Neural Networks: Recurrent Networks

Prof. dr. sc. Sven Lončarić

Doc. dr. sc. Tomislav Petković

University of Zagreb

Faculty of Electrical Engineering and Computing

http://www.fer.unizg.hr/en/course/neunet_a

Lecture Overview

- Introduction
- Hopfield Network
  - Discrete Hopfield Network
  - Storage and Retrieval Phase
  - Energy (Lyapunov) Function
  - Spurious States
  - Storage Capacity
- Stochastic Machines
  - Boltzmann Machine
- Simulated Annealing

Introduction

- Multilayer and radial neural networks are well known examples of non-recurrent or feedforward neural networks
  - connections between neurons do not form a cycle
- In this lecture we will discuss recurrent or feedback neural networks
  - connections between neurons form a directed cycle
- We will also discuss stochastic machines which use recurrent connections
  - neurons are stochastic

Introduction

- Key properties of recurrent neural networks
  - connections between neurons form a directed cycle
  - directed cycle includes some form of time delay, i.e. it is a *feedback structure* and not an *algebraic loop*
  - they exhibit dynamic temporal behavior
- Modeling multiple feedback connections is hard
  - network may become unstable
- Usable designs usually limit the number of feedback connections or impose other limits to make analysis feasible

A Short History

- Recurrent neural networks
  - Hopfield neural network (*Hopfield 1982, Little 1974*)
  - Bidirectional associative memory or BAM (*Kosko 1988*)
  - Long short-term memory or LSTM *(Hochreiter & Schmidhuber 1997*)
- Stochastic machines
  - Boltzmann machine (*Hinton, Terrence & Ackley 1984*)
  - Restricted Boltzmann machine or RBM (*Smolensky 1986*)
  - Sigmoid belief networks (*Neal 1992*)
  - Helmholtz machine (*Dayan 1995*, *Hinton 1995*)

Hopfield Neural Network

- A Hopfield neural network consists of a set of neurons and of a corresponding set of unit delays which form a multiple-loop feedback system
  - the output of each neuron is feed to all other neurons
  - there is no self-feedback
- Such architecture exhibits emergent collective properties
  - content addressable memory (CAM)
- Network architecture is simple enough to allow full analytical analysis
  - network stability and memory capacity

Hopfield Network as Content Addressable Memory

- Each memorized sample is stored as a local minimum of network's energy function
  - a memorized sample is called fundamental memory
  - it is an attractor of the energy function
  - each attractor is the lowest point of some valley of the energy function
- As the network operates its state evolves in time from some starting state to the closest attractor
  - All input states which converge to the same attractor from a basin of attraction

