Boltzmann Machines

Neuroscience

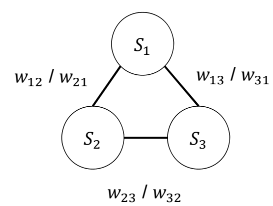
- Donald Hebb (Long Term Potentiation): Neurons that fire together, wire together. Neurons that fire out of sync, fail to link.
- Associative memory: memories stored and retrieved as a string of associations
- Learning involves forming these associations

Boltzmann Machines

- The Hopfield network was developed to model associative memory
- Boltzmann machine is a stochastic version of a Hopfield network
- Boltzmann machine is also closely related to the Ising model from physics

Boltzmann Machines

- Fully-connected network i.e. each node is connected to all other nodes
- Node state S_i is binary (e.g. 0 for "off" and 1 for "on")
- Weight w_{ij} between node i and node j. The weight is symmetric i.e. $w_{ij} = w_{ji}$



Think of the weights as:
- Excitatory constraints (node i is similar to node j) if positive
- Inhibitory constraints (node i is not similar to node j) if negative

Boltzmann Machines

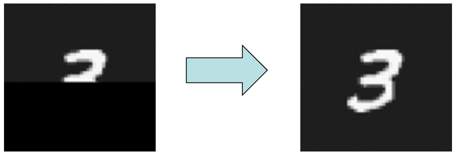
Used to solve two different problems:

1. Search problem: weights are fixed, need to find values of the states that minimize the "energy" of the whole system
2. Learning problem: given training data, learn the weights

5

The Search Problem

- Example use case: fix a corrupted or partially hidden image
- Use a Boltzmann machine trained on images of 3



6

The Search Problem

- The energy of a state corresponds to how well it satisfies these constraints
- Want to figure out which of $S_i = 0$ or $S_i = 1$ has lower energy based on the current states of the other nodes (we call this updating the state)
- Input z_i to a node i:

$$z_i = \sum_j s_j w_{ij} + b_i$$

Bias term. Side note: some papers call this a threshold $\theta_i = -b_i$

- Stochastic update for the state of node i:

$$P(S_i = 1) = \frac{1}{1 + e^{-z_i}}$$

7

The Search Problem

- We want to compute the total energy of the Boltzmann Machine and minimize it
- The energy of a state vector $\boldsymbol{s} = (s_1, \dots, s_n)$ is

$$E(\boldsymbol{s}) = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}$$

- Probability of a state vector is:

$$P(\boldsymbol{s}) = \frac{e^{-E(\boldsymbol{s})}}{\sum_{\boldsymbol{s}'} e^{-E(\boldsymbol{s}')}} = \frac{1}{Z} e^{-(-\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij})}$$

8

The Search Problem

- If you update the states sequentially in any order not dependent on their total inputs, you get to an equilibrium
- Relative probability of two global states $\boldsymbol{s}$ and $\boldsymbol{s}'$ follows a Boltzmann distribution:

$$\frac{p_{\boldsymbol{s}}}{p_{\boldsymbol{s}'}} = e^{-(E_{\boldsymbol{s}}-E_{\boldsymbol{s}'})/T}$$

- Note: because the update is stochastic, you can "jump" out of local minima

The Search Problem

- Can improve search with simulated annealing

$$P(S_i = 1) = \frac{1}{1 + e^{-z_i/T}}$$

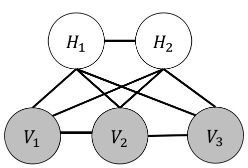
- Temperature parameter T starts large and is reduced (annealed) over time
  – Higher temperature: more likely to go to a higher energy state (gets you out of local minima), can nudge you to a global optimum
  – Lower temperature: favors low energy states and converges faster

Learning

During learning, you can have hidden states ($\mathbf{H}$) and visible states ($\boldsymbol{V}$)

$$P(\boldsymbol{V}) = \sum_{\boldsymbol{H}} P(\boldsymbol{H}, \boldsymbol{V}) = \frac{1}{Z}\sum_{\boldsymbol{H}} e^{-E(\boldsymbol{H},\boldsymbol{V})/T} = \frac{\sum_H e^{-E(\boldsymbol{H},\boldsymbol{V})/T}}{\sum_H \sum_V e^{-E(\boldsymbol{H},\boldsymbol{V})/T}}$$

Where $E(\boldsymbol{H}, \boldsymbol{V}) = -\sum_i s_i b_i - \sum_{i<j} s_i s_j w_{ij}$



The s_i notation here means take the ith node from the joint vector of hidden and visible nodes (H, V)

Learning

We distinguish between $P_{data}(h, v)$ and $P_{model}(v)$.

<u>Data-dependent term</u>

$P_{data}(\boldsymbol{h}, \boldsymbol{v}) = P(\boldsymbol{h}|\boldsymbol{v})P_{data}(\boldsymbol{v})$

where $P_{data}(\boldsymbol{v}) = \frac{1}{N}\sum_k \delta(\boldsymbol{v} - \boldsymbol{v}^k)$

This is the empirical distribution. The δ symbol is a Dirac delta meaning $\delta(\boldsymbol{v} - \boldsymbol{v}^k)$ =1 if $(v = v^k)$ and 0 otherwise.

