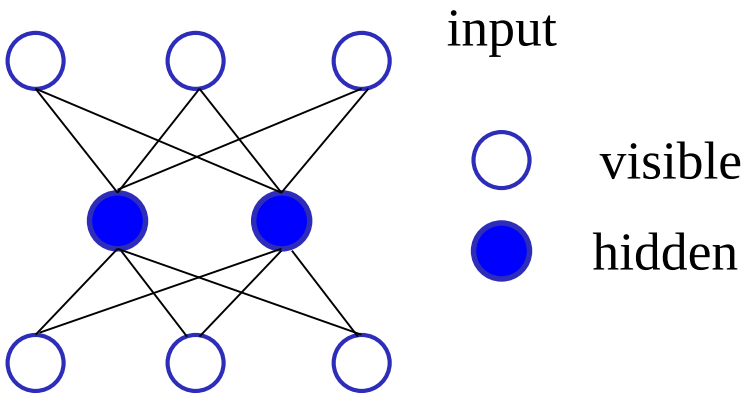
CSE 5526: Introduction to Neural Networks

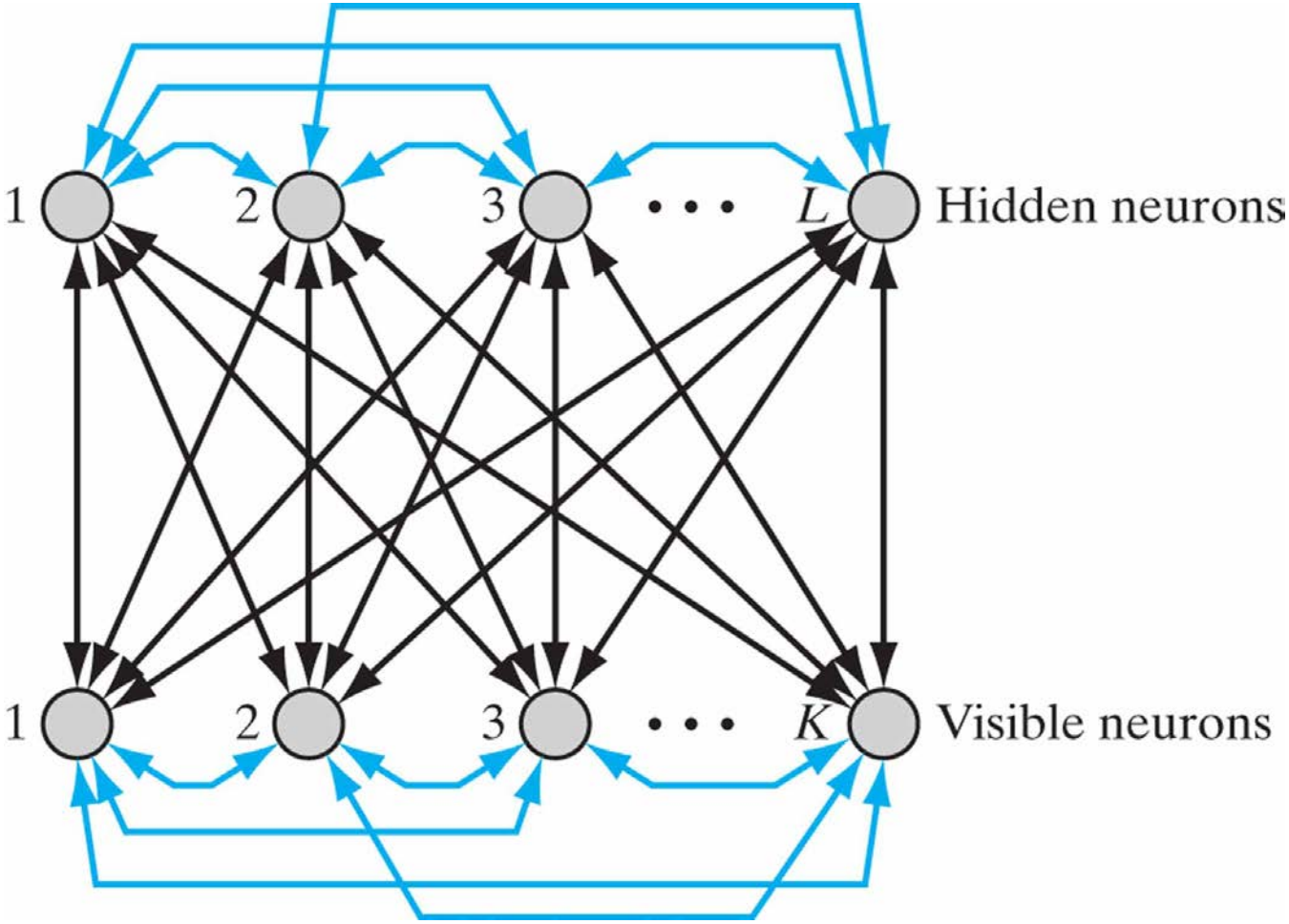
Boltzmann Machines

Introduction

- Boltzmann machine is a stochastic learning machine that consists of visible and hidden units and symmetric connections
- The network can be layered and visible units can be either input or output



Another architecture



Stochastic neurons

- Definition:

$$x_i = \begin{cases} 1 & \text{with prob. } \varphi(v_i) \\ -1 & \text{with prob. } 1 - \varphi(v_i) \end{cases}$$

- For symmetric connections, i.e. $w_{ji} = w_{ij}$, there is an energy function:

$$E(\mathbf{x}) = -\frac{1}{2}\sum_i \sum_j w_{ji} x_i x_j$$

Boltzmann-Gibbs distribution

- Consider a physical system with a large number of states. Let p_i denote the prob. of occurrence of state i of the stochastic system. Let E_i denote the energy of state i
- From statistical mechanics, when the system is in thermal equilibrium, it satisfies the Boltzmann-Gibbs distribution

$$p_i = \frac{1}{Z}\exp(-\frac{E_i}{T})$$

and

$$Z = \sum_i \exp(-\frac{E_i}{T})$$

- Z is called the partition function, and T is called the temperature

Remarks

- Lower energy states have higher prob. of occurrences
- As T decreases, the prob. is concentrated on a small subset of low energy states

Boltzmann machines

- Boltzmann machines use stochastic neurons
- For neuron i:

$$v_i = \sum_j w_{ij} x_j$$

  - A bias term can be included

$$\varphi(v) = \frac{1}{1 + \exp(-\frac{2v}{T})}$$

Objective of Boltzmann machines

- The primary goal of Boltzmann learning is to produce a network that correctly models the probability distribution of visible neurons
  - Such a net can be used for pattern completion, part of associative memory, among other tasks

Positive and negative phases

- Divide the entire net into the subset $\mathbf{x}_\alpha$ of visible units and $\mathbf{x}_\beta$ of hidden units. There are two phases to the learning process:
