Hopfield network – local minima

We look for "deep" minima of E ....

We may get suck in a shallow minimum.

Solution: In every state we allow transition to states with higher energy. This transition has a small probability (which will be higher at the beginning and decrease throughout computation).

Boltzmann activity

Activity: States of neurons initially set to values of $\{-1, 1\}$, i.e., $y_j^{(0)} \in \{-1, 1\}$ for $j \in \{1, \ldots, n\}$.

In the step $t + 1$ update value of a randomly chosen neuron $j \in \{1, \ldots, n\}$ as follows: Compute the inner potential

$$\xi_j^{(t)} = \sum_{i=1}^{n} w_{ji} y_i^{(t)}$$

choose $y_j^{(t+1)} \in \{-1, 1\}$ randomly so that

$$\mathbf{P}\left[y_j^{(t+1)} = 1\right] = \sigma\left(\xi_j^{(t)}\right)$$

where

$$\sigma(\xi) = \frac{1}{1 + e^{-2\xi/T(t)}}$$

The parameter $T(t)$ is called **temperature** in time t.

Temperature and energy

- High temperature $T(t)$ implies that $\mathbf{P}\left[y_j^{(t+1)} = 1\right] \approx \frac{1}{2}$ and thus the network behaves almost randomly.
- Very low temperature $T(t)$ implies that either $\mathbf{P}\left[y_j^{(t+1)} = 1\right] \approx 1$ or $\mathbf{P}\left[y_j^{(t+1)} = 1\right] \approx 0$ depending on whether $\xi_j^{(t)} > 0$ or $\xi_j^{(t)} < 0$. Thus the network behaves almost deterministically (as in the original activity of Hopfield network).

Notes:

- Boltzmann activity = Hopfield activity + random noise,
- energy $E(\vec{y}) = -\frac{1}{2}\sum_{j=1}^{n}\sum_{i=1}^{n} w_{ji}y_jy_i$ may jump to higher levels (with probability depending on the temperature),
- the probability of transition to higher energy decreases exponentially with the size of the "energy jump".

Simulated annealing

The following approach may help to reach deep minima of E:

- Start with higher temperature $T(t)$
- Gradually reduce the temperature, e.g. as follows:
  - $T(t) = \eta^t \cdot T(0)$ where $\eta < 1$ is close to 1
  - or $T(t) = T(0)/\log(1+t)$
- This process resembles annealing used in metallurgy that alters the physical and sometimes chemical properties of a material to increase its ductility and reduce its hardness.
- It also extends physical motivation of Hopfield networks: magnet orientation is now, in addition, influenced by thermal fluctuations.

... and it gets us close to Boltzmann machines.

Boltzmann machine

Architecture:

- Neural network with cycles and symmetric connections (i.e. arbitrary graph)
- N is a *set of all neurons*.
- Denote by ξ_j the inner potential and by y_j the output (i.e. state) of neuron j.
  State of the machine: $\vec{y} \in \{-1, 1\}^{|N|}$.
- Denote by $w_{ji} \in \mathbb{R}$ the weight of the connection from i to j (and thus also from j to i).
- No bias and assume $w_{jj} = 0$ for all $j \in N$.

Boltzmann machine

Activity: States of neurons initially set to values of $\{-1, 1\}$, i.e. $y_j^{(0)} \in \{-1, 1\}$ for $j \in N$.

In the step $t + 1$ do the following:

- Choose a neuron $j \in N$ randomly with the uniform probability.
- Compute the inner potential of j:

$$\xi_j^{(t)} = \sum_{i \in j_{\leftarrow}}^{n} w_{ji} y_i^{(t)}$$

- Choose $y_j^{(t+1)} \in \{-1, 1\}$ *randomly* so that $\mathbf{P}\left[y_j^{(t+1)} = 1\right] = \sigma(\xi_j^{(t)})$ where

$$\sigma(\xi) = \frac{1}{1 + e^{-2\xi / T(t)}}$$

($T(t)$ is a **temperature** at time t.)

