

1 Unsupervised learning B

1.1 Dimensionality reduction

1.1.1 PCA, and local embedding methods

See slides

1.1.2 Deep auto-encoders

See: Pascal Vincent, Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol, Université de Montréal, ICML 2008.

1.2 The Boltzmann machine

1.2.1 General one-layer module

Our last model that uses unsupervised learning is again a general learning machine invented by Geoffrey Hinton and Terrance Sejnowski in the mid 1980 called **Boltzmann machine**. This machine is a general form of a recurrent neural network with **visible nodes** that receive input or provide output, and **hidden nodes** that are not connected to the outside world directly. Such a stochastic dynamic network, a recurrent system with hidden nodes, together with the adjustable connections, provide the system with enough degrees of freedom to approximate any dynamical system. While this has been recognized for a long time, finding practical training rules for such systems have been a major challenge for which there was only recently major progress. These machines use unsupervised learning to learn hierarchical representations based on the statistics of the world. Such representations are key to more advanced applications of machine learning and to human abilities.

The basic building block is a one-layer network with one visible layer and one hidden layer. An example of such a network is shown in Fig. 1.1. The nodes represent random variable similar to the Bayesian networks discussed before. We will specifically consider binary nodes that mimic neuronal states which are either firing or not. The connections between them have weights w_{ij} which specify how much they influence the on-state of connected nodes. Such systems can be described by an energy function. The energy between two nodes that are symmetrically connected with strength w_{ij} is

$$H^{nm} = -\frac{1}{2} \sum_{ij} w_{ij} s_i^n s_j^m. \quad (1.1)$$

The state variables, s , have superscripts n or m which can have values (v) or (h) to indicate visible and hidden nodes. We consider again the probabilistic update rule,

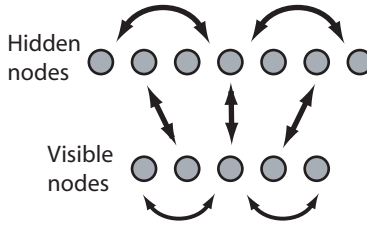


Fig. 1.1 A Boltzmann machine with one visible and one hidden layer.

$$p(s_i^n = +1) = \frac{1}{1 + \exp(-\beta \sum_j w_{ij} s_j^n)}, \quad (1.2)$$

with inverse temperature, β , which is called the Glauber dynamics in physics and describes the competitive interaction between minimizing the energy and the randomizing thermal force. The probability distribution for such a stochastic system is called the Boltzmann–Gibbs distribution. Following this distribution, the distribution of visible states, in thermal equilibrium, is given by

$$p(\mathbf{s}^v; \mathbf{w}) = \frac{1}{Z} \sum_{m \in h} \exp(-\beta H^{vm}), \quad (1.3)$$

where we summed over all hidden states. In other words, this function describes the distribution of visible states of a Boltzmann machine with specific parameters, \mathbf{w} , representing the weights of the recurrent network. The normalization term, $Z = \sum_{n,m} \exp(-\beta H^{nm})$, is called the **partition function**, which provides the correct normalization so that the sum of the probabilities of all states sums to one. These stochastic networks with symmetrical connections have been termed **Boltzmann machines** by **Ackley, Hinton and Sejnowski**.

Let us consider the case where we have chosen enough hidden nodes so that the system can, given the right weight values, implement a generative model of a given world. Thus, by choosing the right weight values, we want this dynamical system to approximate the probability function, $p(\mathbf{s}^v)$, of the sensory states (states of visible nodes) caused by the environment. To derive a learning rule, we need to define an objective function. In this case, we want to minimize the difference between two density functions. A common measure for the difference between two probabilistic distributions is the Kulbach–Leibler divergence (see Appendix ??),

$$\text{KL}(p(\mathbf{s}^v), p(\mathbf{s}^v; \mathbf{w})) = \sum_{\mathbf{s}} p(\mathbf{s}^v) \log \frac{p(\mathbf{s}^v)}{p(\mathbf{s}^v; \mathbf{w})} \quad (1.4)$$

$$= \sum_{\mathbf{s}} p(\mathbf{s}^v) \log p(\mathbf{s}^v) - \sum_{\mathbf{s}} p(\mathbf{s}^v) \log p(\mathbf{s}^v; \mathbf{w}). \quad (1.5)$$