Discrete Hopfield Network

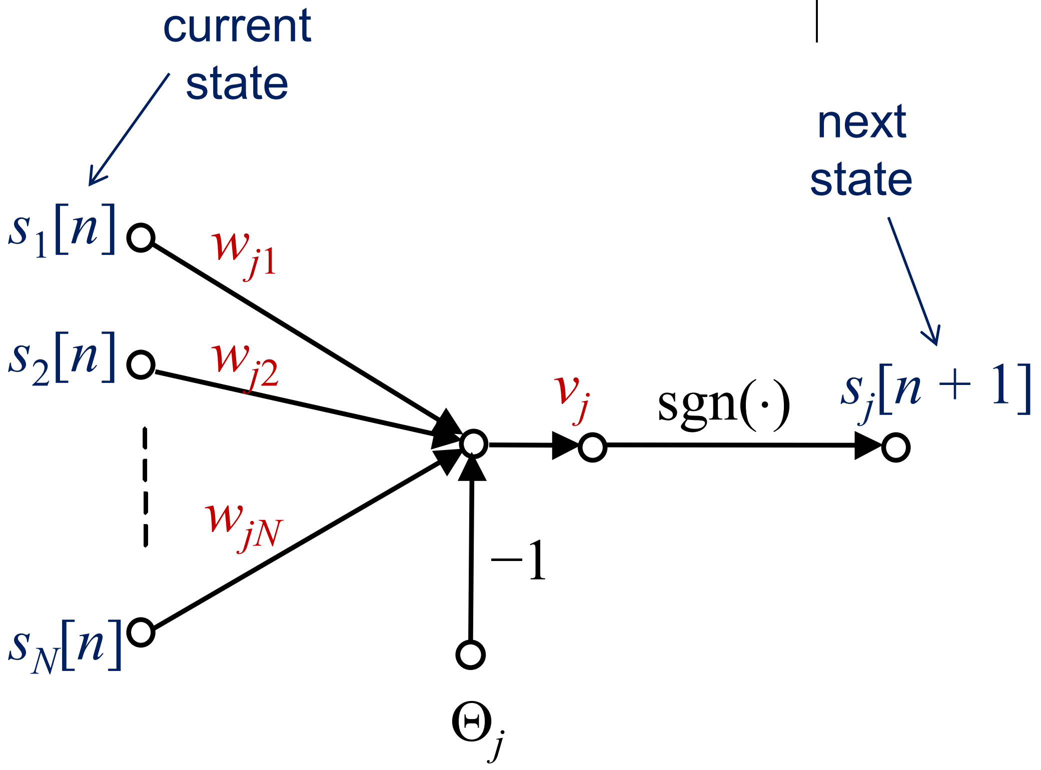
- The discrete Hopfield network is based on the McCulloch-Pitts model
  - the output of each neuron is either $+1$ or -1
- The state of the discrete Hopfield model comprised of N neurons is a state vector $\mathbf{s} = [s_1, s_2, \ldots, s_N]^T$ where s_j is the output of jth neuron
- A feedback from ith to jth neuron has weight w_{ji}
  - w_{ji} *determines the contribution of the state* s_i *to the* jth neuron
  - there is no self-feedback, i.e. $w_{ii} = 0$

Discrete Hopfield Network

- Activation potential v_j of jth neuron is:

$$v_j = \sum_{i=1}^{N} w_{ji} s_i[n] - \Theta_j$$

$$s_j[n+1] = \mathrm{sgn}(v_j)$$



There is no self-feedback, i.e. $w_{jj} = 0$

Discrete Hopfield Network

- The output of jth neuron is

$$s_j = \begin{cases} +1, & v_j > 0 \\ -1, & v_j < 0 \end{cases}$$

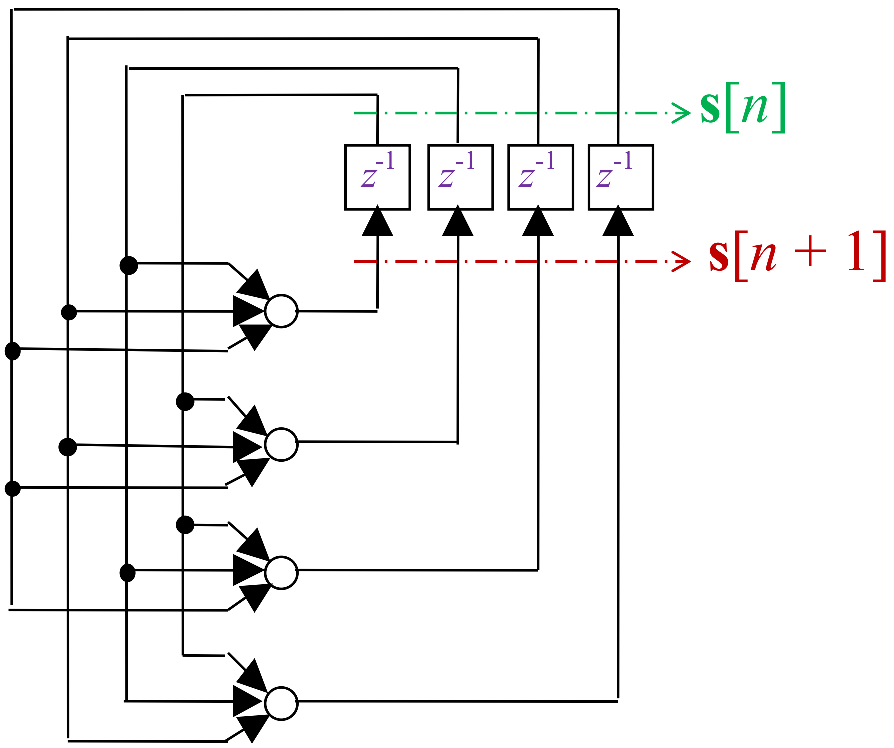
  which for $v_j \neq 0$ may be written as

