and one could ensure that strings in which color adjectives follow size adjectives are more probable than the the reverse.

Fitting a PoE to data appears difficult because it appears to be necessary to compute the derivatives , wrt the parameters , of the partition function that is used in the renormalization . As we shall see , however , these derivatives can be finessed by optimizing a less obvious objective function than the log likelihood of the data.

2 Learning PoEs by ML

We consider individual expert models for which it is tractable to compute the derivative of the log probability of a data vector wrt the

parameters of the expert. We combine n individual expert models as follows:

$$p(\mathbf{d} \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n) = \frac{\Pi_m f_m(\mathbf{d} \mid \boldsymbol{\theta}_m)}{\sum_{\mathbf{c}} \Pi_m f_m(\mathbf{c} \mid \boldsymbol{\theta}_m)}, \tag{2.1}$$

where $\mathbf{d}$ is a data vector in a discrete space, $\boldsymbol{\theta}_m$ is all the parameters of individual model m, $f_m(\mathbf{d} \mid \boldsymbol{\theta}_m)$ is the probability of $\mathbf{d}$ under model m, and $\mathbf{c}$ indexes all possible vectors in the data space.[1] For continuous data spaces, the sum is replaced by the appropriate integral.

For an individual expert to fit the data well, it must give high probability to the observed data, and it must waste as little probability as possible on the rest of the data space. A PoE, however, can fit the data well even if each expert wastes a lot of its probability on inappropriate regions of the data space, provided different experts waste probability in different regions.

The obvious way to fit a PoE to a set of observed i.i.d. data vectors is to follow the derivative of the log likelihood of each observed vector, $\mathbf{d}$, under the PoE.

$$\frac{\partial \log p(\mathbf{d} \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)}{\partial \boldsymbol{\theta}_m} = \frac{\partial \log f_m(\mathbf{d} \mid \boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta}_m} - \sum_{\mathbf{c}} p(\mathbf{c} \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n) \frac{\partial \log f_m(\mathbf{c} \mid \boldsymbol{\theta}_m)}{\partial \boldsymbol{\theta}_m}. \tag{2.2}$$

The second term on the rhs of eq (2.2) is just the expected derivative of the log probability of an expert on fantasy data, $\mathbf{c}$, that is generated from the PoE . So assuming that each of the individual experts has a tractable derivative , the obvious difficulty in estimating the derivative of the log probability of the data under the PoE is generating correctly distributed fantasy data . This can be done in various ways . For discrete data , it is possible to use rejection sampling . Each expert generates a data vector independently , and this process is repeated until all the experts happen to agree. Rejection sampling is a good way of understanding how a PoE specifies an overall probability distribution and how different it is from a causal model , but it is typically very inefficient . Gibbs sampling is typically much more efficient . In Gibbs sampling , each variable draws a sample from its posterior distribution given the current states of the other variables (Geman & Geman , 1984). Given the data, the hidden states of all the experts can always be updated in parallel because they are conditionally independent . This is an important consequence of the product formulation.[3] If the

[1] So long as $f_m(\mathbf{d} \mid \theta_m)$ is positive, it does not need to be a probability at all, though it will generally be a probability in this article.

[3] The conditional independence is obvious in the undirected graphical model of a PoE because the only path between the hidden states of two experts is via the observed data.

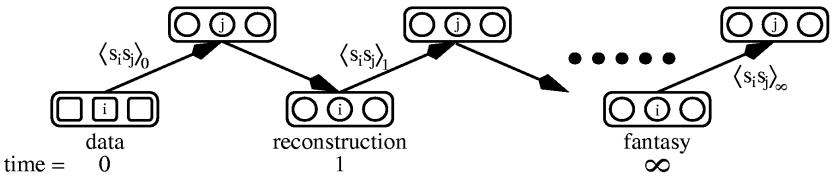


Figure 1: A visualization of alternating Gibbs sampling. At time 0, the visible variables represent a data vector, and the hidden variables of all the experts are updated in parallel with samples from their posterior distribution given the visible variables. At time 1, the visible variables are all updated to produce a reconstruction of the original data vector from the hidden variables, and then the hidden variables are again updated in parallel. If this process is repeated sufficiently often, it is possible to get arbitrarily close to the equilibrium distribution. The correlations $\langle s_i s_j \rangle$ shown on the connections between visible and hidden variables are the statistics used for learning in RBMs, which are described in section 7.

individual experts also have the property that the components of the data vector are conditionally independent given the hidden state of the expert, the hidden and visible variables form a bipartite graph, and it is possible to update all of the components of the data vector in parallel given the hidden states of all the experts. So Gibbs sampling can alternate between parallel updates of the hidden and visible variables (see Figure 1). To get an unbiased estimate of the gradient for the PoE, it is necessary for the Markov chain to converge to the equilibrium distribution.

Unfortunately, even if it is computationally feasible to approach the equilibrium distribution before taking samples, there is a second, serious difficulty. Samples from the equilibrium distribution generally have high variance since they come from all over the model's distribution. This high variance swamps the estimate of the derivative. Worse still, the variance in the samples depends on the parameters of the model. This variation in the variance causes the parameters to be repelled from regions of high variance even if the gradient is zero. To understand this subtle effect, consider a horizontal sheet of tin that is resonating in such a way that some parts have strong vertical oscillations and other parts are motionless. Sand scattered on the tin will accumulate in the motionless areas even though the time-averaged gradient is zero everywhere.

3 Learning by Minimizing CD

Maximizing the log likelihood of the data (averaged over the data distribution) is equivalent to minimizing the K-L divergence between the data distribution, P_0, and the equilibrium distribution over the visible variables,

P_θ^∞, that is produced by prolonged Gibbs sampling from the generative model,[4]

$$P^0 \parallel P_\theta^\infty = \sum_{\mathbf{d}} P^0(\mathbf{d}) \log P^0(\mathbf{d}) - \sum_{\mathbf{d}} P^0(\mathbf{d}) \log P_\theta^\infty(\mathbf{d})$$
$$= -H(P^0) - \langle \log P_\theta^\infty \rangle_{P^0}, \tag{3.1}$$

where $\parallel$ denotes a K-L divergence , the angle brackets denote expectations over the distribution specified as a subscript . P_0 does not depend on the parameters of the model, so $H(P^0)$ can be ignored during the optimization . Note that $P_\theta^\infty(\mathbf{d}) = p(\mathbf{d} \mid \boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n)$. Eq (2.2), averaged over the data distribution:

$$\left\langle \frac{\partial \log P_\theta^\infty(\mathbf{D})}{\partial \boldsymbol{\theta}_m} \right\rangle_{P^0} = \left\langle \frac{\partial \log f_{\theta_m}}{\partial \boldsymbol{\theta}_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_{\theta_m}}{\partial \boldsymbol{\theta}_m} \right\rangle_{P_\theta^\infty}, \tag{3.2}$$

where $\log f_{\theta_m}$ is a random variable that could be written as $\log f_m(\mathbf{D} \mid \boldsymbol{\theta}_m)$ with **D** itself being a random variable corresponding to the data. There is a simple and effective alternative to maximum likelihood learning that eliminates almost all of the computation required to get samples from the equilibrium distribution and also eliminates much of the variance that masks the gradient signal. This alternative approach involves optimizing a different objective function. Instead of just minimizing $P^0 \parallel P_\theta^\infty$, we minimize the difference between $P^0 \parallel P_\theta^\infty$ and $P_\theta^1 \parallel P_\theta^\infty$ where P_θ^1 is the distribution over the "one-step" reconstructions of the data vectors generated by one full step of Gibbs sampling (see Figure 1).

The intuitive motivation for using this "contrastive divergence" is that we would like the Markov chain that is implemented by Gibbs sampling to leave the initial distribution P^0 over the visible variables unaltered. Instead of running the chain to equilibrium and comparing the initial and final derivatives, we can simply run the chain for one full step and then update the parameters to reduce the tendency of the chain to wander away from the initial distribution on the first step. Because P_θ^1 is one step closer to the equilibrium distribution than P^0, we are guaranteed that $P^0 \parallel P_\theta^\infty$ exceeds $P_\theta^1 \parallel P_\theta^\infty$ unless $P^0 = P_\theta^1$, so the CD can never be negative.
Also, for Markov chains in which all transitions have nonzero probability,

$$P^0 = P_\theta^1 \implies P^0 = P_\theta^\infty,$$

because if the distribution does not change at all on the first step,

[4] P^0 is a natural way to denote the data distribution if we imagine starting a Markov chain at the data distribution at time 0.

it must already be at equilibrium, so the CD = 0 only if the model is perfect.[5]

Another way of understanding CD learning is to view it as a method of eliminating all the ways in which the PoE model would like to distort the true data. This is done by ensuring that, on average, the reconstruction is no more probable under the PoE model than the original data vector.

