Encyclopedia of Machine Learning and Data Mining
DOI 10.1007/978-1-4899-7502-7_31-1

Boltzmann Machines

Geoffrey Hinton*
University of Toronto, Toronto, ON, Canada

Synonyms

Boltzmann machines

Definition

A Boltzmann machine is a network of symmetrically connected, neuron-like units that make stochastic decisions about whether to be on or off. Boltzmann machines have a simple learning algorithm (Hinton and Sejnowski 1983) that allows them to discover interesting features that represent complex regularities in the training data. The learning algorithm is very slow in networks with many layers of feature detectors, but it is fast in "restricted Boltzmann machines" that have a single layer of feature detectors. Many hidden layers can be learned efficiently by composing restricted Boltzmann machines, using the feature activations of one as the training data for the next.

Boltzmann machines are used to solve two quite different computational problems. For a search problem, the weights on the connections are fixed and are used to represent a cost function. The stochastic dynamics of a Boltzmann machine then allow it to sample binary state vectors that have low values of the cost function. For a learning problem, the Boltzmann machine is shown a set of binary data vectors, and it must learn to generate these vectors with high probability. To do this, it must find weights on the connections so that relative to other possible binary vectors, the data vectors have low values of the cost function. To solve a learning problem, Boltzmann machines make many small updates to their weights, and each update requires them to solve many different search problems.

Motivation and Background

The brain is very good at settling on a sensible interpretation of its sensory input within a few hundred milliseconds, and it is also very good, over a much longer timescale, at learning the code that is used to express its interpretations. It achieves both the settling and the learning using spiking neurons which, over a period of a few milliseconds, have a state of 1 or 0. These neurons have intrinsic noise caused by the quantal release of vesicles of neurotransmitter at the synapses between the neurons.

Boltzmann machines were designed to model both the settling and the learning and were based on two seminal ideas that appeared in 1982. Hopfield (1982) showed that a neural network composed of binary units would settle to a minimum of a simple, quadratic energy function

*E-mail: hinton@cs.toronto.edu

DOI 10.1007/978-1-4899-7502-7_31-1

provided that the units were updated asynchronously and the pairwise connections between units were symmetrically weighted. Kirkpatrick et al. (1983) showed that systems that were settling to energy minima could find deeper minima if noise was added to the update rule so that the system could occasionally increase its energy to escape from poor local minima.

Adding noise to a Hopfield net allows it to find deeper minima that represent more probable interpretations of the sensory data. More significantly, by using the right kind of noise, it is possible to make the log probability of finding the system in a particular global configuration be a linear function of its energy. This makes it possible to manipulate log probabilities by manipulating energies, and since energies are simple local functions of the connection weights, this leads to a simple, local learning rule.

Structure of Learning System

The learning procedure for updating the connection weights of a Boltzmann machine is very simple, but to understand why it works, it is first necessary to understand how a Boltzmann machine models a probability distribution over a set of binary vectors and how it samples from this distribution.

The Stochastic Dynamics of a Boltzmann Machine

When unit i is given the opportunity to update its binary state, it first computes its total input, x_i, which is the sum of its own bias, b_i, and the weights on connections coming from other active units:

$$x_i = b_i + \sum_j s_j w_{ij} \tag{1}$$

where w_{ij} is the weight on the connection between i and j and s_j is 1 if unit j is on and 0, otherwise. Unit i then turns on with a probability given by the logistic function

$$\text{prob}(s_i = 1) = \frac{1}{1 + \text{e}^{-x_i}} \tag{2}$$

If the units are updated sequentially in any order that does not depend on their total inputs, the network will eventually reach a Boltzmann distribution (also called its equilibrium or stationary distribution) in which the probability of a state vector, $\mathbf{v}$, is determined solely by the "energy" of that state vector relative to the energies of all possible binary state vectors:

$$P(\mathbf{v}) = \text{e}^{-E(\mathbf{v})} / \sum_{\mathbf{u}} \text{e}^{-E(\mathbf{u})} \tag{3}$$

As in Hopfield nets, the energy of state vector $\mathbf{v}$ is defined as

$$E(\mathbf{v}) = -\sum_i s_i^{\mathbf{v}} b_i - \sum_{i<j} s_i^{\mathbf{v}} s_j^{\mathbf{v}} w_{ij} \tag{4}$$

DOI 10.1007/978-1-4899-7502-7_31-1

where $s_i^{\mathbf{v}}$ is the binary state assigned to unit i by state vector $\mathbf{v}$.

