Learning multiple layers of representation

Geoffrey E. Hinton
Department of Computer Science, University of Toronto
10 Kings College Road
Toronto, Canada M5S 3G4
hinton@cs.toronto.edu
Tel 416-978-7564
FAX 416-978-1455

October 20, 2007

Abstract

To achieve its' impressive performance at tasks such as speech or object recognition, the brain extracts multiple levels of representation from the sensory input. Backpropagation was the first computationally efficient model of how neural networks could learn multiple layers of representation, but it required labeled training data and it did not work well in deep networks. The limitations of backpropagation learning can now be overcome by using multi-layer neural networks that contain top-down connections and training them to *generate* sensory data rather than to classify it. Learning multilayer generative models appears to be difficult, but a recent discovery makes it easy to learn non-linear, distributed representations one layer at a time. The multiple layers of representation learned in this way can subsequently be fine-tuned to produce generative or discriminative models that work much better than previous approaches.

Learning feature detectors

To enable the perceptual system to make the fine distinctions that are required to control behaviour, sensory cortex needs an efficient way of adapting the synaptic weights of multiple layers of feature-detecting neurons. The backpropagation learning procedure[1] iteratively adjusts all of the weights to optimize some measure of the network's classification performance, but this requires labeled training data. To learn multiple layers of feature detectors when labeled data is scarce or non-existent, some objective other than classification is required. In a neural network that contains both bottom-up "recognition" connections and top-down "generative" connections it is possible to recognize data using a bottom-up pass and to generate data using a top-down pass. If the neurons are stochastic, repeated top-down passes will generate a whole distribution of data vectors. This suggests a sensible objective for learning: Adjust the weights on the top-down connections to maximize the probability that the network would generate the training data. The neural network's model of the training data then resides in its top-down connections. The role of the bottom-up connections is to allow the network to figure out activations for the features in each layer that constitute a plausible explanation of how the network could have generated an observed sensory data-vector. The hope is that the active features in the higher layers will be a much better guide to appropriate actions than the raw sensory data or the lower-level features. As we shall see, this is not just wishful thinking – if three layers of feature detectors are trained on unlabeled images of handwritten digits, the complicated non-linear features in the top layer allow excellent recognition of poorly written digits like those in 4b [2].

There are several reasons for believing that our visual systems contain multi-layer generative models in which top-down connections can be used to generate low-level features of images from high-level representations, and bottom-up connections can be used to infer the high-level representations that would have generated an observed set of low-level features. Single cell recordings [3] and the reciprocal connectivity between cortical areas [4] both suggest a hierarchy of progressively more complex features in which each layer can influence the layer below. Vivid visual imagery, dreaming, and the disambiguating effect of context on the interpretation of local image regions [5] also suggest that the visual system can perform top-down generation.

The aim of this review is to complement the neural and psychological evidence for generative models by reviewing recent computational advances that make multi-layer generative models easier to learn and better at discrimination than their feed-forward counterparts. The advances are illustrated in the domain of hand-written digits where they learn unsurpassed generative and discriminative models.

Inference in generative models

The crucial computational step in fitting a generative model to data is figuring out how the model, with its current generative parameters, might have used its hidden variables to generate an observed data-vector. Stochastic generative models generally have many different ways of generating any particular data-vector, so the best we can hope for is to infer a probability distribution over the various possible settings of the hidden variables. Consider, for example, a mixture of Gaussians model in which each data-vector is assumed to come from exactly one of the multivariate Gaussian distributions in the mixture. Inference then consists of computing the posterior probability that a particular data-vector came from each of the Gaussians. This is easy because the posterior probability assigned to each Gaussian in the mixture is simply proportional to the probability density of the data-vector under that Gaussian times the prior probability of using that Gaussian when generating data.

The generative models that are most familiar in statistics and machine learning are the ones for which the posterior distribution can be inferred efficiently and exactly because the model has been strongly constrained. These generative models include:

- **Factor Analysis** in which there is a single layer of Gaussian hidden variables that have a linear effects on the visible variables (see figure 1). In addition, independent Gaussian noise is added to each visible variable[6, 7, 8]. Given a visible vector, it is impossible to infer the exact state of the factors that generated it, but it is easy to infer the mean and covariance of the Gaussian posterior distribution over the factors and this is sufficient to allow the parameters of the model to be improved.