The mathematical motivation for the CD is that the intractable expectation over P_θ^∞ on the rhs of eq (3.2) cancels out:

$$-\frac{\partial}{\partial \boldsymbol{\theta}_m}(P^0 \parallel P_\theta^\infty - P_\theta^1 \parallel P_\theta^\infty) = \left\langle \frac{\partial \log f_{\theta_m}}{\partial \boldsymbol{\theta}_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_{\theta_m}}{\partial \boldsymbol{\theta}_m} \right\rangle_{P_\theta^1} + \frac{\partial P_\theta^1}{\partial \boldsymbol{\theta}_m} \frac{\partial (P_\theta^1 \parallel P_\theta^\infty)}{\partial P_\theta^1}. \tag{3.3}$$

If each expert is chosen to be tractable, it is possible to compute the exact values of the derivative of $\log f_m(\mathbf{d} \mid \boldsymbol{\theta}_m)$ for a data vector, $\mathbf{d}$. It is also straightforward to sample from P^0 and P_θ^1, so the first two terms on the rhs of equation 3.3 are tractable. By definition, the following procedure produces an unbiased sample from P_θ^1:

1. Pick a data vector, $\mathbf{d}$, from the distribution of the data P^0.
2. Compute, for each expert separately, the posterior probability distribution over its latent variables given the data vector, $\mathbf{d}$.
3. Pick a value for each latent variable from its posterior distribution.
4. Given the chosen values of all the latent variables, compute the conditional distribution over all the visible variables by multiplying together the conditional distributions specified by each expert and renormalizing.
5. Pick a value for each visible variable from the conditional distribution. These values constitute the reconstructed data vector, $\mathbf{d}^\wedge$.

The third term on the rhs of eq 3.3 represents the effect on $P_\theta^1 \parallel P_\theta^\infty$ of the change of the distribution of the one-step reconstructions

caused by a change in $\boldsymbol{\theta}_m$. It is problematic to compute, but extensive simulations (see section 10) show that it can safely be ignored because it is small

[5] It is obviously possible to make the contrastive divergence small by using a Markov chain that mixes very slowly, even if the data distribution is very far from the eventual equilibrium distribution. It is therefore important to ensure mixing by using techniques such as weight decay that ensure that every possible visible vector has a nonzero probability given the states of the hidden variables.

and seldom opposes the resultant of the other two terms. The parameters of the experts can therefore be adjusted in proportion to the approximate derivative of CD:

$$\Delta\boldsymbol{\theta}_m \propto \left\langle \frac{\partial \log f_{\theta_m}}{\partial \boldsymbol{\theta}_m} \right\rangle_{P^0} - \left\langle \frac{\partial \log f_{\theta_m}}{\partial \boldsymbol{\theta}_m} \right\rangle_{P^1_\theta}. \tag{3.4}$$

This works very well in practice even when a single reconstruction of each data vector is used in place of the full probability distribution over reconstructions. The difference in the derivatives of the data vectors and their reconstructions has some variance because the reconstruction procedure is stochastic. But when the PoE is modeling the data moderately well, the one-step reconstructions will be very similar to the data, so the variance will be very small. The close match between a data vector and its reconstruction reduces sampling variance in much the same way as the use of matched pairs for experimental and control conditions in a clinical trial. The low variance makes it feasible to perform on-line learning after each data vector is presented, though the simulations described in this article use mini-batch learning in which the parameter updates are based on the summed gradients measured on a rotating subset of the complete training set.[6]

There is an alternative justification for the learning algorithm in equation 3.4. In high-dimensional data sets, the data nearly always lie on, or close to, a much lower-dimensional, smoothly curved manifold. The PoE needs to find parameters that make a sharp ridge of log probability along the low-dimensional manifold. By starting with a point on the manifold and ensuring that the typical reconstructions from the latent variables of all the experts do not have significantly higher probability, the PoE ensures that the probability distribution has the right local structure. It is possible that the PoE will accidentally assign high probability to other distant and unvisited parts of the data space, but this is unlikely if the log probability surface is smooth and both its height and its local curvature are constrained at the data points. It is also possible to find and eliminate such points by performing prolonged Gibbs sampling without any data, but this is just a way of improving the learning and not, as in Boltzmann machine learning, an essential part of it.

4 A Simple Example

PoEs should work very well on data distributions that can be factorized into a product of lower-dimensional distributions. This is demonstrated in Figures 2 and 3. There are 15 "unigauss" experts, each of which is a

[6] Mini-batch learning makes better use of the ability of Matlab to vectorize across training examples.

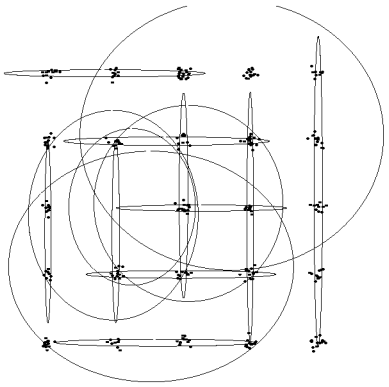


Figure 2: Each dot is a data point. The data have been fitted with a product of 15 experts. The ellipses show the one standard deviation contours of the gaussians in each expert. The experts are initialized with randomly located, circular gaussians that have about the same variance as the data. The five unneeded experts remain vague, but the mixing proportions, which determine the prior probability with which each of these unigauss experts selects its gaussian rather than its uniform, remain high.

mixture of a uniform distribution and a single axis-aligned gaussian. In the fitted model, each tight data cluster is represented by the intersection of two experts' gaussians, which are elongated along different axes. Using a conservative learning rate, the fitting required 2000 updates of the parameters. For each update of the parameters, the following computation is performed on every observed data vector:

1. Given the data, $\mathbf{d}$, calculate the posterior probability of selecting the gaussian rather than the uniform in each expert and compute the first term on the rhs of eq (3.4).
2. For each expert, stochastically select the gaussian or the uniform according to the posterior. Compute the normalized product of the selected gaussians, which is itself a gaussian, and sample from it to get a "reconstructed" vector in the data space. To avoid problems, there is one special expert that is constrained to always pick its gaussian.
3. Compute the negative term in eq (3.4) using the reconstructed data vector.

5 Learning a Population Code

A PoE can also be a very effective model when each expert is quite broadly tuned on every dimension and precision is obtained by the intersection of a

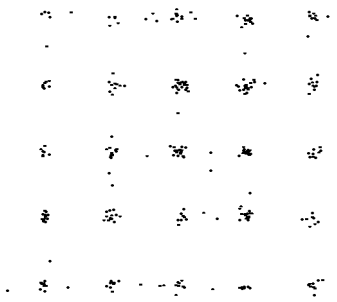


Figure 3: Three hundred data points generated by prolonged Gibbs sampling from the 15 experts fitted in Figure 1. The Gibbs sampling started from a random point in the range of the data and used 25 parallel iterations with annealing. Notice that the fitted model generates data at the grid point that is missing in the real data.

large number of experts. Figure 4 shows what happens when the contrastive divergence learning algorithm is used to fit 40 unigauss experts to 100-dimensional synthetic images that each contain one edge. The edges varied in their orientation, position, and intensities on each side of the edge. The intensity profile across the edge was a sigmoid. Each expert also learned a variance for each pixel, and although these variances varied, individual experts did not specialize in a small subset of the dimensions. Given an image, about half of the experts have a high probability of picking their gaussian rather than their uniform. The products of the chosen gaussians are excellent reconstructions of the image. The experts at the top of Figure 4 look like edge detectors in various orientations, positions, and polarities. Many of the experts farther down have even symmetry and are used to locate one end of an edge. They each work for two different sets of edges that have opposite polarities and different positions.