If the weights on the connections are chosen so that the energies of state vectors represent the cost of those state vectors, then the stochastic dynamics of a Boltzmann machine can be viewed as a way of escaping from poor local optima while searching for low-cost solutions. The total input to unit i, x_i, represents the difference in energy depending on whether the unit is off or on, and the fact that unit i occasionally turns on even if x_i is negative means that the energy can occasionally increase during the search, thus allowing the search to jump over energy barriers.

The search can be improved by using simulated annealing. This scales down all of the weights and energies by a factor, T, which is analogous to the temperature of a physical system. By reducing T from a large initial value to a small final value, it is possible to benefit from the fast equilibration at high temperatures and still have a final equilibrium distribution that makes low-cost solutions much more probable than high-cost ones. At a temperature of 0, the update rule becomes deterministic and a Boltzmann machine turns into a Hopfield network.

Learning in Boltzmann Machines Without Hidden Units

Given a training set of state vectors (the data), the learning consists of finding weights and biases (the parameters) that make those state vectors good. More specifically, the aim is to find weights and biases that define a Boltzmann distribution in which the training vectors have high probability. By differentiating (3) and using the fact that

$$\partial E(\mathbf{v})/\partial w_{ij} = -s_i^{\mathbf{v}} s_j^{\mathbf{v}} \tag{5}$$

it can be shown that

$$\left\langle \frac{\partial \log P(\mathbf{v})}{\partial w_{ij}} \right\rangle_{\text{data}} = \langle s_i s_j \rangle_{\text{data}} - \langle s_i s_j \rangle_{\text{model}} \tag{6}$$

where $\langle \cdot \rangle_{\text{data}}$ is an expected value in the data distribution and $\langle \cdot \rangle_{\text{model}}$ is an expected value when the Boltzmann machine samples state vectors from its equilibrium distribution at a temperature of 1. To perform gradient ascent in the log probability that the Boltzmann machine would generate the observed data when sampling from its equilibrium distribution, w_{ij} is incremented by a small learning rate times the RHS of (6). The learning rule for the bias, b_i, is the same as (6), but with s_j omitted.

If the observed data specifies a binary state for every unit in the Boltzmann machine, the learning problem is convex: there are no nonglobal optima in the parameter space. However, sampling from $\langle \cdot \rangle_{\text{model}}$ may involve overcoming energy barriers in the binary state space.

Learning with Hidden Units

Learning becomes much more interesting if the Boltzmann machine consists of some "visible" units whose states can be observed and some "hidden" units whose states are not specified by the observed data. The hidden units act as latent variables (features) that allow the Boltzmann machine to model distributions over visible state vectors that cannot be modeled by direct pairwise interactions between the visible units. A surprising property of Boltzmann machines is that, even

DOI 10.1007/978-1-4899-7502-7_31-1

with hidden units, the learning rule remains unchanged. This makes it possible to learn binary features that capture higher-order structure in the data. With hidden units, the expectation $\langle s_i s_j \rangle_{\text{data}}$ is the average, over all data vectors, of the expected value of $s_i s_j$ when a data vector is clamped on the visible units, and the hidden units are repeatedly updated until they reach equilibrium with the clamped data vector.

It is surprising that the learning rule is so simple because $\partial \log P(\mathbf{v})/\partial w_{ij}$ depends on all the other weights in the network. Fortunately, the locally available difference in the two correlations in (6) tells w_{ij} everything it needs to know about the other weights. This makes it unnecessary to explicitly propagate error derivatives, as in the backpropagation algorithm.