$$s_j = \operatorname{sgn}(v_j)$$

- If $v_j = 0$ then jth neuron remains in its previous state, i.e. the output does not change
  - this is just one of many possible choices
  - however, this choice makes flow-diagram *symmetrical*

Discrete Hopfield Network

- Architecture of a discrete Hopfield network comprised of four neurons ($N = 4$)
- Blocks denoted using z^{-1} are unit delay blocks
  - output of unit delay blocks is the current state of the network
  - input into unit delay blocks is the next state of the network



Discrete Hopfield Network

- There are two phases of operation of the discrete Hopfield network:
  - storage phase (learning)
  - retrieval phase (exploitation)
- In storage phase data is stored into network memory
- In retrieval phase a vector is imposed on the Hopfield network as its state; the network then retrieves the best stored match after it converges to the closest stable state
  - the retrieval is not instantaneous

Discrete Hopfield Network: Storage Phase

- Let us assume we want to store a set of p N-dimensional vectors:

$$\{\xi_m \mid m = 1, \ldots, p\}$$

- These p vectors are called fundamental memories
- Let $\xi_{m,i}$ be the ith element of the vector ξ_m and let its value be either $+1$ or -1
- According to outer product rule (the generalization of *Hebb's postulate of learning*) the synaptic weight from ith to jth neuron is:

$$w_{ji} = \frac{1}{N} \sum_{m=1}^{p} \xi_{m,j} \xi_{m,i}$$

Discrete Hopfield Network: Storage Phase

- In discrete Hopfield network $w_{ii} = 0$ for each i
  - The neurons have no self-feedback
- Let $\mathbf{W}$ denote the $N \times N$ *synaptic weight matrix*
- Then the synaptic weight matrix $\mathbf{W}$ may be computed using:

$$\mathbf{W} = \frac{1}{N} \sum_{m=1}^{p} \xi_m \xi_m{}^T - \frac{p}{N} \mathbf{I}$$

outer product of the fundamental memory ξ_m

Discrete Hopfield Network: Storage Phase

- From the equation to compute **W** we may reconfirm the following:
  - The output of each neuron in the network is fed back to all other neurons
  - There is no self-feedback in the network, i.e. $w_{ii} = 0$ for each i
  - The synaptic weight matrix **W** is symmetric matrix, i.e. $w_{ji} = w_{ij}$

Discrete Hopfield Network: Retrieval Phase

- In the retrieval phase a N-dimensional vector $\mathbf{x}$ whose elements are either $+1$ or -1 is imposed on the discrete Hopfield network as its state
  - vector $\mathbf{x}$ is called a probe
- The probe vector $\mathbf{x}$ typically represents an incomplete or noisy version of a fundamental memory of the network

Discrete Hopfield Network: Retrieval Phase

- After imposing the probe vector $\mathbf{x}$ as state the network operates using *asynchronous* (or *serial*) dynamical updating rule:

  Each neuron j of the network randomly but at some fixed rate examines its activation potential v_j and updates its output if necessary.
- Therefore:
  - The selection of neuron to perform the update is random
  - The state update is deterministic
- This dynamical updating procedure is repeated until there are no further changes of state to report

Discrete Hopfield Network: Retrieval Phase

- Once there are no changes to the state, we say the network has produced a time-invariant state vector $\mathbf{s}$ which satisfies the stability (or alignment) condition:

$$s_j = \operatorname{sgn}\left(\sum_{i=1}^{N} w_{ji} s_i - \Theta_j\right), \quad j = 1,2,\ldots,N,$$

which in the matrix form becomes

$$\mathbf{s} = \operatorname{sgn}(\mathbf{W}\mathbf{s} - \mathbf{\Theta})$$

where $\mathbf{W}$ is synaptic weight matrix and $\mathbf{\Theta}$ is threshold vector (sometimes a bias $\mathbf{b} = -\mathbf{\Theta}$ is used instead of the threshold $\mathbf{\Theta}$).