  1. **Positive phase**: the net operates in the "clamped" condition, where visible units take on training patterns with the desired prob. distribution
  2. **Negative phase**: the net operates freely without the influence of external input

Positive and negative phases (cont.)

- In the negative phase, the prob. of having visible units in state α is

$$P(\mathbf{x}_\alpha) = \frac{1}{Z} \sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right)$$

which is the marginal distribution

Learning

- By adjusting the weight vector **w**, the objective of Boltzmann learning is to maximize the likelihood of the visible units taking on training patterns during the negative phase
- Assuming that each pattern of the training sample is statistically independent. The log prob. of the training sample is:

$$L(\mathbf{w}) = \log \prod_{\mathbf{x}_\alpha} P(\mathbf{x}_\alpha) = \sum_{\mathbf{x}_\alpha} \log P(\mathbf{x}_\alpha)$$

$$= \sum_{\mathbf{x}_\alpha} \left[\log \sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right) - \log \sum_{\mathbf{x}} \exp\left(-\frac{E(\mathbf{x})}{T}\right) \right]$$

Learning (cont.)

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}}$$

$$= \sum_{\mathbf{x}_\alpha} \left[\frac{\partial}{\partial w_{ji}} \log \sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right) - \frac{\partial}{\partial w_{ji}} \log \sum_{\mathbf{x}} \exp\left(-\frac{E(\mathbf{x})}{T}\right) \right]$$

$$= \sum_{\mathbf{x}_\alpha} \left[\frac{-\frac{1}{T}\sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right) \frac{\partial E(\mathbf{x})}{\partial w_{ji}}}{\sum_{\mathbf{x}_\beta} \exp\left(-\frac{E(\mathbf{x})}{T}\right)} + \frac{\frac{1}{T}\sum_{\mathbf{x}} \exp\left(-\frac{E(\mathbf{x})}{T}\right) \frac{\partial E(\mathbf{x})}{\partial w_{ji}}}{\sum_{\mathbf{x}} \exp\left(-\frac{E(\mathbf{x})}{T}\right)} \right]$$