<u>Data-independent term</u>

$P_{model}(\boldsymbol{v}) = \frac{1}{Z}\sum_h e^{-E(\boldsymbol{h},\boldsymbol{v})/T}$

Learning

To train, you minimize D:

$$D = \sum_{\boldsymbol{v}} P_{model}(\boldsymbol{v}) log \frac{P_{data}(\boldsymbol{h}, \boldsymbol{v})}{P_{model}(\boldsymbol{v})}$$

You can minimize this with gradient descent using the gradient update:

$$\frac{\partial D}{\partial w_{ij}} = -\frac{1}{T}\left(E_{P_{data}(\boldsymbol{h},\boldsymbol{v})}\left[s_i s_j\right] - E_{P_{model}(\boldsymbol{v})}\left[s_i s_j\right]\right)$$

Note: Both these probability distributions must be measured at equilibrium

13

Learning

(From Hinton's Coursera notes Lecture 12.1, Slide 7)

A learning algorithm (from Hinton and Sejnowski (1983)

Computing $E_{P_{data}(\boldsymbol{h},\boldsymbol{v})}\left[s_i s_j\right]$

- Fix visible nodes to their values from the training vector
- Set hidden nodes to random states
- Update hidden nodes one at a time until equilibrium reached at $T = 1$.
- Sample $s_i s_j$ for every connected pair of nodes
- Repeat for all training data vectors and average

Computing $E_{P_{model}(\boldsymbol{v})}\left[s_i s_j\right]$

- Set all nodes to random states
- Update nodes one at a time until equilibrium reached at $T = 1$.
- Sample $s_i s_j$ for every connected pair of nodes
- Repeat many times and average

Note: this algorithm is very old and inefficient. Over time, researchers have developed more efficient methods

14

Learning

Exact minimization is intractable

- Computing $E_{P_{data}(\boldsymbol{h},\boldsymbol{v})}\left[s_i s_j\right]$ is exponential in the number of hidden units
- Computing $E_{P_{model}(\boldsymbol{v})}\left[s_i s_j\right]$ is exponential in the number of hidden and visible units.

Convexity

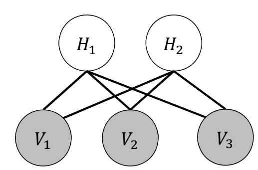
- No hidden units: the minimization of D is concave and gradient descent converges to a global minimum
- Has hidden units: the minimization of D is non-concave, gets stuck in local minima

15

Restricted Boltzmann Machines

RBMs (Smolensky 1986) restrict connectivity to a bipartite graph to make inference and learning easier:

- Only one layer of hidden states
- No visible-visible state edges
- No hidden-hidden state edges



16

Restricted Boltzmann Machines

- Makes hidden nodes independent of each other given visible nodes
- Allows parallel computation of hidden nodes
- Reaches equilibrium in one step when visible units clamped

$$P(H_j = 1) = \frac{1}{1 + e^{-b_j - \sum_{i \in V} v_i w_{ij}}}$$

- Can quickly compute $E_{P_{data}(\boldsymbol{h},\boldsymbol{v})}[v_i h_j]$

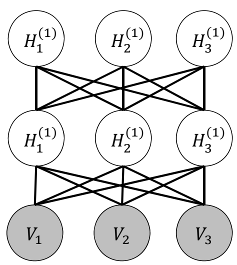
Restricted Boltzmann Machines

Many efficient learning algorithms based on MCMC have been proposed over the years eg.

- Tieleman, T. (2008). Training restricted Boltzmann machines using approximations to the likelihood gradient. In Machine Learning: Proceedings of the Twenty-First International Conference, (pp. 1033-1040).
- Carreira-Perpiñán, M. A. and Hinton, G. (2005). On contrastive divergence learning. *Artificial Intelligence and Statistics*.
- Neal, R. M. (1992). Connectionist learning of belief networks. Artificial Intelligence, 56(1), 71-113.

Restricted Boltzmann Machines

- RBMs can be stacked on top of each other
- First layer: train a RBM on training data (visible nodes)
- Second layer: take hidden layer from first RBM and treat like visible layer for training 2nd RBM and so on…
- Final step uses supervised learning (e.g. backprop) to fine tune the network



Restricted Boltzmann Machines

What does this greedy, layer-wise training do?

- Effectively pre-training each layer
- RBM is a generative model that is trained in an unsupervised manner
- Instead of random weight initialization, RBM initializes them in a more informative way
- Leads to better learned representations

Restricted Boltzmann Machines

- Past attempts at building deep neural networks failed due to the vanishing gradient problem
- Stacked RBMs (trained with Contrastive Divergence) did not experience vanishing gradients and were fast to train
- Lead to some of the earliest deep architectures (e.g. Deep Belief Nets, Deep Boltzmann Machines)

Resources

- Hinton, G. E. and Sejnowski, T. J. (1983). Optimal Perceptual Inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, (pp. 448-453).
- Hinton, G. E. and Sejnowski, T. J. (1983). Analyzing Cooperative Computation. In Proceedings of the 5th Annual Conference of the Cognitive Science Society.
- Hinton's Boltzmann Machines Coursera lecture (https://www.youtube.com/watch?v=MMBX--6_hA4)
- Hinton's Restricted Boltzmann Machine Coursera lecture (https://www.youtube.com/watch?v=JvF3gninXi8&index=57&list=PLoRl3Ht4JOcdU872GhiYWf6jwrk_SNhz9)
- Boltzmann Machines notes by Geoff Hinton (https://www.cs.toronto.edu/~hinton/csc321/readings/boltz321.pdf)
- Chapter 14 of Rojas' book on Neural Networks (https://page.mi.fu-berlin.de/rojas/neural/chapter/K14.pdf)