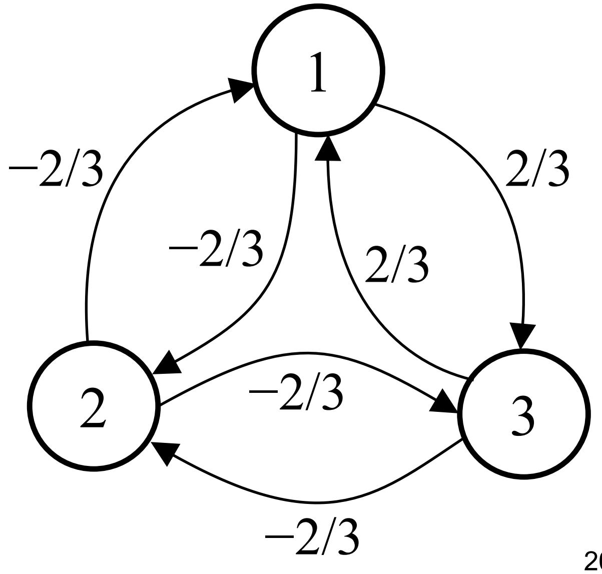
Discrete Hopfield Network: Retrieval Phase

- The state **s** which satisfies the stability condition is called *stable state* or *fixed point* of the state-space of the system
- The discrete Hopfield network always converges to a stable state if *asynchronous* (or *serial*) updating rule is used
- An alternative to asynchronous updating is *synchronous* (or *parallel*) updating where output state is simultaneously computed for all neurons
  - the state will converge either to a fixed point or to a limit cycle of length at most 2

Discrete Hopfield Network: Example 14.2 from Haykin

- Consider a Hopfield network which consists of three neurons and which has synaptic weight matrix $\mathbf{W}$
- The matrix $\mathbf{W}$ is symmetric and has zeros on the main diagonal
- Let threshold vector be zero, i.e. $\Theta = 0$

$$\mathbf{W} = \frac{1}{3}\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}$$



Discrete Hopfield Network: Example 14.2 from Haykin

- With three neurons there are exactly $2^3 = 8$ possible states of the network
- Of these eight states only two states are stable
- Stable states must satisfy the equation

$$\mathbf{y} = \operatorname{sgn}(\mathbf{W}\mathbf{y} - \boldsymbol{\Theta})$$

- The stable states or fixed points are
  - $[+1, -1, +1]^T$
  - $[-1, +1, -1]^T$

Discrete Hopfield Network: Example 14.2 from Haykin

- Let us verify the state $y = [+1, -1, +1]^{\mathrm{T}}$ satisfies the stability condition:

$$\mathbf{Wy} = \frac{1}{3}\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}\begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} = \frac{1}{3}\begin{bmatrix} +4 \\ -4 \\ +4 \end{bmatrix}$$

$$\operatorname{sgn}(\mathbf{Wy}) = \operatorname{sgn}\left(\frac{1}{3}\begin{bmatrix} +4 \\ -4 \\ +4 \end{bmatrix}\right) = \begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix} = \mathbf{y}$$

Discrete Hopfield Network: Example 14.2 from Haykin

- Remaining six states are not stable which is easily verified as they do not satisfy the stability condition
- The network has two stable states
  - $[+1, -1, +1]^T$
  - $[-1, +1, -1]^T$
- We may verify that two stable states are indeed two ($p = 2$) fundamental memories of the network by computing the synaptic weight matrix:

$$\mathbf{W} = \frac{1}{N}\sum_{m=1}^{p} \xi_m \xi_m{}^T - \frac{p}{N}\mathbf{I}$$