6 Initializing the Experts

One way to initialize a PoE is to train each expert separately, forcing the experts to differ by giving them different or differently weighted training cases or by training them on different subsets of the data dimensions, or by using different model classes for the different experts. Once each expert has been initialized separately, the individual probability distributions need to be raised to a fractional power to create the initial PoE.

Separate initialization of the experts seems like a sensible idea, but simulations indicate that the PoE is far more likely to become trapped in poor local optima if the experts are allowed to specialize separately. Better solutions are obtained by simply initializing the experts randomly with very vague distributions and using the learning rule in eq (3.4).

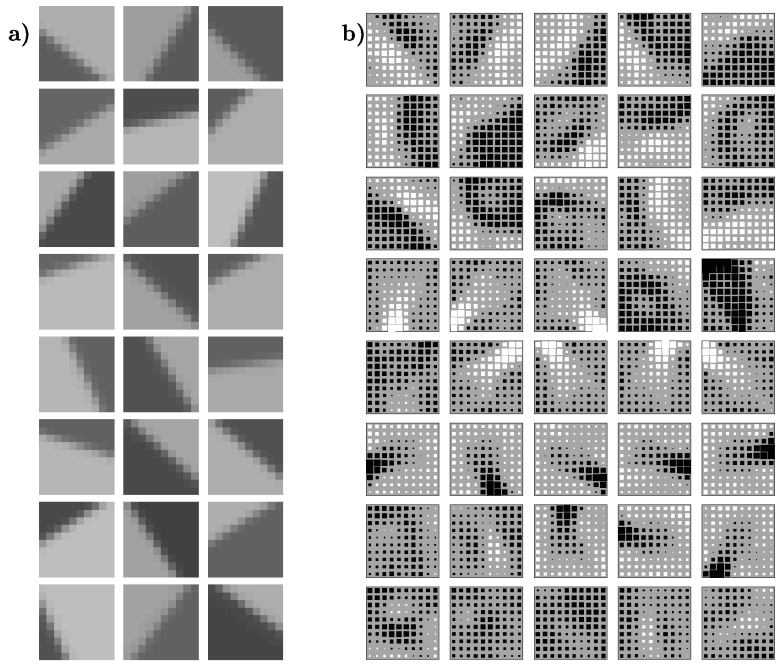


Figure 4: (a) Some 10×10 images that each contain a single intensity edge. The location, orientation, and contrast of the edge all vary. (b) The means of all the 100-dimensional gaussians in a product of 40 experts, each of which is a mixture of a gaussian and a uniform. The PoE was fitted to 500 images of the type shown on the left. The experts have been ordered by hand so that qualitatively similar experts are adjacent.

7 PoEs and BMs

The BM learning algorithm (Hinton & Sejnowski , 1986) is theoretically elegant and easy to implement in hardware but very slow in networks with interconnected hidden units because of the variance problems described in section 2. Smolensky (1986) introduced a restricted type of BM with one visible layer, one hidden layer, and no intralayer connections . Freund and Haussler (1992) realized that in this RBM , the probability of generating a visible vector is proportional to the product of the probabilities that the visible vector would be generated by each of the hidden units acting alone. An RBM is therefore a PoE with one expert per hidden unit .[7] When the hidden unit

[7] BMs and PoEs are very different classes of PGM and the intersection of the two classes is RBMs.

of an expert is off, it specifies a separable probability distribution in which each visible unit is equally likely to be on or off. When the hidden unit is on, it specifies a different factorial distribution by using the weight on its connection to each visible unit to specify the log odds that the visible unit is on. Multiplying together the distributions over the visible states specified by different experts is achieved by simply adding the log odds. Exact inference of the hidden states given the visible data is tractable in an RBM because the states of the hidden units are conditionally independent given the data.

The learning algorithm given by eq (2.2) is exactly equivalent to the standard Boltzmann learning algo for an RBM. Consider the derivative of the log probability of the data wrt the weight w_{ij} between a visible unit i and a hidden unit j. The first term on the rhs of eq(2.2) is

$$\frac{\partial \log f_j(\mathbf{d} \mid \mathbf{w}_j)}{\partial w_{ij}} = \langle s_i s_j \rangle_{\mathbf{d}} - \langle s_i s_j \rangle_{P_\theta^\infty(j)}, \tag{7.1}$$

where $\mathbf{w}_j$ is the vector of weights connecting hidden unit j to the visible units, $\langle s_i s_j \rangle_{\mathbf{d}}$ is the expected value of $s_i s_j$ when $\mathbf{d}$ is clamped on the visible units and s_j is sampled from its posterior distribution given $\mathbf{d}$, and $\langle s_i s_j \rangle_{P_\theta^\infty(j)}$ is the expected value of $s_i s_j$ when alternating Gibbs sampling of the hidden and visible units is iterated to get samples from the equilibrium distribution in a network whose only hidden unit is j.

The second term on the rhs of eq (2.2) is:

$$\sum_{\mathbf{c}} p(\mathbf{c} \mid \mathbf{w}) \frac{\partial \log f_j(\mathbf{c} \mid \mathbf{w}_j)}{\partial w_{ij}} = \langle s_i s_j \rangle_{P_\theta^\infty} - \langle s_i s_j \rangle_{P_\theta^\infty(j)}, \tag{7.2}$$

where $\mathbf{w}$ is all of the weights in the RBM and $\langle s_i s_j \rangle_{P_\theta^\infty}$ is the expected value of $s_i s_j$ when alternating Gibbs sampling of all the hidden and all the visible units is iterated to get samples from the equilibrium distribution of the RBM.
eq (7.1) - eq (7.2) ; taking expectations over the distribution of data == >

$$\left\langle \frac{\partial \log P_\theta^\infty}{\partial w_{ij}} \right\rangle_{P^0} = -\frac{\partial (P^0 \parallel P_\theta^\infty)}{\partial w_{ij}} = \langle s_i s_j \rangle_{P^0} - \langle s_i s_j \rangle_{P_\theta^\infty}. \tag{7.3}$$

The time required to approach equilibrium and the high sampling variance in $\langle s_i s_j \rangle_{P_\theta^\infty}$ make learning difficult. It is much more effective to use the approximate gradient of the contrastive divergence. For an RBM, this approximate gradient is particularly easy to compute:

$$-\frac{\partial}{\partial w_{ij}} (P^0 \parallel P_\theta^\infty - P_\theta^1 \parallel P_\theta^\infty) \approx \langle s_i s_j \rangle_{P^0} - \langle s_i s_j \rangle_{P_\theta^1}, \tag{7.4}$$

where $\langle s_i s_j \rangle_{P_\theta^1}$ is the expected value of $s_i s_j$ when one-step reconstructions are clamped on the visible units and s_j is sampled from its posterior distribution given the reconstruction (see Figure 1).

8 Learning the Features of Handwritten Digits

When presented with real high-dimensional data, a restricted Boltzmann machine trained to minimize the contrastive divergence using equation 7.4 should learn a set of probabilistic binary features that model the data well. To test this conjecture, an RBM with 500 hidden units and 256 visible units was trained on 8000 16×16 real-valued images of handwritten digits from all 10 classes. The images, from the "br" training set on the USPS Cedar ROM1, were normalized in width and height, but they were highly variable in style. The pixel intensities were normalized to lie between 0 and 1 so that they could be treated as probabilities, and equation 7.4 was modified to use probabilities in place of stochastic binary values for both the data and the one-step reconstructions:

$$-\frac{\partial}{\partial w_{ij}}(P^0 \parallel P_\theta^\infty - P_\theta^1 \parallel P_\theta^\infty) \approx \langle p_i p_j \rangle_{P^0} - \langle p_i p_j \rangle_{P_\theta^1}. \tag{8.1}$$

Stochastically chosen binary states of the hidden units were still used for computing the probabilities of the reconstructed pixels, but instead of picking binary states for the pixels from those probabilities, the probabilities themselves were used as the reconstructed data vector.