- **Independent Components Analysis** which generalizes factor analysis by allowing non-Gaussian hidden variables, but maintains tractable inference by eliminating the observation noise in the visible variables and using the same number of hidden and visible variables. These restrictions ensure that the posterior distribution collapses to a single point because there is only one setting of the hidden variables that can generate each visible vector exactly[9, 10, 11].

- **Mixture models** in which each data-vector is assumed to be generated by one of the component distributions in the mixture and it is easy to compute the density under each of the component distributions.

If factor analysis is generalized to allow non-Gaussian hidden variables, it can model the development of low-level visual receptive fields[12]. However, if the extra constraints used in independent components analysis are not imposed, it is no longer easy to infer, or even to represent, the posterior distribution over the hidden variables. This is because of a phenomenon known as explaining away [13] (see figure 2b).

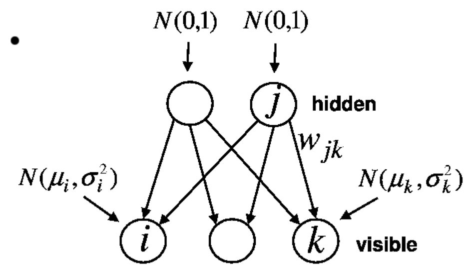
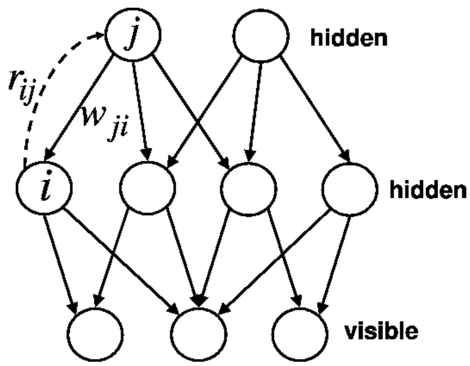


Figure 1: The generative model used in factor analysis. Each real-valued hidden factor is chosen independently from a Gaussian distribution, $N(0,1)$, with zero mean and unit variance. Then the factors are linearly combined and Gaussian observation noise is added independently to each real-valued visible variable.

Multi-layer generative models

Generative models with only one hidden layer are much too simple for modeling the high-dimensional and richly structured sensory data that arrives at the cortex, but they have been pressed into service because, until recently, it was too difficult to perform inference in the more complicated, multi-layer, non-linear models that are clearly required. There have been many attempts to develop multi-layer, non-linear models [14, 15, 16, 17, 18]. In Bayes nets (also called belief nets) which have been studied intensively in artificial intelligence and statistics, the hidden variables typically have discrete values. Exact inference is possible if every variable only has a few parents. This may occur in Bayes nets that are used to formalize expert knowledge in limited domains [19], but for more densely connected Bayes nets, exact inference is generally intractable.



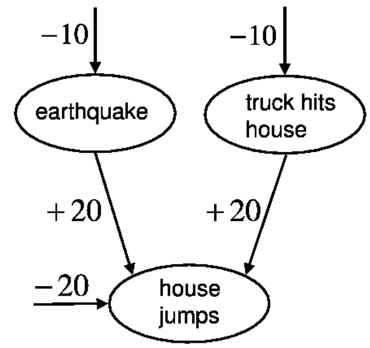


Figure 2: a) A multi-layer belief net composed of logistic binary units. To generate fantasies from the model, we start by picking a random binary state of 1 or 0 for each top-level unit. Then we perform a stochastic downwards pass in which the probability, $\hat{h}_i$, of turning on each unit, i, is determined by applying the logistic function, $\sigma(x) = 1/(1 + \exp(-x))$, to the total input $\sum_j h_j w_{ji}$ that i receives from the units, j, in the layer above, where h_j is the binary state that has already been chosen for unit j. It is easy to give each unit an additional bias, but to simplify this review biases will usually be ignored. r_{ij} is a recognition weight. b) An illustration of "explaining away" in a very simple logistic belief net containing two independent, rare, hidden causes that become highly anti-correlated when we observe the house jumping. The bias of -10 on the earthquake unit means that, in the absence of any observation, this unit is e^{10} times more likely to be off than on. If the earthquake unit is on and the truck unit is off, the jump unit has a total input of 0 which means that it has an even chance of being on. This is a much better explanation of the observation that the house jumped than the odds of e^{-20} which apply if neither of the hidden causes is active. But it is wasteful to turn on both hidden causes to explain the observation because the probability of them both happening is approximately e^{-20}.