Boltzmann machine

- High temperature $T(t)$ implies that $\mathbf{P}\left[y_j^{(t+1)} = 1\right] \approx \frac{1}{2}$ and thus the machine behaves almost randomly.
- Low temperature $T(t)$ means that either $\mathbf{P}\left[y_j^{(t+1)} = 1\right] \approx 1$ or $\mathbf{P}\left[y_j^{(t+1)} = 1\right] \approx 0$ depending on whether $\xi_j^{(t)} > 0$ or $\xi_j^{(t)} < 0$. Then the machine behaves almost deterministically (as the Hopfield network).

Boltzmann machine represents probability

Goal: Construct a network representing a distribution on a set of vectors $\{-1, 1\}^{|N|}$.

Rough idea: Boltzmann machine has states in $\{-1, 1\}^{|N|}$, moves randomly from state to state during computation.

If we let the machine run for sufficiently long time (with a fixed temperature), the *relative frequencies* of visits to states will be independent of the initial state.

We consider these frequencies as probabilities of the states. This gives a probability distribution on $\{-1, 1\}^{|N|}$ represented by the machine.

During learning, a probability distribution on states of $\{-1, 1\}^{|N|}$ will be given, and we adapt weights so that the frequencies match the given probabilities.

Equilibrium

Fix a temperature T (i.e. $T(t) = T$ for $t = 1, 2, \ldots$).

Theorem

For every $\gamma^* \in \{-1, 1\}^{|N|}$ we have that

$$\lim_{t \to \infty} \mathbf{P}\left[\vec{y}(t) = \gamma^*\right] = \frac{1}{Z} e^{-E(\gamma^*)/T}$$

where

$$Z = \sum_{\gamma \in \{-1,1\}^{|N|}} e^{-E(\gamma)/T} \qquad E(\gamma) = -\frac{1}{2} \sum_{i,j} w_{ij} y_i^\gamma y_j^\gamma$$

the Boltzmann distribution.

Define $p_N(\gamma^*) := \lim_{t \to \infty} \mathbf{P}\left[\vec{y}(t) = \gamma^*\right]$ for every $\gamma^* \in \{-1, 1\}^{|N|}$.

Equilibrium probabilities

Note that

- p_N is a probability distribution on $\{-1,1\}^{|N|}$ represented by the machine,
- for a state γ^*, we have that $p_N(\gamma^*)$ is the probability of γ^* in the *thermal equilibrium*,
- $p_N(\gamma^*)$ can be estimated by $\mathbf{P}\left[\vec{y}^{(t^*)} = \gamma^*\right]$ for sufficiently large t^*

  That is, in order to compute $p_N(\gamma^*)$ it is sufficient to simulate a computation several times for t^* steps and then compute the relative frequency of stopping in γ^*.
- By Markov chains theory, $p_N(\gamma^*)$ is the long-run frequency of visits to γ^*.

  This gives an alternative procedure for estimating $p_N(\gamma^*)$: Execute the machine for *very* long time, compute the relative frequency of visits to γ^* along the computation.

Boltzmann machine – learning

To be able to capture more probability distributions, we introduce hidden neurons.

Divide N into two disjoint sets:

- **visible** neurons V
- **hidden** neurons H

For $\alpha \in \{-1, 1\}^{|V|}$ denote

$$p_V(\alpha) = \sum_{\beta \in \{-1,1\}^{|H|}} p_N(\alpha, \beta)$$

the probability that the state of visible neurons in the thermal equilibrium is α.

Our goal is to adapt weights so that p_V corresponds to a given probability distribution on $\{-1, 1\}^{|V|}$.

Boltzmann machine – learning

Learning:
Let p_d be a probability distribution on the states of visible neurons, i.e. on $\{-1, 1\}^{|V|}$.

The distribution p_d can be determined by a sequence of training examples:

$$\mathcal{T} = \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_m$$

then

$$p_d(\alpha) = \#(\alpha, \mathcal{T})/m$$

here $\#(\alpha, \mathcal{T})$ is the number of occurrences of α in $\mathcal{T}$.

Our goal is to find a configuration of the network W such that $p_V \approx p_d$.