It takes 10 hours in Matlab 5.3 on a 500 MHz pentium II workstation to perform 658 epochs of learning. This is much faster than standard Boltzmann machine learning, comparable with the wake-sleep algorithm (Hinton, Dayan, Frey, & Neal, 1995) and considerably slower than using EM to fit a mixture model with the same number of parameters. In each epoch, the weights were updated 80 times using the approximate gradient of the contrastive divergence computed on mini-batches of size 100 that contained 10 exemplars of each digit class. The learning rate was set empirically to be about one-quarter of the rate that caused divergent oscillations in the parameters. To improve the learning speed further, a momentum method was used. After the first 10 epochs, the parameter updates specified by equation 8.1 were supplemented by adding 0.9 times the previous update.

The PoE learned localized features whose binary states yielded almost perfect reconstructions. For each image, about one-third of the features were turned on. Some of the learned features had on-center off-surround receptive fields or vice versa, some looked like pieces of stroke, and some looked like Gabor filters or wavelets. The weights of 100 of the hidden units, selected at random, are shown in Figure 5.

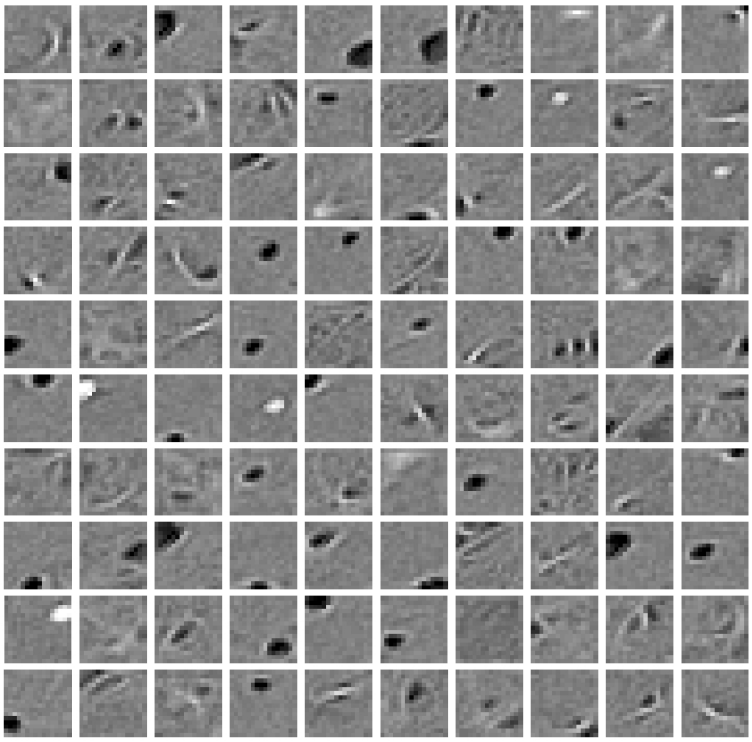


Figure 5: The receptive fields of a randomly selected subset of the 500 hidden units in a PoE that was trained on 8000 images of digits with equal numbers from each class. Each block shows the 256 learned weights connecting a hidden unit to the pixels. The scale goes from $+2$ (white) to -2 (black).

9 Using Learned Models of Handwritten Digits for Discrimination

An attractive aspect of PoEs is that it is easy to compute the numerator in equation 2.1, so it is easy to compute the log probability of a data vector up to an additive constant, $\log Z$, which is the log of the denominator in equation 2.1. Unfortunately, it is very hard to compute this additive constant. This does not matter if we want to compare only the probabilities of two different data vectors under the PoE, but it makes it difficult to evaluate the model learned by a PoE, because the obvious way to measure the success of learning is to sum the log probabilities that the PoE assigns to test data vectors drawn from the same distribution as the training data but not used during training.

For a novelty detection task, it would be possible to train a PoE on "normal" data and then to learn a single scalar threshold value for the unnormal-

ized log probabilities in order to optimize discrimination between "normal" data and a few abnormal cases. This makes good use of the ability of a PoE to perform unsupervised learning. It is equivalent to naive Bayesian classification using a very primitive separate model of abnormal cases that simply returns the same learned log probability for all abnormal cases.

An alternative way to evaluate the learning procedure is to learn two different PoEs on different data sets such as images of the digit 2 and images of the digit 3. After learning, a test image, $\mathbf{t}$, is presented to PoE_2 and PoE_3, and they compute $\log p(\mathbf{t} \mid \theta_2) + \log Z_2$ and $\log p(\mathbf{t} \mid \theta_3) + \log Z_3$, respectively. If the difference between $\log Z_2$ and $\log Z_3$ is known, it is easy to pick the most likely class of the test image, and since this difference is only a single number, it is quite easy to estimate it discriminatively using a set of validation images whose labels are known. If discriminative performance is the only goal and all the relevant classes are known in advance, it is probably sensible to train all the parameters of a system discriminatively. In this article, however, discriminative performance on the digits is used simply as a way of demonstrating that unsupervised PoEs learn very good models of the individual digit classes.

Figure 6 shows features learned by a PoE that contains a layer of 100 hidden units and is trained on 800 images of the digit 2. Figure 7 shows some previously unseen test images of 2s and their one-step reconstructions from the binary activities of the PoE trained on 2s and from an identical PoE trained on 3s.

Figure 8 shows the unnormalized log probability scores of some training and test images under a model trained on 825 images of the digit 4 and a model trained on 825 images of the digit 6. Unfortunately, the official test set for the USPS digits violates the standard assumption that test data should be drawn from the same distribution as the training data, so here the test images were drawn from the unused portion of the official training set. Even for the previously unseen test images, the scores under the two models allow perfect discrimination. To achieve this excellent separation, it was necessary to use models with two hidden layers and average the scores from two separately trained models of each digit class. For each digit class, one model had 200 units in its first hidden layer and 100 in its second hidden layer. The other model had 100 in the first hidden layer and 50 in the second. The units in the first hidden layer were trained without regard to the second hidden layer. After training the first hidden layer, the second hidden layer was trained using the probabilities of feature activation in the first hidden layer as the data. For testing, the scores provided by the two hidden layers were simply added together. Omitting the score provided by the second hidden layer leads to considerably worse separation of the digit classes on test data.

Figure 9 shows the unnormalized log probability scores for previously unseen test images of 7s and 9s, which are the most difficult classes to discriminate. Discrimination is not perfect, but it is encouraging that all

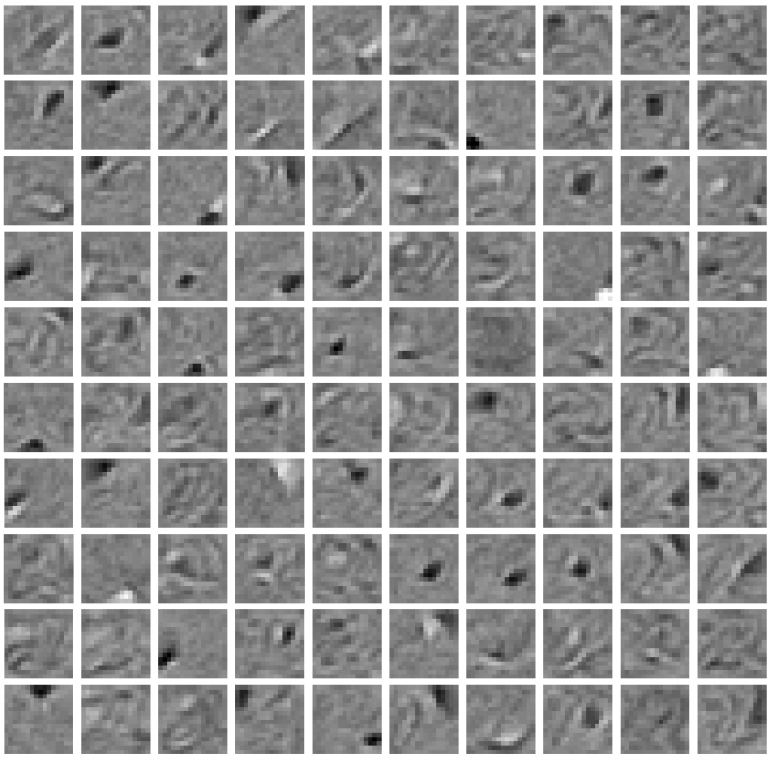


Figure 6: The weights learned by 100 hidden units trained on 16×16 images of the digit 2. The scale goes from $+3$ (white) to -3 (black). Note that the fields are mostly quite local. A local feature like the one in column 1, row 7 looks like an edge detector, but it is best understood as a local deformation of a template. Suppose that all the other active features create an image of a 2 that differs from the data in having a large loop whose top falls on the black part of the receptive field. By turning on this feature, the top of the loop can be removed and replaced by a line segment that is a little lower in the image.

of the errors are close to the decision boundary, so there are no confident misclassifications.