It is important to realize that if some way can be found to infer the posterior distribution over the hidden variables for each data-vector, learning a multilayer generative model is relatively straightforward. Learning is also straighforward if we can get unbiased samples from the posterior distribution. In this case, we simply adjust the parameters so as to increase the probability that the sampled states of the hidden variables in each layer would generate the sampled states of the hidden or visible variables in the layer below. In the case of the logistic belief net shown in figure 2a, which will be a major focus of this review, the learning rule for each training case is a version of the delta rule [20]. The inferred state, h_i, of the "postsynaptic" unit, i, acts as a target value and the probability, $\hat{h}_i$, of activating i given the inferred states, h_j, of all the "presynaptic" units, j, in the layer above acts as a prediction:

$$\Delta w_{ji} \propto h_j(h_i - \hat{h}_i) \tag{1}$$

If i is a visible unit, h_i is replaced by the actual state of i in the training example.

If training vectors are selected with equal probability from the training set and the hidden states are sampled from their posterior distribution given the training vector, the learning rule in Eq. 1 has a positive expected effect on the probability that the generative model would produce exactly the N training vectors if it was run N times.

Approximate inference for multilayer generative models

The generative model in figure 2a is defined by the weights on its top-down, generative connections, but it also has bottom-up, recognition connections that can be used to perform approximate inference in a single, bottom-up pass. The inferred probability that $h_j = 1$ is $\sigma(\sum_i h_i r_{ij})$. This inference procedure is fast and simple, but it is incorrect because it ignores explaining away. Surprisingly, learning is still possible with incorrect inference because there is a more general objective function that the learning rule in Eq. 1 is guaranteed to improve [21, 22].

Instead of just considering the log probability of generating each training case, we can also take the accuracy of the inference procedure into account.

Other things being equal, we would like our approximate inference method to be as accurate as possible, and we might even prefer a model that is slightly less likely to generate the data if it allows more accurate inference of the hidden representations. So it makes sense to use the inaccuracy of inference on each training case as a penalty term when maximizing the log probability of the observed data. This leads to a new objective function that is easy to maximize and is a "variational" lower-bound on the log probability of generating the training data[23]. Learning by optimizing a variational bound is now a standard way of dealing with the intractability of inference in complex generative models[24, 25, 26, 27]. An approximate version of this type of learning has been proposed as a model of learning in sensory cortex (see box 1), but it is very slow in deep networks if the weights are initialized randomly.

`TEXTBOX 1 GOES ABOUT HERE`

A non-linear module with fast, exact inference

We now turn to a very different type of model called a "restricted Boltzmann machine"[28] (see figure 3a). Despite its undirected, symmetric connections, the RBM is the key to finding an efficient learning procedure for deep, directed, generative models.

Images composed of binary pixels can be modeled by using the hidden layer of an RBM to model the higher-order correlations between pixels[29]. To learn a good set of feature detectors from a set of training images, we start with zero weights on the symmetric connections between each pixel i and each feature detector j. Then we repeatedly update each weight, w_{ij}, using the difference between two measured, pairwise correlations

$$\Delta w_{ij} = \epsilon(<v_i h_j>_{data} - <v_i h_j>_{recon}) \tag{2}$$

where ϵ is a learning rate, $<v_i h_j>_{data}$ is the frequency with which pixel i and feature detector j are on together when the feature detectors are being driven by images from the training set and $<v_i h_j>_{recon}$ is the corresponding frequency when the feature detectors are being driven by reconstructed images. A similar learning rule can be used for the biases.