Boltzmann machine – learning

A suitable measure of difference between probability distributions p_V and p_d is relative entropy weighted by probabilities of states (Kullback-Leibler divergence):

$$\mathcal{E}(W) = \sum_{\alpha \in \{-1,1\}^{|V|}} p_d(\alpha) \ln \frac{p_d(\alpha)}{p_V(\alpha)}$$

For p_d given by a training set $\mathcal{T} = \vec{x}_1, \vec{x}_2, \ldots, \vec{x}_m$ we have that

minimizing $\mathcal{E}(W)$ is equivalent to maximizing likelihood of $\mathcal{T}$.

Boltzmann machine – learning

Minimize $\mathcal{E}(\vec{w})$ using gradient descent, i.e. compute a sequence of weight matrices: $W^{(0)}, W^{(1)}, \ldots$

- initialise $W^{(0)}$ randomly, close to 0
- in step $t+1$ compute $W^{(t+1)}$ as follows:

$$W_{ji}^{(t+1)} = W_{ji}^{(t)} + \Delta W_{ji}^{(t)}$$

where

$$\Delta W_{ji}^{(t)} = -\varepsilon(t) \cdot \frac{\partial \mathcal{E}}{\partial w_{ji}}(W^{(t)})$$

is the update of the weight w_{ji} in the step $t+1$ and $0 < \varepsilon(t) \leq 1$ is the learning rate in the step $t+1$.

It remains to compute $\frac{\partial \mathcal{E}}{\partial w_{ji}}(W)$.

Boltzmann machine – learning

For sufficiently large t^* (i.e. in thermal equilibrium) we have

$$\frac{\partial \mathcal{E}}{\partial w_{ji}} \approx -\frac{1}{T}\left(\left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{fixed} - \left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{free}\right)$$

- $\left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{fixed}$ is the expected value of $y_j^{(t^*)} y_i^{(t^*)}$ in the thermal equilibrium assuming that values of visible neurons are **fixed** at the beginning of computation according to p_d.
- $\left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{free}$ is the expected value of $y_j^{(t^*)} y_i^{(t^*)}$ in the thermal equilibrium (no values fixed).

Thus

$$\begin{aligned}\Delta w_{ji}^{(\ell)} &= -\varepsilon(\ell) \cdot \frac{\partial \mathcal{E}}{\partial w_{ji}}(W^{(\ell-1)}) \\ &= \frac{\varepsilon(\ell)}{T}\left(\left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{fixed} - \left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{free}\right)\end{aligned}$$

Boltzmann machine – learning

Compute $\left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{fixed}$ as follows:

- Let $\mathcal{Y} := 0$ and do the following q times:
  1. choose $\alpha \in \{-1, 1\}^{|V|}$ randomly according to p_d,
  2. fix values of visible neurons to α and *do not* update them throughout the remaining steps 3. and 4.,
  3. simulate t^* steps, now the current values of neurons j and i are $y_j^{(t^*)}$ and $y_i^{(t^*)}$, respectively,
  4. add $y_j^{(t^*)} y_i^{(t^*)}$ to $\mathcal{Y}$.
- For sufficiently large q, the value $\mathcal{Y}/q$ will be a good estimate of $\left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{fixed}$.

$\left\langle y_j^{(t^*)} y_i^{(t^*)} \right\rangle_{free}$ can be estimated similarly, the only difference is that the steps 1. and 2. are omitted.

Boltzmann machine – learning

For completeness, the analytic version:

$$\left\langle y_i^{(t^*)} y_j^{(t^*)} \right\rangle_{fixed} =$$

$$= \sum_{\alpha \in \{-1,1\}^{|V|}} p_d(\alpha) \sum_{\beta \in \{-1,1\}^{|S|}} \frac{p_N(\alpha,\beta)}{p_V(\alpha)} y_j^{\alpha\beta} y_i^{\alpha\beta}$$

here $y_j^{\alpha\beta}$ is the output of the neuron j in the state (α, β).

$$\left\langle y_i^{(t^*)} y_j^{(t^*)} \right\rangle_{free} = \sum_{\gamma \in \{-1,1\}^{|N|}} p_N(\gamma) y_j^{\gamma} y_i^{\gamma}$$