9.1 Dealing with Multiple Classes. If there are 10 different PoEs for the 10 digit classes, it is slightly less obvious how to use the 10 unnormalized scores of a test image for discrimination. One possibility is to use a validation set to train a logistic discrimination network that takes the unnormalized log probabilities given by the PoEs and converts them into a probability distribution across the 10 labels. Figure 10 shows the weights in a logistic

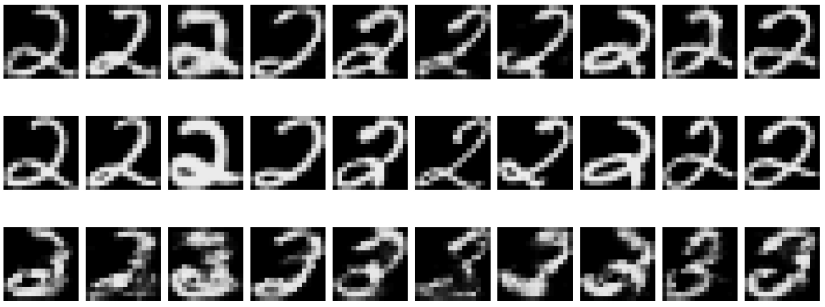
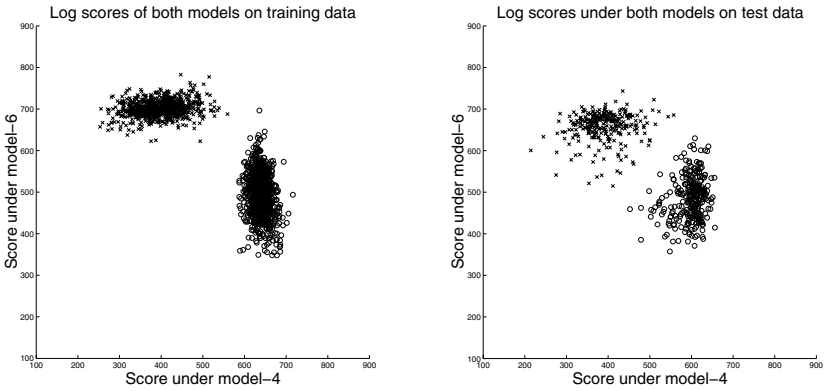


Figure 7: The center row is previously unseen images of 2s. (Top) Pixel probabilities when the image is reconstructed from the binary activities of 100 feature detectors that have been trained on 2s. (Bottom) Pixel probabilities for reconstructions from the binary states of 100 feature detectors trained on 3s.



(a) (b)

Figure 8: (a) The unnormalized log probability scores of the training images of the digits 4 and 6 under the learned PoEs for 4 and 6. (b) The log probability scores for previously unseen test images of 4s and 6s. Note the good separation of the two classes.

discrimination network that is trained after fitting 10 PoE models to the 10 separate digit classes. In order to see whether the second hidden layers were providing useful discriminative information, each PoE provided two scores. The first score was the unnormalized log probability of the pixel intensities under a PoE model that consisted of the units in the first hidden layer. The second score was the unnormalized log probability of the probabilities of activation of the first layer of hidden units under a PoE model that consisted of the units in the second hidden layer. The weights in Figure 10 show that the second layer of hidden units provides useful additional information.

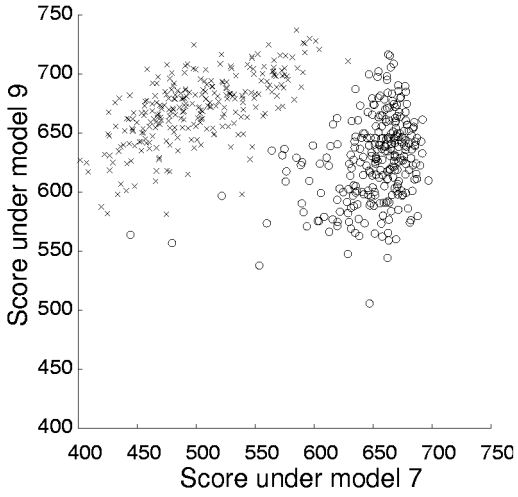


Figure 9: The unnormalized log probability scores of the previously unseen test images of 7s and 9s. Although the classes are not linearly separable, all the errors are close to the best separating line, so there are no confident errors.

Presumably this is because it captures the way in which features represented in the first hidden layer are correlated. A second-level model should be able to assign high scores to the vectors of hidden activities that are typical of the first-level 2 model when it is given images of 2s and low scores to the hidden activities of the first-level 2 model when it is given images that contain combinations of features that are not normally present at the same time in a 2.

The error rate is 1.1% which compares very favorably with the 5.1% error rate of a simple nearest-neighbor classifier on the same training and test sets and is about the same as the very best classifier based on elastic models of the digits (Revow, Williams, & Hinton, 1996). If 7% rejects are allowed (by choosing an appropriate threshold for the probability level of the most probable class), there are no errors on the 2750 test images.

Several different network architectures were tried for the digit-specific PoEs, and the results reported are for the architecture that did best on the test data. Although this is typical of research on learning algorithms, the fact that test data were used for model selection means that the reported results are a biased estimate of the performance on genuinely unseen test images. Mayraz and Hinton (2001) report good comparative results for the larger MNIST database, and they were careful to do all the model selection using subsets of the training data so that the official test data were used only to measure the final error rate.

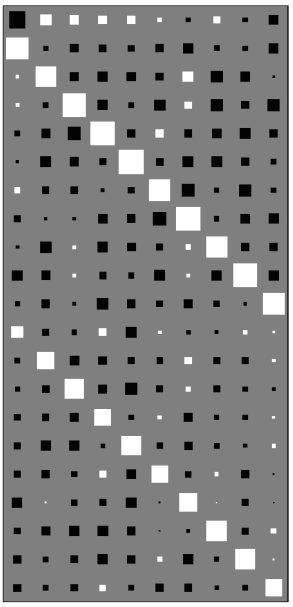


Figure 10: The weights learned by doing multinomial logistic discrimination on the training data with the labels as outputs and the unnormalized log probability scores from the trained, digit-specific PoEs as inputs. Each column corresponds to a digit class, starting with digit 1. The top row is the biases for the classes. The next 10 rows are the weights assigned to the scores that represent the log probability of the pixels under the model learned by the first hidden layer of each PoE. The last 10 rows are the weights assigned to the scores that represent the log probabilities of the probabilities on the first hidden layer under the model learned by the second hidden layer. Note that although the weights in the last 10 rows are smaller, they are still quite large, which shows that the scores from the second hidden layers provide useful, additional discriminative information. Note also that the scores produced by the 2 model provide useful evidence in favor of 7s.

10 How Good Is the Approximation?

The fact that the learning procedure in equation 3.4 gives good results in the simulations described in sections 4, 5, and 9 suggests that it is safe to ignore the final term in the right-hand side of equation 3.3 that comes from the change in the distribution P^1_θ.