To minimize this divergence with a gradient method, we need to calculate the derivative of this ‘distance measure’ with respect to the weights. The first term in the difference in eqn 1.5 is the entropy (see Appendix ??) of sensory states, which does not depend on

the weights of the Boltzmann machine. Minimizing the Kulbach–Leibler divergence is therefore equivalent to maximizing the average log-likelihood function,

$$l(\mathbf{w}) = \sum_{\mathbf{s}}^v p(\mathbf{s}^v) \log p(\mathbf{s}^v; \mathbf{w}) = \langle \log p(\mathbf{s}^v; \mathbf{w}) \rangle. \quad (1.6)$$

In other words, we treat the probability distribution produced by the Boltzmann machine as a function of the parameters, w_{ij} , and choose the parameters which maximize the likelihood of the training data (the actual world states). Therefore, the averages of the model are evaluated over actual visible states generated by the environment. The log-likelihood of the model increases the better the model approximates the world. A standard method of maximizing this function is gradient ascent, for which we need to calculate the derivative of $l(\mathbf{w})$ with respect to the weights. We omit the detailed derivation here, but we note that the resulting learning rule can be written in the form

$$\Delta w_{ij} = \eta \frac{\partial l}{\partial w_{ij}} = \eta \frac{\beta}{2} (\langle s_i s_j \rangle_{\text{clamped}} - \langle s_i s_j \rangle_{\text{free}}). \quad (1.7)$$

The meaning of the terms on the right-hand side is as follows. The term labelled ‘clamped’ is the thermal average of the correlation between two nodes when the states of the visible nodes are fixed. The term labelled ‘free’ is the thermal average when the recurrent system is running freely. The Boltzmann machine can thus be trained, in principle, to represent any arbitrary density functions, given that the network has a sufficient number of hidden nodes.

This result is encouraging as it gives us an exact algorithm to train general recurrent networks to approximate arbitrary density functions. The learning rule looks interesting since the clamped phase could be associated with a sensory driven agent during an awake state, whereas the freely running state could be associated with a sleep phase. Unfortunately, it turns out that this learning rule is too demanding in practice. The reason for this is that the averages, indicated by the angular brackets in eqn 1.7, have to be evaluated at thermal equilibrium. Thus, after applying each sensory state, the system has to run for a long time to minimize the initial transient response of the system. The same has to be done for the freely running phase. Even when the system reaches equilibrium, it has to be sampled for a long time to allow sufficient accuracy of the averages so that the difference of the two terms is meaningful. Further, the applicability of the gradient method can be questioned since such methods are even problematic in recurrent systems without hidden states since small changes of system parameters (weights) can trigger large changes in the dynamics of the dynamical systems. These problems prevented, until recently, more practical progress in this area. Recently, Hinton and colleagues developed more practical, and biologically more plausible, systems which are described next.

1.2.2 The restricted Boltzmann machine and contrastive Hebbian learning

Training of the Boltzmann machine with the above rule is challenging because the states of the nodes are always changing. Even with the visible states clamped, the

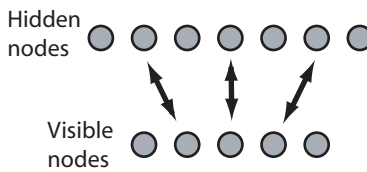


Fig. 1.2 Restricted Boltzmann machine in which recurrences within each later are removed.

states of the hidden nodes are continuously changing for two reasons. First, the update rule is probabilistic, which means that even with constant activity of the visible nodes, hidden nodes receive variable input. Second, the recurrent connections between hidden nodes can change the states of the hidden nodes rapidly and generate rich dynamics in the system. We certainly want to keep the probabilistic update rule since we need to generate different responses of the system in response to sensory data. However, we can simplify the system by eliminating recurrent connections within each layer, although connections between the layers are still bidirectional. While the simplification of omitting collateral connections is potentially severe, much of the abilities of general recurrent networks with hidden nodes can be recovered through the use of many layers which bring back indirect recurrences. A **restricted Boltzmann machine** (RBM) is shown in Fig. 1.2.