since $\frac{\partial E(\mathbf{x})}{\partial w_{ji}} = -x_j x_i$

Gradient of log probability

- We have

$$\frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \frac{1}{T}\sum_{\mathbf{x}_\alpha}\left[\sum_{\mathbf{x}_\beta}\frac{\exp\left(-\frac{E(\mathbf{x})}{T}\right)x_j x_i}{\sum_{\mathbf{x}_\beta}\exp\left(-\frac{E(\mathbf{x})}{T}\right)} - \frac{\sum_{\mathbf{x}}\exp\left(-\frac{E(\mathbf{x})}{T}\right)x_j x_i}{Z}\right]$$

$$= \frac{1}{T}\sum_{\mathbf{x}_\alpha}\left[\sum_{\mathbf{x}_\beta}\frac{\exp\left(-\frac{E(\mathbf{x})}{T}\right)x_j x_i}{\sum_{\mathbf{x}_\beta}\exp\left(-\frac{E(\mathbf{x})}{T}\right)} - \sum_{\mathbf{x}} P(\mathbf{x})x_j x_i\right]$$

training patterns

constant w.r.t. $\Sigma_{\mathbf{x}_\alpha}$.

Gradient of log probability (cont.)

$$= \frac{1}{T} \sum_{\mathbf{x}_\alpha} \left[\sum_{\mathbf{x}_\beta} P(\mathbf{x}_\beta | \mathbf{x}_\alpha) x_j x_i - < x_j x_i > \right]$$

$$= \frac{1}{T} (\rho_{ji}^{+} - \rho_{ji}^{-})$$

where $\rho_{ji}^{+} = \sum_{\mathbf{x}_\alpha} \sum_{\mathbf{x}_\beta} P(\mathbf{x}_\beta | \mathbf{x}_\alpha) x_j x_i$

is the mean correlation between neurons *i* and *j* when the machine operates in the positive phase

$\rho_{ji}^{-} = < x_j x_i >$ is the mean correlation between *i* and *j* when the machine operates in the negative phase

Maximization of $L(\mathbf{w})$

- To maximize $L(\mathbf{w})$, we use gradient ascent:

$$\triangle w_{ji} = \varepsilon \frac{\partial L(\mathbf{w})}{\partial w_{ji}} = \frac{\varepsilon}{T}\left(\rho_{ji}{}^{+} - \rho_{ji}{}^{-}\right)$$
$$= \eta\left(\rho_{ji}{}^{+} - \rho_{ji}{}^{-}\right)$$

where the learning rate η incorporates the temperature T

- **Remarks**: The Boltzmann learning rule is a local rule, concerning only “presynaptic” and “ postsynaptic” neurons

Gibbs sampling and simulated annealing

- Consider a *K*-dimensional random vector $\mathbf{x} = (x_1, \ldots, x_K)^T$. Suppose we know the conditional distribution x_k given the values of the remaining random variables. Gibbs sampling operates in iterations

Gibbs sampling

- For iteration n:

  $x_1(n)$ is drawn from the conditional distribution of x_1 given $x_2(n-1), x_3(n-1),\ldots, x_K(n-1)$

  …

  $x_k(n)$ is drawn from the conditional distribution of x_k given $x_1(n), x_2(n),..., x_{k-1}(n), x_{k+1}(n-1),..., x_K(n-1)$

  …

  $x_K(n)$ is drawn from the conditional distribution of x_K given $x_1(n),..., x_{K-1}(n)$

Gibbs sampling (cont.)

- In other words, each iteration samples a random variable once in the natural order, and newly sampled values are used immediately (i.e., asynchronous sampling)

Prob. of flipping a single neuron

- For Boltzmann machines, each step of Gibbs sampling corresponds to updating a single stochastic neuron
- Equivalently, we can consider the prob. of flipping a single neuron i:

$$P(x_i \to -x_i) = \frac{1}{1 + \exp\left(\frac{\Delta E_i}{T}\right)}$$

where ΔE_i is the energy change due to the flip (proof is a homework problem)