Discrete Hopfield Network: Example 14.2 from Haykin

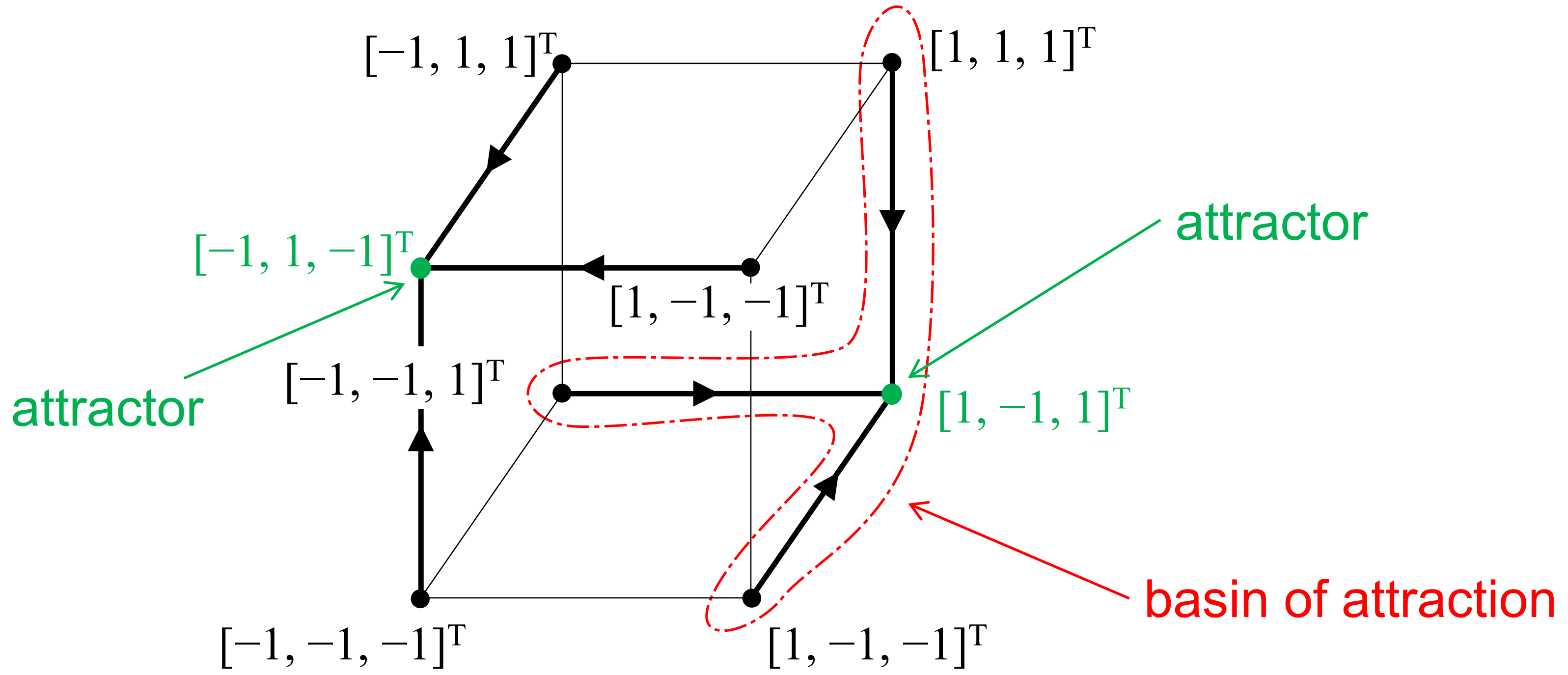
- The synaptic weight matrix is:

$$\mathbf{W} = \frac{1}{3}\begin{bmatrix} +1 \\ -1 \\ +1 \end{bmatrix}\begin{bmatrix} +1 & -1 & +1 \end{bmatrix} + \frac{1}{3}\begin{bmatrix} -1 \\ +1 \\ -1 \end{bmatrix}\begin{bmatrix} -1 & +1 & -1 \end{bmatrix} - \frac{2}{3}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \frac{1}{3}\begin{bmatrix} 0 & -2 & +2 \\ -2 & 0 & -2 \\ +2 & -2 & 0 \end{bmatrix}$$