To get an idea of the relative magnitude of the term that is being ignored, extensive simulations were performed using restricted Boltzmann machines with small numbers of visible and hidden units. By performing computations that are exponential in the number of hidden units and the

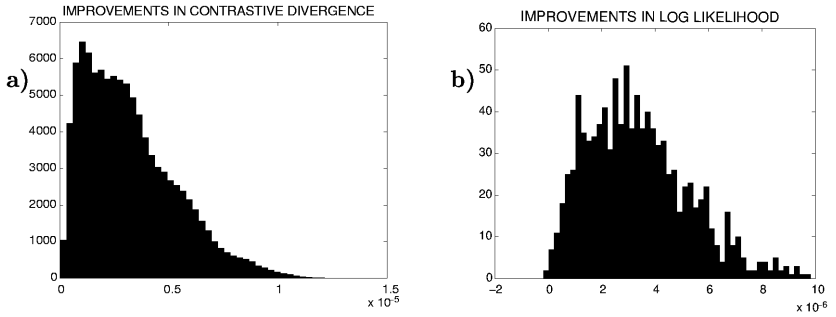


Figure 11: (a) A histogram of the improvements in the contrastive divergence as a result of using equation 7.4 to perform one update of the weights in each of 10^5 networks. The expected values on the right-hand side of equation 7.4 were computed exactly. The networks had eight visible and four hidden units. The initial weights were randomly chosen from a gaussian with mean 0 and standard deviation 20. The training data were chosen at random. (b) The improvements in the log likelihood of the data for 1000 networks chosen in exactly the same way as in Figure 11a. Note that the log likelihood decreased in two cases. The changes in the log likelihood are the same as the changes in $P^0 \parallel P^\infty_\theta$ but with a sign reversal.

number of visible units, it is possible to compute the exact values of $\langle s_i s_j \rangle_{P^0}$ and $\langle s_i s_j \rangle_{P^1_\theta}$. It is also possible to measure what happens to $P^0 \parallel P^\infty_\theta - P^1_\theta \parallel P^\infty_\theta$ when the approximation in equation 7.4 is used to update the weights by an amount that is large compared with the numerical precision of the machine but small compared with the curvature of the CD.

The RBMs used for these simulations had random training data and random weights. They did not have biases on the visible or hidden units. The main result can be summarized as follows: For an individual weight, the rhs of eq (7.4), summed over all training cases, occasionally differs in sign from the lhs. But for networks containing more than two units in each layer it is almost certain that a parallel update of all the weights based on the rhs of eq (7.4) will improve the CD. In other words, when we average over the train- ing data, the vector of parameter updates given by the rhs is almost certain to have a positive cosine with the true gradient defined by the left-hand side. Figure 11a is a histogram of the improvements in the CD when eq (7.4) was used to perform one parallel weight update in each of 100,000 networks. The networks contained eight visible and four hidden units, and their weights were chosen from a gaussian distribution with mean zero and standard deviation 20. With smaller weights or larger networks, the approximation in equation 7.4 is even better.

Figure 11b shows that the learning procedure does not always improve the log likelihood of the training data, though it has a strong tendency to do so. Note that only 1000 networks were used for this histogram.

Figure 12 compares the contributions to the gradient of the contrastive divergence made by the right-hand side of equation 7.4 and by the term that is being ignored. The vector of weight updates given by equation 7.4 makes the contrastive divergence worse if the dots in Figure 12 are above the diagonal line, so it is clear that in these networks, the approximation in equation 7.4 is quite safe. Intuitively, we expect P^1_θ to lie between P^0 and P^∞_θ, so when the parameters are changed to move P^∞_θ closer to P^0, the changes should also move P^1_θ toward P^0 and away from the previous position of P^∞_θ. The ignored changes in P^1_θ should cause an increase in $P^1_\theta \parallel P^\infty_\theta$ and thus an improvement in the contrastive divergence (which is what Figure 12 shows).

It is tempting to interpret the learning rule in equation 3.4 as approximate optimization of the contrastive log likelihood:

$$\langle \log P^\infty_\theta \rangle_{P^0} - \langle \log P^\infty_\theta \rangle_{P^1_\theta}.$$

Unfortunately, the contrastive log likelihood can achieve its maximum value of 0 by simply making all possible vectors in the data space equally probable. The CD differs from the contrastive log like- lihood by including the entropies of the distributions P^0 and $P_\theta{}^1$ (see equation 8.1), and so the high entropy of $P_\theta{}^1$ rules out the solution in which all possible data vectors are equiprobable.

11 Other Types of Expert

Binary stochastic pixels are not unreasonable for modeling preprocessed images of handwritten digits in which ink and background are represented as 1 and 0. In real images, however, there is typically very high mutual information between the real-valued intensity of one pixel and the real-valued intensities of its neighbors. This cannot be captured by models that use binary stochastic pixels because a binary pixel can never have more than 1 bit of mutual information with anything. It is possible to use "multinomial" pixels that have n discrete values. This is a clumsy solution for images because it fails to capture the continuity and one-dimensionality of pixel intensity, though it may be useful for other types of data. A better approach is to imagine replicating each visible unit so that a pixel corresponds to a whole set of binary visible units that all have identical weights to the hidden units. The number of active units in the set can then approximate a real-valued intensity. During reconstruction, the number of active units will be binomially distributed, and because all the replicas have the same weights, the single probability that controls this binomial distribution needs to be computed only once. The same trick can be used to allow replicated hidden units to

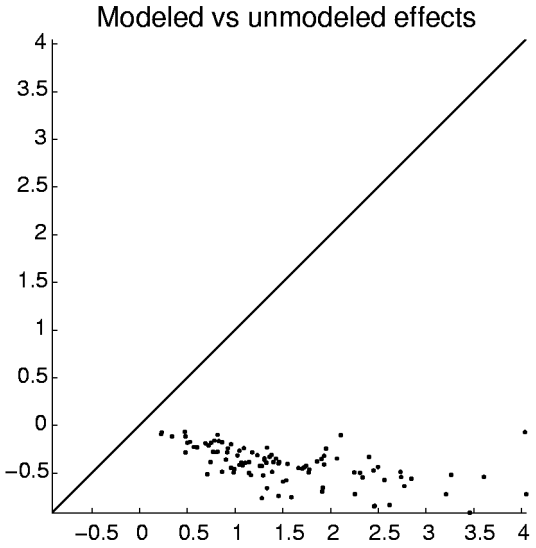


Figure 12: A scatter plot that shows the relative magnitudes of the modeled and unmodeled effects of a parallel weight update on the contrastive divergence. The 100 networks used for this figure have 10 visible and 10 hidden units, and their weights are drawn from a zero-mean gaussian with a standard deviation of 10. The horizontal axis shows $(P^0 \parallel P^\infty_{\theta(old)} - P^1_{\theta(old)} \parallel P^\infty_{\theta(old)}) - (P^0 \parallel P^\infty_{\theta(new)} - P^1_{\theta(old)} \parallel P^\infty_{\theta(new)})$ where *old* and *new* denote the distributions before and after the weight update. This modeled reduction in the contrastive divergence differs from the true reduction because it ignores the fact that changing the weights changes the distribution of the one-step reconstructions. The increase in the contrastive divergence due to the ignored term, $P^1_{\theta(old)} \parallel P^\infty_{\theta(new)} - P^1_{\theta(new)} \parallel P^\infty_{\theta(new)}$, is plotted on the vertical axis. Points above the diagonal line would correspond to cases in which the unmodeled increase outweighed the modeled decrease, so that the net effect was to make the contrastive divergence worse. Note that the unmodeled effects almost always cause an additional improvement in the contrastive divergence rather than being in conflict with the modeled effects.

approximate real values using binomially distributed integer states. A set of replicated units can be viewed as a computationally cheap approximation to units whose weights actually differ, or it can be viewed as a stationary approximation to the behavior of a single unit over time, in which case the number of active replicas is a firing rate. Teh and Hinton (2001) have shown that this type of rate coding can be quite effective for modeling real-valued images of faces, provided the images are normalized.

An alternative to replicating hidden units is to use "unifactor" experts that each consist of a mixture of a uniform distribution and a factor analyzer

with just one factor. Each expert has a binary latent variable that specifies whether to use the uniform or the factor analyzer and a real-valued latent variable that specifies the value of the factor (if it is being used). The factor analyzer has three sets of parameters: a vector of factor loadings that specify the direction of the factor in image space, a vector of means in image space, and a vector of variances in image space.[8] Experts of this type have been explored in the context of directed acyclic graphs (Hinton, Sallans, & Ghahramani, 1998), but they should work better in a PoE.

An alternative to using a large number of relatively simple experts is to make each expert as complicated as possible, while retaining the ability to compute the exact derivative of the log likelihood of the data with respect to the parameters of an expert. In modeling static images, for example, each expert could be a mixture of many axis-aligned gaussians. Some experts might focus on one region of an image by using very high variances for pixels outside that region. But so long as the regions modeled by different experts overlap, it should be possible to avoid block boundary artifacts.