When applying the learning rule of eqn 1.7 to one layer of an RBM, we can expect faster convergence of the rule due to the restricted dynamics in the hidden layer. We can also write the learning rule in a slightly different form by using the following procedure. A sensory input state is applied to the input layer, which triggers some probabilistic recognition in the hidden layer. The states of the visible and hidden nodes can then be used to update the expectation value of the correlation between these nodes, $\langle s_i^v s_j^h \rangle^0$, at the initial time step. The pattern in the hidden layer can then be used to approximately reconstruct the pattern of visible nodes. This **alternating Gibbs sampling** is illustrated in Fig. 1.3 for a connection between one visible node and one hidden node, although this learning can be done in parallel for all connections. The learning rule can then be written in form,

$$\Delta w_{ij} \propto \langle s_i^v s_j^h \rangle^0 - \langle s_i^v s_j^h \rangle^\infty. \quad (1.8)$$

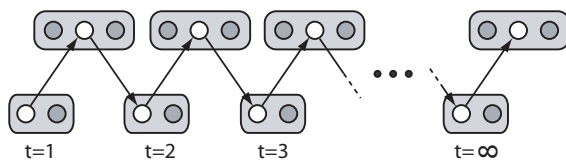


Fig. 1.3 Alternating Gibbs sampling.

Alternating Gibbs sampling becomes equivalent to the Boltzmann machine learning rule (eqn 1.7) when repeating this procedure for an infinite number of time steps, at which point it produces pure fantasies. However, this procedure still requires averaging over long sequences of simulated network activities, and sufficient evaluations of thermal averages can still take a long time. Also, the learning rule of eqn 1.8 does

not seem to correspond to biological learning. While developmental learning also takes some time, it does not seem reasonable that the brain produces and evaluates long sequences of responses to individual sensory stimulations. Instead, it seems more reasonable to allow some finite number of alternations between hidden responses and the reconstruction of sensory states. While this does not formally correspond to the mathematically derived gradient learning rule, it is an important step in solving the learning problem for practical problems, which is a form of **contrastive divergence** introduced by Geoffrey Hinton. It is heuristically clear that such a restricted training procedure can work. In each step we create only a rough approximation of ideal average fantasies, but the system learns the environment from many examples, so that it continuously improves its expectations. While it might be reasonable to use initially longer sequences, as infants might do, Hinton and colleagues showed that learning with only a few reconstructions is able to self-organize the system. The self-organization, which is based on input from the environment, is able to form internal representations that can be used to generate reasonable sensory expectations and which can also be used to recognize learned and novel sensory patterns.

The basic Boltzmann machine with a visible and hidden layer can easily be combined into hierarchical networks by using the activities of hidden nodes in one layer as inputs to the next layer. Hinton and colleagues have demonstrated the power of restricted Boltzmann machines for a number of examples. For example, they applied layered RBMs as auto-encoders where restricted alternating Gibbs sampling was used as pre-training to find appropriate initial internal representations that could be fine-tuned with backpropagation techniques to yield results surpassing support vector machines. However, for our discussions of brain functions it is not even necessary to yield perfect solutions in a machine learning sense, and machines can indeed outperform humans in some classification tasks solved by machine learning methods. For us, it is more important to understand how the brain works.