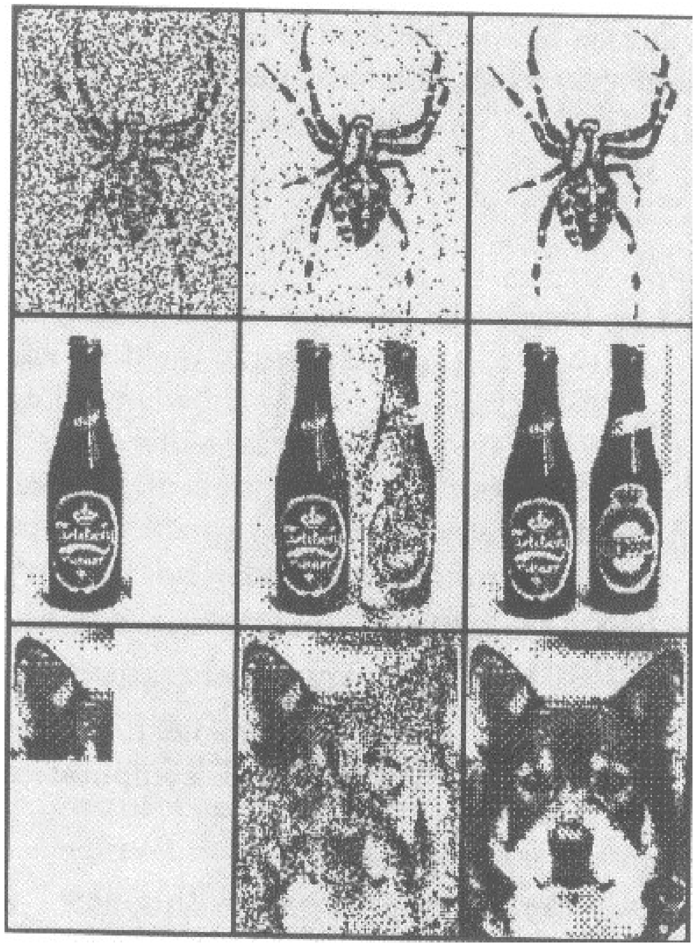
Discrete Hopfield Network: Example 14.2 from Haykin

- The state-space of the Hopfield network
  - green dots denote stable states (fixed points, attractors)
  - arrows show the flow of the network toward stable states



Applications of Hopfield Network

- Application of Hopfield network as an associative image memory which can reconstruct corrupt images
  - figure 2.1 from John A. Hertz, Anders S. Krogh, and Richard G. Palmer. “*Introduction to the theory of neural computation*”, 1991.
  - a sparsely connected Hopfield network storing seven 130×180 binary images was used
  - the middle column shows intermediate states during retrieval phase



Discrete Hopfield Network: The Energy Function

- The energy function of a discrete Hopfield network when $\Theta = 0$ is:

$$E = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1, j\neq i}^{N} w_{ji}s_i s_j = -\tfrac{1}{2}\mathbf{s}^T\mathbf{W}\mathbf{s}$$

- Adding a constant to E (so $E \geq 0$) makes it a Lyapunov function of the Hopfield network
  - Useful in analyzing the stability of the network
- For fixed weights w_{ji} the energy depends on the network state vector $\mathbf{s} = [s_1, s_2, \ldots, s_N]^T$

The Energy Function Always Decreases in Time

- Let us assume that during the retrieval phase using asynchronous update jth neuron changes its state s_j ($+1$ into -1 or -1 into $+1$)
- The change of the total energy caused by this transition of jth neuron is:

$$\Delta E = -\Delta s_j \sum_{i=1, i \neq j}^{N} w_{ji} s_i$$

- A careful analysis of the energy change ΔE reveals that the total energy E of the network always decreases until a local minimum of the energy function is reached
  - Let us demonstrate that in the following two slides