11.1 Products of HMMs. HMMs are of great practical value in modeling sequences of discrete symbols or sequences of real-valued vectors because there is an efficient algorithm for updating the parameters of the HMM to improve the log likelihood of a set of observed sequences. HMMs are, however, quite limited in their generative power because the only way that the portion of a string generated up to time t can constrain the portion of the string generated after time t is by the discrete hidden state of the generator at time t. So if the first part of a string has, on average, n bits of mutual information with the rest of the string, the HMM must have 2^n hidden states to convey this mutual information by its choice of hidden state. This exponential inefficiency can be overcome by using a product of HMMs as a generator. During generation, each HMM gets to pick a hidden state at each time so the mutual information between the past and the future can be linear in the number of HMMs. It is therefore exponentially more efficient to have many small HMMs than one big one. However, to apply the standard forward-backward algorithm to a product of HMMs, it is necessary to take the cross-product of their state-spaces, which throws away the exponential win. For products of HMMs to be of practical significance, it is necessary to find an efficient way to train them.

Andrew Brown (Brown & Hinton, 2001) has shown that the learning algorithm in equation 3.4 works well. The forward-backward algorithm is used to get the gradient of the log likelihood of an observed or reconstructed

[8] The last two sets of parameters are exactly equivalent to the parameters of a "unigauss" expert introduced in section 4, so a "unigauss" expert can be considered to be a mixture of a uniform with a factor analyzer that has no factors.

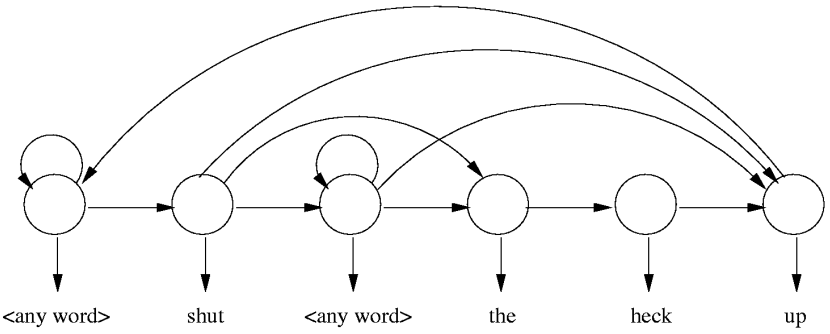


Figure 13: A HMM. The first and third nodes have output dis- tributions that are uniform across all words . If the first node has a high transition probability to itself, most strings of English words are given the same low prob- ability by this expert. Strings that contain the word *shut* followed directly or indirectly by the word *up* have higher probability under this expert.

sequence with respect to the parameters of an individual expert. The one-step reconstruction of a sequence is generated as follows:

1. Given an observed sequence, use the forward-backward algorithm in each expert separately to calculate the posterior probability distribution over paths through the hidden states.
2. For each expert, stochastically select a hidden path from the posterior given the observed sequence.
3. At each time step, select an output symbol or output vector from the product of the output distributions specified by the selected hidden state of each HMM.

If more realistic products of HMMs can be trained successfully by minimizing the contrastive divergence, they should be better than single HMMs for many different kinds of sequential data. Consider, for example, the HMM shown in Figure 13. This expert concisely captures a nonlocal regularity. A single HMM that must also model all the other regularities in strings of English words could not capture this regularity efficiently because it could not afford to devote its entire memory capacity to remembering whether the word *shut* had already occurred in the string.

12 Discussion

There have been previous attempts to learn representations by adjusting parameters to cancel out the effects of brief iteration in a recurrent network (Hinton & McClelland, 1988; O'Reilly, 1996; Seung, 1998), but these were

not formulated using a stochastic generative model and an appropriate objective function.

Minimizing CD has an unexpected similarity to the learning algorithm proposed by Winston (1975). Winston's program compared arches made of blocks with "near misses" supplied by a teacher, and it used the differences in its representations of the correct and incorrect arches to decide which aspects of its representation were relevant. By using a stochastic generative model , we can dispense with the teacher , but it is still the differences between the real data and the near misses generated by the model that drive the learning of the significant features.

12.1 Logarithmic Opinion Pools. The idea of combining the opinions of multiple different expert models by using a weighted average in the log probability domain is far from new (Genest & Zidek, 1986; Heskes, 1998), but research has focused on how to find the best weights for combining experts that have already been learned or programmed separately (Berger, Della Pietra, & Della Pietra, 1996) rather than training the experts cooperatively. The geometric mean of a set of probability distributions has the attractive property that its K-L divergence from the true distribution, P, is smaller than the average of the K-L divergences of the individual distributions, Q:

$$KL\left(P \parallel \frac{\Pi_m Q_m^{w_m}}{Z}\right) \leq \sum_m w_m KL(P \parallel Q_m) \tag{12.1}$$

where the w_m are nonnegative and sum to 1, and $Z = \sum_{\mathbf{c}} \Pi_m Q_m^{w_m}(\mathbf{c})$. When all of the individual models are identical, $Z = 1$. Otherwise, Z is less than one, and the difference between the two sides of equation 12.1 is $\log(1/Z)$. This makes it clear that the benefit of combining very different experts comes from the fact that they make Z small, especially in parts of the data space that do not contain data.

It is tempting to augment PoEs by giving each expert, m, an additional adaptive parameter, w_m, that scales its log probabilities. However, this makes inference much more difficult (Yee-Whye Teh, personal communication, 2000). Consider, for example, an expert with $w_m = 100$. This is equivalent to having 100 copies of an expert but with their latent states all tied together, and this tying affects the inference. It is easier to fix $w_m = 1$ and allow the PoE learning algorithm to determine the appropriate sharpness of the expert.

12.2 Relationship to Boosting. Within the supervised learning literature, methods such as bagging or boosting attempt to make "experts" different from one another by using different, or differently weighted, training data. After learning, these methods use the weighted average of the outputs of the individual experts. The way in which the outputs are combined

is appropriate for a conditional PoE model in which the output of an individual expert is the mean of a gaussian, its weight is the inverse variance of the gaussian, and the combined output is the mean of the product of all the individual gaussians. Zemel and Pitassi (2001) have recently shown that this view allows a version of boosting to be derived as a greedy method of optimizing a sensible objective function.

The fact that boosting works well for supervised learning suggests that it should be tried for unsupervised learning. In boosting, the experts are learned sequentially, and each new expert is fitted to data that have been reweighted so that data vectors that are modeled well by the existing experts receive very little weight and hence make very little contribution to the gradient for the new expert. A PoE fits all the experts at once, but it achieves an effect similar to reweighting by subtracting the gradient of the reconstructed data so that there is very little learning on data that are already modeled well. One big advantage of a PoE over boosting is that a PoE can focus the learning on the parts of a training vector that have high reconstruction errors. This allows PoEs to learn local features. With boosting, an entire training case receives a single scalar weight rather than a vector of reconstruction errors, so when boosting is used for unsupervised learning, it does not so easily discover local features.

12.3 Comparison with DAG Models. Inference in a PoE is trivial because the experts are individually tractable and the product formulation ensures that the hidden states of different experts are conditionally independent given the data .[9] This makes them relevant as models of biological perceptual systems , which must be able to do inference very rapidly . Alternative approaches based on DAG models (Neal, 1992) suffer from the " explaining-away" phenomenon . When such graphical models are densely connected , exact inference is intractable , so it is necessary to resort to clever but implausibly slow iterative techniques for approximate inference (Saul & Jordan , 2000) or to use crude approximations that ignore explaining away during inference and rely on the learning algorithm to find representations for which the shoddy inference technique is not too damaging (Hinton et al., 1995).

Unfortunately, the ease of inference in PoEs is balanced by the difficulty of generating fantasy data from the model. This can be done trivially in one ancestral pass in a directed acyclic graphical model but requires an iterative procedure such as Gibbs sampling in a PoE. If, however, equation 3.4 is used for learning, the difficulty of generating samples from the model is not a major problem.