Given a training image, we set the binary state, h_j, of each feature detector to be 1 with probability

$$p(h_j = 1) = \sigma(b_j + \sum_i v_i w_{ij}) \tag{3}$$

where $\sigma(\cdot)$ is the logistic function, b_j is the bias of j and v_i is the binary state of pixel i. Once binary states have been chosen for the hidden units we produce a "reconstruction" of the training image by setting the state of each pixel to be 1 with probability

$$p(v_i = 1) = \sigma(b_i + \sum_j h_j w_{ij}) \tag{4}$$

The learned weights and biases directly determine the conditional distributions $p(\mathbf{h}|\mathbf{v})$ and $p(\mathbf{v}|\mathbf{h})$ using Eqs. 3 and 4. Indirectly, the weights and biases define the joint and marginal distributions $p(\mathbf{v}, \mathbf{h})$, $p(\mathbf{v})$ and $p(\mathbf{h})$. Sampling from the joint distribution is difficult, but it can be done by using "alternating Gibbs sampling". This starts with a random image and then alternates between updating all of the features in parallel using Eq. 3 and updating all of the pixels in parallel using Eq. 4. After Gibbs sampling for sufficiently long, the network reaches "thermal equilibrium". The states of pixels and features detectors still change, but the probability of finding the system in any particular binary configuration does not. By observing the fantasies on the visible units at thermal equilibrium, we can see the distribution over visible vectors that the model believes in.

The RBM has two major advantages over directed models with one hidden layer. First, inference is very easy because there is no explaining away: Given a visible vector, the posterior distribution over hidden vectors factorizes into a product of independent distributions for each hidden unit. So to get a sample from the posterior we simply turn on each hidden unit with a probability given by Eq. 3. Second, as we shall see, it is easy to learn deep directed networks one layer at a time by stacking RBM's. Layer–by-layer learning does not work nearly as well when the individual modules are directed, because each directed module bites off more than it can chew: It tries to learn hidden causes that are marginally independent. This is generally beyond its abilities so it settles for a generative model in which independent causes generate a poor approximation to the data distribution.

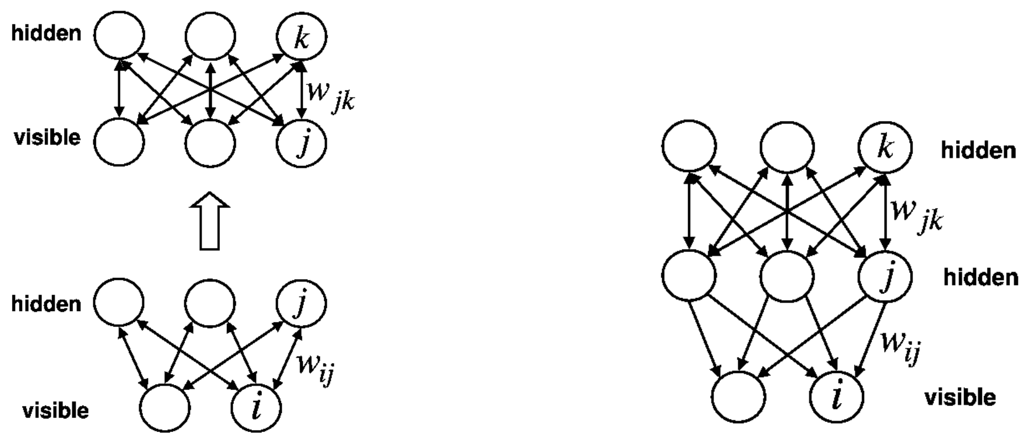


Figure 3: a) Two separate restricted Boltzmann machines (RBM's). The stochastic, binary variables in the hidden layer of each RBM are symmetrically connected to the stochastic, binary variables in the visible layer. There are no connections within a layer. The higher-level RBM is trained by using the hidden activities of the lower RBM as data. b) The composite generative model produced by composing the two RBM's. Note that the connections in the lower layer of the composite generative model are directed. The hidden states are still inferred by using bottom-up recognition connections, but these are no longer part of the generative model.