The Energy Function Always Decreases in Time

Let the right term of the product ΔE be positive:

$$\sum_{i=1, i \neq j}^{N} w_{ji} s_i > 0$$

Then the new state s_j of the jth neuron must also be positive:

$$s_j = \operatorname{sgn}\left(\sum_{i=1, i \neq j}^{N} w_{ji} s_i \right) = +1, \quad \Delta s_j = 2$$

As both terms of ΔE are positive the overall change is negative due to the minus sign:

$$\Delta E = -\Delta s_j \sum_{i=1, i \neq j}^{N} w_{ji} s_i < 0$$

The Energy Function Always Decreases in Time

Let the right term of the product ΔE be negative:

$$\sum_{i=1, i \neq j}^{N} w_{ji} s_i < 0$$

Then the new state s_j of the jth neuron must also be negative:

$$s_j = \operatorname{sgn}\left(\sum_{i=1, i \neq j}^{N} w_{ji} s_i \right) = -1, \quad \Delta s_j = -2$$

As both terms of ΔE are negative their product is positive and the overall change is again negative:

$$\Delta E = -\Delta s_j \sum_{i=1, i \neq j}^{N} w_{ji} s_i < 0$$

The Energy Function Always Decreases in Time

- We have shown that ΔE is always negative
- This means the energy function E of a discrete Hopfield network is a decreasing function of time
    - Valid only for asynchronous update
- The energy stops decreasing once a stable point is reached
    - Stable or fixed points are minima of the energy function

Discrete Hopfield Network: The Energy (Lyapunov) Function

- Existence of a Lyapunov function implies stability:

Theorem

Let $s = s_0$ be an equilibrium point for the recurrence equation

$$s(n+1) = f(s(n))$$

where $f{:}D \to \mathbb{R}^N$ is locally Lipschitz in $D \subset \mathbb{R}^N$ and $s_0 \in D$. Suppose there exists a function $E{:}D \to \mathbb{R}$ which is continuous and such that

$$E(0) = 0 \text{ and } E(s - s_0) > 0,\ \forall\ s \in D \backslash \{s_0\}$$

$$E(f(s - s_0)) - E(s - s_0) \leq 0,\ \forall\ s \in D$$

Then the equilibrium point $s = s_0$ is stable.

Hopfield Network as an Associative Memory

- For fixed weights w_{ji} the energy depends on the network state vector $\mathbf{s} = [s_1, s_2, \ldots, s_N]^T$
- This dependency may be thought of as an energy landscape having *valleys* and *hills*
- The local minima in the energy landscape are located in the valleys and act as attractors
- Two conditions are required for the Hopfield network to function as an associative memory:
  - Memorized samples must be attractors
  - Each memorized sample must have a basin of attraction of sufficient size

Hopfield Network as an Associative Memory: Spurious States

- When we try to store many fundamental memories (patterns) ξ_m using Hebb's rule spurious states may occur.
- Spurious states s_{spurious} are states which are not in the set of fundamental memories ξ_m, $m = 1,\ldots, p$, but which correspond to local minima of the energy function
- We usually distinguish between three types of spurious states:
  1. Mirror states
  2. Mixture states
  3. Spin glass states

Hopfield Network as an Associative Memory: Spurious Mirror States

- Energy function of the Hopfield network is symmetrical
  - Changing the sign of all activation potentials v_j yields the same energy
  - If the energy function has a local minima for a fundamental memory ξ then a local minima also exists at $-\xi$
  - $-\xi$ is mirrored or reversed state
- This means the mirror of a fundamental memory also behaves as a fundamental memory
  - This issue may be mitigated in practice: if mirror state is not a fundamental memory we establish a convention how to convert between ξ and $-\xi$