Different Types of Boltzmann Machine

The stochastic dynamics and the learning rule can accommodate more complicated energy functions (Sejnowski 1986). For example, the quadratic energy function in (4) can be replaced by an energy function that has typical term $s_i s_j s_k w_{ijk}$. The total input to unit i that is used in the update rule must then be replaced by

$$x_i = b_i + \sum_{j<k} s_j s_k w_{ijk}. \tag{7}$$

The only change in the learning rule is that $s_i s_j$ is replaced by $s_i s_j s_k$.

Boltzmann machines model the distribution of the data vectors, but there is a simple extension, the "conditional Boltzmann machine" for modeling conditional distributions (Ackley et al. 1985). The only difference between the visible and the hidden units is that when sampling $\langle s_i s_j \rangle_{\text{data}}$, the visible units are clamped and the hidden units are not. If a subset of the visible units are also clamped when sampling $\langle s_i s_j \rangle_{\text{model}}$, this subset acts as "input" units and the remaining visible units act as "output" units. The same learning rule applies, but now it maximizes the log probabilities of the observed output vectors conditional on the input vectors.

Instead of using units that have stochastic binary states, it is possible to use "mean field" units that have deterministic, real-valued states between 0 and 1, as in an analog Hopfield net. Equation (2) is used to compute an "ideal" value for a unit's state, given the current states of the other units, and the actual value is moved toward the ideal value by some fraction of the difference. If this fraction is small, all the units can be updated in parallel. The same learning rules can be used by simply replacing the stochastic, binary values by the deterministic real values (Peterson and Anderson 1987), but the learning algorithm is hard to justify and the mean field nets have problems in modeling multimodal distributions.

The binary stochastic units used in Boltzmann machines can be generalized to "softmax" units that have more than two discrete values, Gaussian units whose output is simply their total input plus Gaussian noise, binomial units, Poisson units, and any other type of unit that falls in the exponential family (Welling et al. 2005). This family is characterized by the fact that the adjustable parameters have linear effects on the log probabilities. The general form of the gradient required for learning is simply the change in the sufficient statistics caused by clamping data on the visible units.

DOI 10.1007/978-1-4899-7502-7_31-1

The Speed of Learning

Learning is typically very slow in Boltzmann machines with many hidden layers because large networks can take a long time to approach their equilibrium distribution, especially when the weights are large and the equilibrium distribution is highly multimodal, as it usually is when the visible units are unclamped. Even if samples from the equilibrium distribution can be obtained, the learning signal is very noisy because it is the difference of two sampled expectations. These difficulties can be overcome by restricting the connectivity, simplifying the learning algorithm, and learning one hidden layer at a time.

Restricted Boltzmann Machines

A restricted Boltzmann machine (Smolensky 1986) consists of a layer of visible units and a layer of hidden units with no visible-visible or hidden-hidden connections. With these restrictions, the hidden units are conditionally independent given a visible vector, so unbiased samples from $\langle s_i s_j \rangle_{\text{data}}$ can be obtained in one parallel step. To sample from $\langle s_i s_j \rangle_{\text{model}}$ still requires multiple iterations that alternate between updating all the hidden units in parallel and updating all of the visible units in parallel. However, learning still works well if $\langle s_i s_j \rangle_{\text{model}}$ is replaced by $\langle s_i s_j \rangle_{\text{reconstruction}}$ which is obtained as follows:

1. Starting with a data vector on the visible units, update all of the hidden units in parallel.
2. Update all of the visible units in parallel to get a "reconstruction."
3. Update all of the hidden units again.

This efficient learning procedure approximates gradient descent in a quantity called "contrastive divergence" and works well in practice (Hinton 2002).

Learning Deep Networks by Composing Restricted Boltzmann Machines

After learning one hidden layer, the activity vectors of the hidden units, when they are being driven by the real data, can be treated as "data" for training another restricted Boltzmann machine. This can be repeated to learn as many hidden layers as desired. After learning multiple hidden layers in this way, the whole network can be viewed as a single, multilayer generative model, and each additional hidden layer improves a lower bound on the probability that the multilayer model would generate the training data (Hinton et al. 2006).