Simulation 1: Hinton

To illustrate the function of an anticipating brain model, we briefly outline a demonstration by the Hinton group. The online demonstration can be run in a browser from <http://www.cs.toronto.edu/~hinton/adi>, and a stand alone version of this demonstration is available at this book's resource page. MATLAB source code for restricted Boltzmann machines are available at Hinton's home page. An image of the demonstration program is shown in Fig. 1.4. The model consists of a combination of restricted Boltzmann machines and a Helmholtz machine. The input layer is called the **model retina** in the figure, and the system also contains a **recognition-readout-and-stimulation** layer. The model retina is used to apply images of handwritten characters to the system. The recognition-readout-and-stimulation layer is a brain imaging and stimulation device from and to the uppermost RBM layer. This device is trained by providing labels as inputs to the RBM for the purpose of 'reading the mind' of the system and to give it high-level instructions. This device learns to recognize patterns in the uppermost layer and map them to their meaning, as supplied during supervised learning of this device. This is somewhat analogous to **brain-computer interfaces** developed with different brain-imaging devices such as EEG, fMRI, or implanted electrodes. The advantage of the simulated device is that it can read the activity of

every neuron in the upper RBM layer. The device can also be used with the learned connections in the opposite direction to stimulate the upper RBM layer with typical patterns for certain image categories.

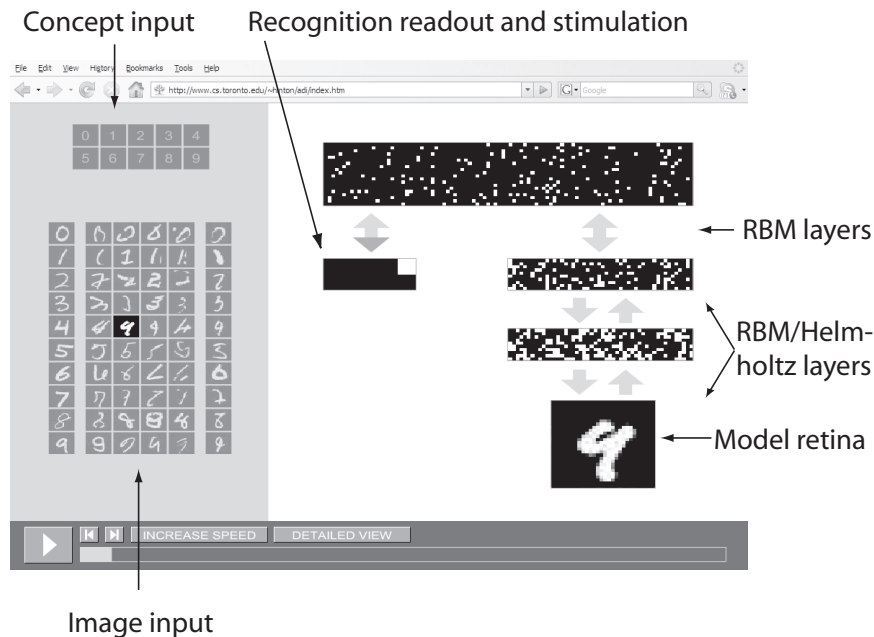


Fig. 1.4 Simulation of restricted Boltzmann machine by Geoffrey Hinton and colleagues, available at www.cs.toronto.edu/~hinton/adi.

The model for this demonstration was trained on images of handwritten numbers from a large database. Some example images can be seen on the left-hand side. All layers of this model were first treated as RBMs with symmetrical weights. Specifically, these were trained by applying images of handwritten characters to the model retina and using three steps of alternating Gibbs sampling for training the different layers. The evolving representations in each layer are thus purely unsupervised. After this basic training, the model was allowed, for fine-tuning purposes, to develop different weight values for the recognition and generative model as in Helmholtz machines with a wake-sleep training algorithm as mentioned above.