Learning many layers of features by composing RBM's

After learning an RBM, the activities of its hidden units, when they are being driven by data, can be used as the "data" for learning a higher-level RBM. To understand why this is a good idea, it is helpful to consider decomposing the problem of modeling the data distribution, P_0, into two subproblems by picking a distribution, P_1, that is easier to model than P_0. The first subproblem is to model P_1 and the second subproblem is to model the transformation from P_1 to P_0. P_1 is the distribution obtained by applying $p(\mathbf{h}|\mathbf{v})$ to the data distribution to get the hidden activities for every data vector in the training set. P_1 is easier for an RBM to model than P_0 because it is obtained from P_0 by allowing an RBM to settle towards a distribution that it can model perfectly: its equilibrium distribution. The RBM's *model* of P_1 is $p(\mathbf{h})$, the distribution over hidden vectors when the RBM is sampling from its equilibrium distribution. The RBM's model of the transformation from P_1 to P_0 is $p(\mathbf{v}|\mathbf{h})$.

After the first RBM has been learned, we keep $p(\mathbf{v}|\mathbf{h})$ as part of the generative model and we keep $p(\mathbf{h}|\mathbf{v})$ as a quick way of performing inference, but we throw away our model of P_1 and replace it by a better model that is obtained, recursively, by treating P_1 as the training data for the second-level RBM. This leads to the composite generative model shown in figure 3b. To generate from this model we need to get an equilibrium sample from the top-level RBM, but then we simply perform a single downwards pass through the bottom layer of weights. So the composite model is a curious hybrid whose top two layers form an undirected associative memory and whose lower layers form a directed generative model. It is shown in [30] that if the second RBM is initialized appropriately, the gain from building a better model of P_1 always outweighs the loss that comes from the fact that $p(\mathbf{h}|\mathbf{v})$ is no longer the correct way to perform inference in the composite generative model shown in 3b. Adding another hidden layer always improves a variational bound on the log probability of the training data unless the top-level RBM is already a perfect model of the data it is trained on.

Modeling images of hand-written digits

Figure 4a shows a network that was used to model the joint distribution of digit images and their labels. It was learned one layer at a time and the top level RBM was trained using "data" vectors that were constructed by concatenating the states of 10 winner-take-all label units with 500 binary features inferred from the image. After greedily learning one layer of weights at a time, all the weights were fine-tuned using a variant of the wake-sleep algorithm (see [30] for details). The fine-tuning significantly improves the ability of the model to generate images that resemble the data, but without the initial layer-by-layer learning, the fine-tuning alone is hopelessly slow.

The model was trained to generate both a label and an image, but it can be used to classify new images. First, the recognition weights are used to infer binary states for the 500 feature units in the second hidden layer,

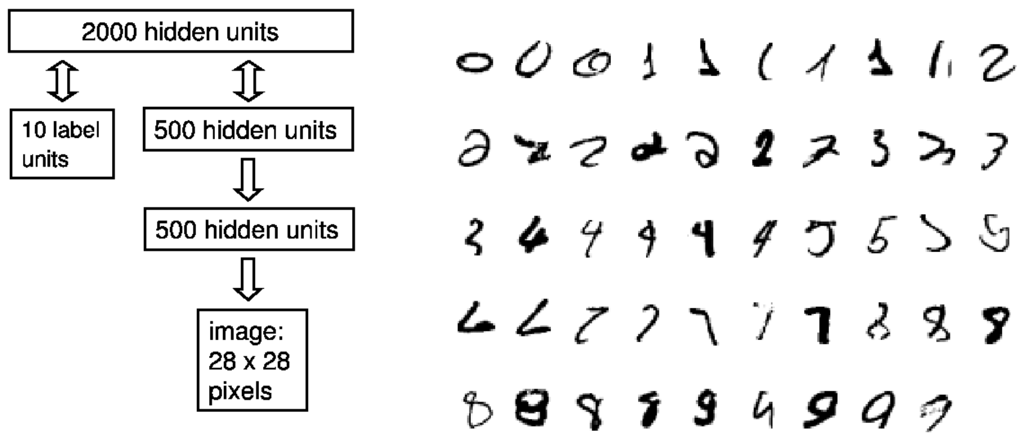


Figure 4: a) The generative model used to learn the joint distribution of digit images and digit labels. b) Some test images that the network classifies correctly even though it has never seen them before.

then alternating Gibbs sampling is applied to the top two layers with these 500 features held fixed. The probability of each label is then represented by the frequency with which it turns on. Using an efficient version of this method, the network significantly outperforms both backpropagation and support vector machines[31] when trained on the same data [30]. A demonstration of the model generating and recognizing digit images is at www.cs.toronto.edu/∼hinton.