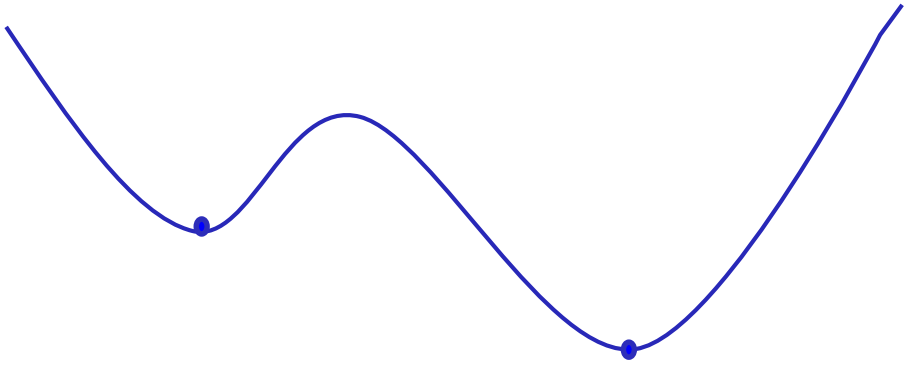
- So a change that decreases the energy is more likely than that increasing the energy

Simulated annealing

- As the temperature T decreases, the average energy of a stochastic system tends to decrease. It reaches the global minimum as $T \rightarrow 0$
- So for optimization problems, we should favor very low temperatures. On the other hand, convergence to thermal equilibrium is very slow at low temperature due to trapping at local minima
- Simulated annealing is a stochastic optimization technique that gradually decreases T. In this case, the energy is interpreted as the cost function and the temperature as a control parameter

Simulated annealing (cont.)

- No guarantee for the global minimum, but higher chances for lower local minima



- Boltzmann machines use simulated annealing to gradually lower T

Simulated annealing algorithm

- The entire algorithm consists of the following nested loops:
  1. Many epochs of adjusting weights
  2. For each epoch, compute $< x_i x_j >$ for each clamped state for a single training pattern, and a free-running, unclamped state
  3. For each state, using simulated annealing by gradually decreasing T
  4. For each T, update the entire net for a number of times with Gibbs sampling
- Boltzmann machines are extremely slow, but potentially effective. Because of its computational complexity, the algorithm has only been applied to toy problems

An example

- The encoder problem (see blackboard)

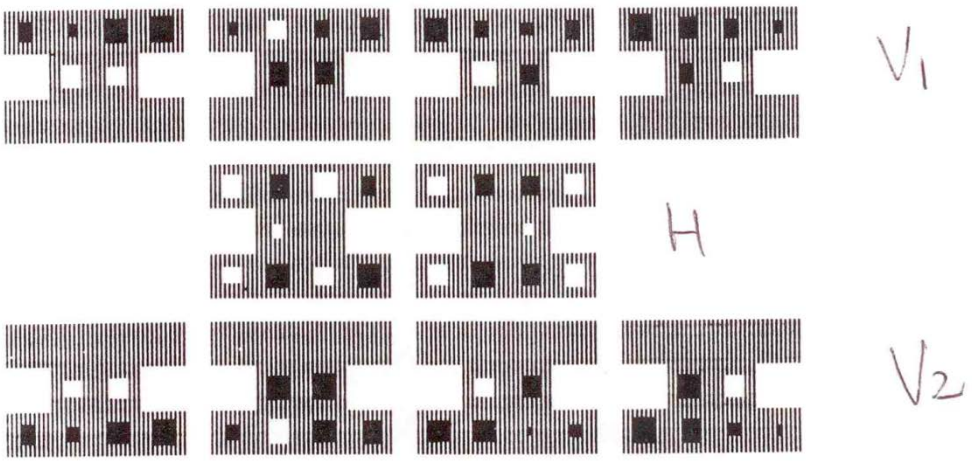


Figure 2 A solution to an encoder problem. The link weights are displayed using a recursive notation. Each unit is represented by a shaded 1-shaped box; from top to bottom the rows of boxes represent groups V_1, H, and V_2. Each shaded box is a map of the entire network, showing the strengths of that unit's connections to other units. At each position in a box, the size of the white (positive) or black (negative) rectangle indicates the magnitude of the weight. In the position that would correspond to a unit connecting to itself (the second position in the top row of the second unit in the top row, for example), the bias is displayed. All connections between units appear twice in the diagram, once in the box for each of the two units being connected. For example, the black square in the top right corner of the left-most unit of V_1 represents the same connection as the black square in the top left corner of the rightmost unit of V_1. This connection has a weight of -30.