In addition to the ease of inference that results from the conditional independence of the experts given the data, PoEs have a more subtle ad-

[9] This ceases to be true when there are missing data.

vantage over generative models that work by first choosing values for the latent variables and then generating a data vector from these latent values. If such a model has a single hidden layer and the latent variables have independent prior distributions, there will be a strong tendency for the posterior values of the latent variables in a well-fitted model to reflect the marginal independence that occurs when the model generates data.[10] For this reason, there has been little success with attempts to learn such generative models one hidden layer at a time in a greedy, bottom-up way. With PoEs, however, even though the experts have independent priors, the latent variables in different experts will be marginally dependent: they can be highly correlated across cases even for fantasy data generated by the PoE itself. So after the first hidden layer has been learned greedily, there may still be lots of statistical structure in the latent variables for the second hidden layer to capture. There is therefore some hope that an entirely unsupervised network could learn to extract a hierarchy of representations from patches of real images, but progress toward this goal requires an effective way of dealing with real-valued data. The types of expert that have been investigated so far have not performed as well as independent component analysis at extracting features like edges from natural images.

The most attractive property of a set of orthogonal basis functions is that it is possible to compute the coefficient on each basis function separately without worrying about the coefficients on other basis functions. A PoE retains this attractive property while allowing nonorthogonal experts and a nonlinear generative model.

12.4 Lateral Connections Between Hidden Units. The ease of inferring the states of the hidden units (latent variables) given the data is a major advantage of PoEs. This advantage appears to be lost if the latent variables of different experts are allowed to interact directly. If, for example, there are direct, symmetrical connections between the hidden units in a restricted Boltzmann machine, it becomes intractable to infer the exact distribution over hidden states when given the visible states. However, preliminary simulations show that the contrastive divergence idea can still be applied successfully using a damped mean-field method to find an approximating distribution over the hidden units for both the data vector and its reconstruction. The learning procedure then has the following steps for each data vector:

[10] This is easy to understand from a coding perspective in which the data are communicated by first specifying the states of the latent variables under an independent prior and then specifying the data given the latent states. If the latent states are not marginally independent, this coding scheme is inefficient, so pressure toward coding efficiency creates pressure toward independence.

1. With the data vector clamped on the visible units, compute the total bottom-up input, x_j, to each hidden unit, j, and initialize it to have real-valued state, $q_j^0 = \sigma(x_j)$, where $\sigma(.)$ is the logistic function.
2. Perform 10 damped mean-field iterations in which each hidden unit computes its total lateral input y_j^t at time t,

$$y_j^t = \sum_k w_{kj} q_k^{t-1},$$

and then adjusts its state, q_j^t,

$$q_j^t = 0.5 q_j^{t-1} + 0.5\sigma(x_j + y_j^t).$$

Measure $s_i q_j^{10}$ for each pair of a visible unit, i, and a hidden unit, j. Also measure $q_j^{10} q_k^{10}$ for all pairs of hidden units, j, k.

3. Pick a final binary state for each hidden unit, j, according to q_j^{10}, and use these binary states to generate a reconstructed data vector.
4. Perform one more damped mean-field iteration on the hidden units starting from the final state found with the data, but using the bottom-up input from the reconstruction rather than the data. Measure the same statistics as above.
5. Update every weight in proportion to the difference in the measured statistics in steps 2 and 4.

It is necessary for the mean-field iterations to converge for this procedure to work, which may require limiting or decaying the weights on the lateral connections. The usual worry—that the mean-field approximation is hopeless for multimodal distributions—may not be relevant for learning a perceptual system, since it makes sense to avoid learning models of the world in which there are multiple, very different interpretations of the same sensory input. CD learning finesses the problem of finding the highly multimodal equilibrium distribution that results if both visible and hidden units are unclamped . It does this by assuming that the observed data cover all of the different modes with roughly the right frequency for each mode.

The fact that the initial simulations work is very promising since networks with multiple hidden layers can always be viewed as networks that have a single laterally connected hidden layer with many of the lateral and feedforward connections missing. However, the time taken for the network to settle means that networks with lateral connections are considerably slower than purely feedforward networks, and more research is required to demonstrate that the loss of speed is worth the extra representational power.

A much simpler way to use fixed lateral interactions is to divide the hidden units into mutually exclusive pools. Within each pool, exactly one hidden unit is allowed to turn on, and this unit is chosen using the "softmax" distribution:

$$p_j = \frac{\exp(x_j)}{\sum_k \exp(x_k)}. \tag{12.2}$$

Curiously, this type of probabilistic winner-take-all competition among the hidden units has no effect whatsoever on the contrastive divergence learning rule. The learning does not even need to know which hidden units compete with each other. The competition can be implemented in a very simple way by making each hidden unit, j, be a Poisson process with a rate of $\exp(x_j)$. The first unit to emit a spike is the winner. The other units in the pool could then be shut off by an inhibitory unit that is excited by the winner. If the pool is allowed to produce a fixed number of spikes before being shut off, the learning rule remains unaltered, but quantities like s_j may take on integer values larger than 1. If a hidden unit is allowed to be a member of several different overlapping pools, the analysis gets much more difficult, and more research is needed.

12.5 Relationship to Analysis-by-Synthesis. PoEs provide an efficient instantiation of the old psychological idea of analysis-by-synthesis. This idea never worked properly because the generative models were not selected to make the analysis easy. In a PoE, it is difficult to generate data from the generative model, but given the model, it is easy to compute how any given data vector might have been generated, and, as we have seen, it is relatively easy to learn the parameters of the generative model. Paradoxically, the generative models that are most appropriate for analysis-by-synthesis may be the ones in which synthesis is intractable.

Acknowledgments

This research was funded by the Gatsby Charitable Foundation. Thanks to Zoubin Ghahramani, David MacKay, Peter Dayan, Radford Neal, David Lowe, Yee-Whye Teh, Guy Mayraz, Andy Brown, and other members of the Gatsby unit for helpful discussions and to the referees for many helpful comments that improved the article.

References

Berger, A., Della Pietra, S., & Della Pietra, V. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*, 39–71.

Brown, A. D., & Hinton, G. E. (2001). Products of HMMs. In *Proceedings of Artificial Intelligence and Statistics 2001* (pp. 3–11). San Mateo, CA: Morgan Kaufmann.

Freund, Y., & Haussler, D. (1992). Unsupervised learning of distributions on binary vectors using two layer networks. In J. E. Moody, S. J. Hanson, & R. P. Lippmann (Eds.), *Advances in neural information processing systems, 4* (pp. 912–919). San Mateo, CA: Morgan Kaufmann.

Geman, S., & Geman, D. (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *6*, 721–741

Genest, C., & Zidek, J. V. (1986). Combining probability distributions: A critique and an annotated bibliography. *Statistical Science*, *1*, 114–148.

Heskes, T. (1998). Bias/variance decompositions for likelihood-based estimators. *Neural Computation*, *10*, 1425–1433.

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

Hinton, G. E., & McClelland, J. L. (1988). Learning representations by recirculation. In D. Z. Anderson (Ed.), *Neural information processing systems* (pp. 358–366). New York: American Institute of Physics.

Hinton, G. E., Sallans, B., & Ghahramani, Z. (1998). Hierarchical communities of experts. In M. I. Jordan (Ed.), *Learning in graphical models*. Norwood, MA: Kluwer.

Hinton, G. E., & Sejnowski, T. J. (1986). Learning and relearning in BMs. In D. E. Rumelhart & J. L. McClelland (Eds.), *Parallel distributed processing : Explorations in the microstructure of cognition. Vol. 1: Foundations*. Cambridge, MA: MIT Press.

Mayraz, G., & Hinton, G. E. (2001). Recognizing hand-written digits using hierarchical products of experts. In T. K. Leen, T. G. Dietterich, & V. Tresp (Eds.), *Advances in neural information processing systems, 13* (pp. 953–959). Cambridge, MA: MIT Press.

Neal, R. M. (1992). Connectionist learning of belief networks. *Artificial Intelligence*, *56*, 71–113.

O'Reilly, R. C. (1996) Biologically plausible error-driven learning using local activation differences: The generalized recirculation algorithm. *Neural Computation*, *8*, 895–938.

Revow, M., Williams, C. K. I., & Hinton, G. E. (1996). Using generative models for handwritten digit recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *18*, 592–606.

Saul, L. K., & Jordan, M. I. (2000). Attractor dynamics in feedforward neural networks. *Neural Computation*, *12*, 1313–1335.

Seung, H. S. (1998). Learning continuous attractors in a recurrent net. In M. I. Jordan, M. J. Kearns, & S. A. Solla (Eds.) *Advances in neural information processing systems, 10* (pp. 654–660). Cambridge, MA: MIT Press.

Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart & J. L. McClelland (Eds.), *Par-*