Learning one hidden layer at a time is a very effective way to learn deep neural networks with many hidden layers and millions of weights. Even though the learning is unsupervised, the highest-level features are typically much more useful for classification than the raw data vectors. These deep networks can be fine-tuned to be better at classification or dimensionality reduction using the backpropagation algorithm (Hinton and Salakhutdinov 2006). Alternatively, they can be fine-tuned to be better generative models using a version of the "wake-sleep" algorithm Hinton et al. (2006).

DOI 10.1007/978-1-4899-7502-7_31-1

Relationships to Other Models

Boltzmann machines are a type of Markov random field (see ▸ Graphical Models), but most Markov random fields have simple, local interaction weights which are designed by hand rather than being learned. Boltzmann machines are also like Ising models, but Ising models typically use random or hand-designed interaction weights. The search procedure for Boltzmann machines is an early example of Gibbs sampling, a ▸ Markov chain Monte Carlo method which was invented independently (Geman and Geman 1984) and was also inspired by simulated annealing.

Boltzmann machines are a simple type of undirected graphical model. The learning algorithm for Boltzmann machines was the first learning algorithm for undirected graphical models with hidden variables (Jordan 1998). When restricted Boltzmann machines are composed to learn a deep network, the top two layers of the resulting graphical model form an undirected Boltzmann machine, but the lower layers form a directed acyclic graph with directed connections from higher layers to lower layers (Hinton et al. (2006)).

Conditional random fields (Lafferty et al. 2001) can be viewed as simplified versions of higher-order, conditional Boltzmann machines in which the hidden units have been eliminated. This makes the learning problem convex but removes the ability to learn new features.

Recommended Reading

Ackley D, Hinton G, Sejnowski T (1985) A Learning algorithm for Boltzmann machines. Cognit Sci 9(1):147–169

Geman S, Geman D (1984) Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans Pattern Anal Mach Intell 6(6):721–741

Hopfield JJ (1982) Neural networks and physical systems with emergent collective computational abilities. Proc Natl Acad Sci U S A 79:2554–2558

Hinton GE (2002) Training products of experts by minimizing contrastive divergence. Neural Comput 14(8):1711–1800

Hinton GE, Osindero S, Teh YW (2006) A fast learning algorithm for deep belief nets. Neural Comput 18:1527–1554

Hinton GE, Salakhutdinov RR (2006) Reducing the dimensionality of data with neural networks. Science 313:504–507

Hinton GE, Sejnowski TJ (1983) Optimal perceptual inference. In: Proceedings of the IEEE conference on computer vision and pattern recognition, Washington, DC, pp 448–453

Jordan MI (1998) Learning in graphical models. MIT, Cambridge

Kirkpatrick S, Gelatt DD, Vecci MP (1983) Optimization by simulated annealing. Science 220(4598):671–680

Lafferty J, McCallum A, Pereira F (2001) Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th international conference on machine learning, Williamstown, pp 282–289. Morgan Kaufmann, San Francisco

Peterson C, Anderson JR (1987) A mean field theory learning algorithm for neural networks. Complex Syst 1(5):995–1019

Sejnowski TJ (1986) Higher-order Boltzmann machines. AIP Conf Proc 151(1):398–403

DOI 10.1007/978-1-4899-7502-7_31-1

Smolensky P (1986) Information processing in dynamical systems: foundations of harmony theory. In: Rumelhart DE, McClelland JL (eds) Parallel distributed processing. Foundations, vol 1. MIT, Cambridge, pp 194–281 Press.

Welling M, Rosen-Zvi M, Hinton GE (2005) Exponential family harmoniums with an application to information retrieval. In: Lawrence K. Saul, Yair Weiss and Leon Bottou (eds) Advances in neural information processing systems, vol 17. MIT, Cambridge, pp 1481–1488