The simulations provided by Hinton demonstrate the ability of the system after training. The system can be tested in two ways, either by supplying a handwritten image and asking for recognition, or by asking the system to produce images of a certain letter. These two modes can be initiated by selecting either an image or by selecting a letter category on the left-hand side. In the example shown in Fig. 1.4, we selected an example of an image of the number 4. When running the simulation, this image triggers response patterns in the layers. These patterns change in every time step, due to the probabilistic nature of the updating rule. The recognition read-out of the uppermost layer does, therefore, also fluctuate. In the shown example, the response of the system is 4, but the letter 9 is also frequently reported. This makes sense, as

this image does also look somewhat like the letter 9. A histogram of responses can be constructed when counting the responses over time, which, when properly normalized, corresponds to an estimate of the probability over high-level concepts generated by this sensory state. Thus, this mode tests the recognition ability of the model.

The stimulation device connected to the upper RBM layer allows us to instruct the system to ‘visualize’ specific letters, which corresponds to testing the generative ability of the model. For example, if we ask the system to visualize a letter 4 by evoking corresponding patterns in the upper layer, the system responds with varying images on the model retina. There is not a single right answer, and the answers of the system change with time. In this way, the system produces examples of possible images of letter 4, proportional to some likelihood that these images are encountered in the sensory world on which the system was trained. The probabilistic nature of the system much better resembles human abilities to produce a variety of responses, in contrast to the neural networks that have been popular in the 1980s, so called multilayer perceptrons, which were only able to produce single answers for each input.

Simulation 2: Simplified one layer model

A simplified version of the RBM trained on some letters are included in folder RBM example on the web resource page. The overage reconstruction error and some examples of reconstructions after training are shown in Fig.1.5.

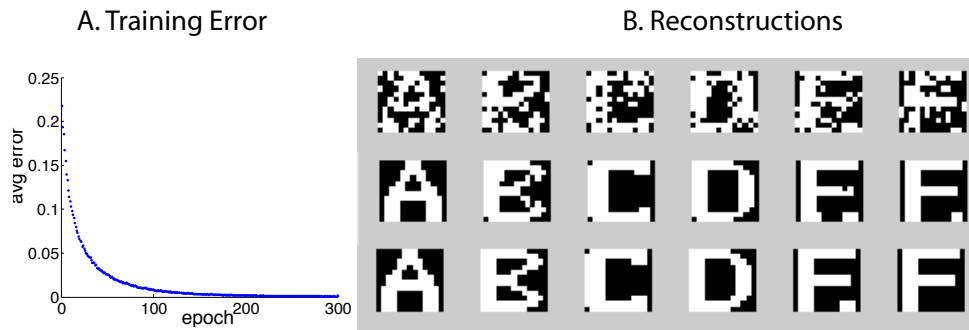


Fig. 1.5 (A) Reconstruction error during training of alphabet letters the letters, and (B) reconstructions after learning.

1.3 Sparse representations

We have here mainly outlined the principles and functionality behind important representational learning machines like RBMs and deep networks. I want to close with a comment on a principle that we have not explored in depth but which I think is essential for learning good representations, that of sparse representations.

We can somewhat distinguish different types of representations in terms of dimensional vectors. Lets illustrate this with binary code. One possible representation is where a class is represented by a single entry like $(0,0,0,1,0,0,0,0,0)$. This is called a

local representation. Somewhat the other extreme is when half of the entries are on, like (0,0,1,1,0,1,0,1,0,1). This is called a **(fully) distributed representation**. Somewhat in the middle, leaning towards the local representation is what we call a **sparse representation**. Thus, sparse representations are characterized by a small number of active cells in a large possible feature vector. This is different than the small distributed representation in an autoencoder. The representation in the smallest layer of the autoencoder is usually a fully distributed representation in a small feature space, whereas a sparse representation would activate only a few features in a large set of possible features (dictionary).

There are different techniques to "encourage" sparse representation in deep networks such as varying the thresholds of the hidden layers (Hinton) or including inhibition between nodes (Hollensen et al). Using sparseness constraints has been proven very useful. For example, it has been shown that reconstruction learning of visual images lead to edge detectors as found in the primary visual cortex only if sparseness constraints are included (Olshausen Field).