Instead of fine-tuning the model to be better at generating the data, backpropagation can be used to fine-tune it to be better at discrimination. This works extremely well [2, 20]. The initial layer-by-layer learning finds features that allow good generation and then the discriminative fine-tuning slightly modifies these features to adjust the boundaries between classes. This has the great advantage that the very limited amount of information in the labels is used only for perturbing features, not for creating them. If the ultimate aim is discrimination it is possible to use autoencoders with a single hidden layer instead of restricted Boltzmann machines for the unsupervised, layer-by-layer learning [32]. This produces the best results ever achieved on the most commonly used benchmark for handwritten digit recognition [33].

Modeling sequential data

This review has focused on static images, but restricted Boltzmann machines can also be applied to high-dimensional sequential data such as video sequences[34] or the joint angles of a walking person[35]. The visible and hidden units are given additional, conditioning, inputs that come from previous visible frames. The conditioning inputs have the effect of dynamically setting the biases of the visible and hidden units. These conditional restricted Boltzmann machines can be composed by using the sequence of hidden activities of one as the training data for the next. This creates multi-layer distributed representations of sequences that are far more powerful than the representations learned by standard methods such as hidden Markov models or linear dynamical systems[34].

Concluding Remarks

A combination of three ideas leads to a novel and very effective way of learning multiple layers of representation. The first idea is to learn a model that *generates* sensory data rather than classifying it. This eliminates the need for large amounts of labeled data. The second idea is to learn one layer of representation at a time using restricted Boltzmann machines. This decomposes the overall learning task into multiple simpler tasks and it eliminates the inference problems that arise in directed generative models. The third idea is to use a separate fine-tuning stage to improve the generative or discriminative abilities of the composite model.

Versions of this approach are currently being applied to tasks as diverse as denoising images[36, 37], retrieving documents[38, 2], extracting optical flow[39] predicting the next word in a sentence[40] and predicting what movies people will like[41]. [42] gives further reasons for believing that this approach holds great promise for artificial intelligence applications, such as human-level speech and object recognition, that have proved too difficult for shallow methods like support vector machines[31] that cannot learn multiple layers of representation.

There is no concise definition of the types of data for which this approach is likely to be successful, but it seems most appropriate when hidden variables generate richly-structured sensory data that provides a great deal of information about the states of the hidden variables. If the hidden variables also generate a label that contains little information or is only occasionally observed, it is a bad idea to try to learn the mapping from sensory data to labels using discriminative learning methods. It is much more sensible to first learn a generative model that infers the hidden variables from the sensory data and then learn the simpler mapping from the hidden variables to the labels.

Acknowledgments

I thank Yoshua Bengio, David MacKay, Terry Sejnowski and my past and present postdoctoral fellows and graduate students for helping me to understand these ideas and NSERC, CIAR, CFI and OIT for support.

Box 1. The wake-sleep algorithm

For the logistic belief net shown in figure 2a, it is easy to improve the generative weights if the network already has a good set of recognition weights. For each data-vector in the training set, the recognition weights are used in a bottom-up pass that stochastically picks a binary state for each hidden unit. Applying the learning rule in Eq. 1 will then follow the gradient of a variational bound on how well the network generates the training data [22].

It is not so easy to compute the derivatives of the bound with respect to the recognition weights, but there is a simple, approximate learning rule which works quite well in practice. If we generate fantasies from the model by using the generative weights in a top-down pass, we know the true causes of the activities in each layer, so we can compare the true causes with the predictions made by the approximate infererence procedure and adjust the recognition weights, r_{ij}, to maximize the probability that the predictions are correct:

$$\Delta r_{ij} \propto h_i \left(h_j - \sigma(\sum_i h_i r_{ij}) \right) \tag{5}$$

The combination of approximate inference for learning the generative weights, and fantasies for learning the recognition weights is known as the "wake-sleep" algorithm [22].

Box 2. Questions for future research

The initial successes of this approach to learning deep networks raise many questions:

- How might this type of algorithm be implemented in cortex? In particular, is the initial perception of sensory input closely followed by a reconstruction that uses top-down connections? Computationally, the learning procedure for restricted Boltzmann machines does not require a "pure" reconstruction. All that is required is that there are two phases which differ in the relative balance of bottom-up and top-down influences, with synaptic potentiation in one phase and synaptic depression in the other.

- Can this approach deal adequately with lateral connections and inhibitory interneurons? Currently, there is no problem in allowing lateral interactions between the visible units of a "semi-restricted" Boltzmann machine[30, 43]. Lateral interactions between the hidden units can be added when these become the visible units of the higher-level, semi-restricted Boltzmann machine. This makes it possible to learn a hierachy of undirected Markov random fields each of which has directed connections to the field below as suggested in [44]. This is a more powerful type of generative model because each level only needs to provide a rough specification of the states at the level below: The lateral interactions at the lower level can settle on the fine details and ensure that they obey learned constraints.

- Can we understand the representations that are learned in the deeper layers? In a generative model, it is easy to see what a distributed pattern of activity over a whole layer means: Simply generate from it to get a sensory input vector (*e.g.* an image). It is much harder to discover the meaning of activity in an individual neuron in the deeper layers because the effects of that activity depend on the states of all the other nonlinear neurons. The fact the some neurons in the ventral stream can be construed as face detectors is intriguing, but I can see no good reason to expect such simple stories to be generally applicable.

References

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

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

[3] T.S. Lee, D. Mumford, R. Romero, and V.A.F. Lamme. The role of the primary visual cortex in higher level vision. *Vision Research*, 38(15):2429–2454, 1998.

[4] D.J. Felleman and D.C. Van Essen. Distributed Hierarchical Processing in the Primate Cerebral Cortex. *Cerebral Cortex*, 1(1):1–47, 1991.

[5] D. Mumford. On the computational architecture of the neocortex. II. The role of cortico-cortical loops. *Biological Cybernetics*, 66(3):241–51, 1992.

[6] P. Dayan and L.F. Abbott. *Theoretical neuroscience: computational and mathematical modeling of neural systems*. MIT Press, 2001.

[7] S. Roweis and Z. Ghahramani. A unifying review of linear gaussian models. *Neural Comnputation*, 11(2):305–345, 1999.

[8] T.K. Marks and J.R. Movellan. Diffusion networks, products of experts, and factor analysis. *Proc. Int. Conf. on Independent Component Analysis*, pages 481–485, 2001.

[9] A.J. Bell and T.J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural Computation*, 7(6):1129–1159, 1995.

[10] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*. Wiley, 2001.

[11] M.S. Bartlett, J.R. Movellan, and T.J. Sejnowski. Face recognition by independent component analysis. *IEEE Transactions on Neural Networks*, 13(6):1450–1464, 2002.

[12] B.A. Olshausen and D. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607–609, 1996.

[13] J. Pearl. *Probabilistic Inference in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[14] M. S. Lewicki and T. J. Sejnowski. Bayesian unsupervised learning of higher order structure. In *Advances in Neural Information Processing Systems*, volume 9, pages 529–535. MIT Press, 1997.

[15] P.O. Hoyer and A. Hyvarinen. A multi-layer sparse coding network learns contour coding from natural images. *Vision Research*, 42(12):1593–1605, 2002.

[16] J. Portilla, V. Strela, M.J. Wainwright, and E.P. Simoncelli. Image denoising using Gaussian scale mixtures in the wavelet domain. *IEEE Transactions on Image Processing*, 12(11):1338–1351, 2004.

[17] O. Schwartz, T.J. Sejnowski, and P. Dayan. Soft Mixer Assignment in a Hierarchical Generative Model of Natural Scene Statistics. *Neural Computation*, 18(11):2680–2718, 2006.

[18] Y. Karklin and M.S. Lewicki. Learning higher-order structures in natural images. *Network: Computation in Neural Systems*, 14(3):483–499, 2003.

[19] R.G. Cowell, S.L. Lauritzen, D.J. Spiegelhater, and A.P. Dawid. *Probabilistic Networks and Expert Systems*. Springer, 2003.

[20] R. C. O'Reilly. Six principles for biologically based computational models of cortical cognition. *Trends in Cognitive Sciences*, 2(11):455–462, 1998.

[21] G. E. Hinton and R. S. Zemel. Autoencoders, minimum description length, and Helmholtz free energy. *Advances in Neural Information Processing Systems*, 6:3–10, 1994.

[22] G. E. Hinton, P. Dayan, B. J. Frey, and R. Neal. The wake-sleep algorithm for self-organizing neural networks. *Science*, 268:1158–1161, 1995.

[23] R. M. Neal and G. E. Hinton. A new view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355—368. Kluwer Academic Publishers, 1998.

[24] M.I. Jordan, Z. Ghahramani, T.S. Jaakkola, and L.K. Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, 1999.

[25] J. Winn and N. Jojic. LOCUS: Learning Object Classes with Unsupervised Segmentation. In *Tenth IEEE International Conference on Computer Vision*, volume 1, pages 756–763, 2005.

[26] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[27] C.M. Bishop, D. Spiegelhalter, and J. Winn. VIBES: A variational inference engine for Bayesian networks. *Advances in Neural Information Processing Systems*, 15, 2002.

[28] G. E. Hinton. Boltzmann machines. *Scholarpedia*, 2007.

[29] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1711–1800, 2002.

[30] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18, 2006.

[31] D. Decoste and B. Schoelkopf. Training invariant support vector machines. *Machine Learning*, 46:161–190, 2002.

[32] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In Schoelkopf B., J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[33] C.P. MarcAurelio Ranzato, S. Chopra, and Y. LeCun. Efficient Learning of Sparse Representations with an Energy-Based Model. In Schoelkopf B., J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[34] I. Sutskever and G. E. Hinton. Learning multilevel distributed representations for high-dimensional sequences. In *Artificial Intelligence and Statistics, 2007*, 2007.

[35] G. W. Taylor, G. E. Hinton, and S. Roweis. Modeling human motion using binary latent variables. In Schoelkopf B., J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*. MIT Press, Cambridge, MA, 2007.

[36] S. Roth and MJ Black. Fields of Experts: a framework for learning image priors. In *Computer Vision and Pattern Recognition*, volume 2. IEEE Computer Society, 2005.

[37] M. Ranzato, Y. Boureau, S. Chopra, and Y. LeCun. A unified energy-based framework for unsupervised learning. In *Artificial Intelligence and Statistics*, Puerto Rico, 2007.

[38] M. Welling, M. Rosen-Zvi, and G. E. Hinton. Exponential family harmoniums with an application to information retrieval. In *Advances in Neural Information Processing Systems* **17**, pages 1481–1488. MIT Press, Cambridge, MA, 2005.

[39] R. F. Memisevic and G. E. Hinton. Unsupervised learning of image transformations. In *Computer Vision and Pattern Recognition*. IEEE Computer Society, 2007.

[40] A. Mnih and G. E. Hinton. Three new graphical models for statistical language modelling. In *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007. To appear.

[41] R. R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007. To appear.

[42] Y. Bengio and Y. Le Cun. Scaling learning algorithms towards ai. In L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, editors, *Large-Scale Kernel Machines*. MIT Press, Cambridge, MA, 2007. To appear.

[43] M. Welling and G. E. Hinton. A new learning algorithm for mean field boltzmann machines. In *ICANN*, Madrid, 2002.

[44] G. E. Hinton, S. Osindero, and K. Bao. Learning Causally Linked Markov Random Fields. In *Artificial Intelligence and Statistics*, Barbados, 2005.