Hopfield Network as an Associative Memory: Spurious Mixture States

- Sums or differences of odd number of fundamental memories may also be local minima of the energy function
- An example is a 3-mixture state $\mathbf{s}_{\text{mix}} = \text{sgn}(\xi_1 + \xi_2 + \xi_3)$
  - On average $\mathbf{s}_{\text{mix}}$ matches the signs of three ξ_i 6 times out of 8
  - Hence $\mathbb{E}[\sum_j \xi_{i,j}\, \mathbf{s}_{\text{mix},j}] = 3N/4$ for $i = 1, 2, 3$
  - If the crosstalk is small (see slide 40) this yields $\mathbf{s}_{\text{mix},j} = \text{sgn}(\sum_i w_{ji}\, \mathbf{s}_{\text{mix},j})$
- Chance of spurious states increases with p
- Self-feedback generates spurious attractors

Hopfield Network as an Associative Memory: Spurious Spin-Glass States

- Spin-Glass states are not correlated with any finite number of fundamental memories
- They are called spin glass states because of close correspondence to *spin glass models* in *statistical mechanics*
- They appear if the number of patterns p to store is large
  - As p becomes large the synaptic weight matrix $\mathbf{W}$ starts to resemble a random matrix
  - Then the term in the sum $\mathrm{sgn}(\sum_i w_{ji}\, \mathbf{x}_{\mathrm{probe},i})$ that favors some particular fundamental memory becomes greatly outweighed by all the others terms
  - The associative memory begins to look like a true *spin glass*

Storage Capacity of the Hopfield Network

- Fundamental memories of a Hopfield network are not always stable
- Spurious states that represent other stable states that are different from the fundamental memories may also appear
- These two phenomena decrease the efficiency of the Hopfield network
- Using a probabilistic approach it is possible to examine the stability of the fundamental memories
  - Note that we do not take into account spurious states
- Such probabilistic analysis reveals the storage capacity of the Hopfield network

Storage Capacity of the Hopfield Network

- Let us examine the stability of the jth neuron for a fundamental memory $\xi_n = [\xi_{n,1}, \xi_{n,2}, \ldots, \xi_{n,N}]^T$
  - jth neuron is often called jth bit
- The stability condition is

$$\xi_{n,j} = \operatorname{sgn}(v_j)$$

$$v_j = \sum_{i=1}^{N} w_{ji}\xi_{n,i}$$

where the weights w_{ji} are given by

$$w_{ji} = \frac{1}{N}\sum_{m=1}^{p} \xi_{m,j}\xi_{m,i}$$

Storage Capacity of the Hopfield Network

- The activation potential of *j*th neuron for the fundamental memory ξ_n is:

$$v_j = \sum_{i=1}^{N} \left(\frac{1}{N} \sum_{m=1}^{p} \xi_{m,j} \xi_{m,i} \right) \xi_{n,i} = \frac{1}{N} \sum_{m=1}^{p} \xi_{m,j} \sum_{i=1}^{N} \xi_{m,i} \xi_{n,i}$$

- Taking out $\xi_{n,j}$ from the sum yields:

$$v_j = \underbrace{\xi_{n,j}}_{\text{signal}} + \underbrace{\frac{1}{N} \sum_{m=1, m\neq n}^{p} \xi_{m,j} \sum_{i=1}^{N} \xi_{n,i} \xi_{m,i}}_{\text{crosstalk (or noise)}}$$

- If the second term (crosstalk) is zero then the activation potential v_j of *j*th neuron is stable as the stability condition holds:

$$\xi_{n,j} = \operatorname{sgn}(v_j)$$

Storage Capacity of the Hopfield Network

- Due to the $\operatorname{sgn}(\cdot)$ activation function the output of jth neuron is stable even for some crosstalk
  - a small amount of crosstalk is not able to change the sign of the activation potential v_j
- Elements of each fundamental memory ξ_n are either $+1$ or -1
  - then weights w_{ji} are in $[+1, -1]$ interval
- Therefore if the number of fundamental memories p is sufficiently small ($p << N$) the crosstalk will be negligible

Storage Capacity of the Hopfield Network

- Let the elements of each fundamental memory ξ_n be Bernoulli random variables over the set $\{-1, +1\}$
  - Each element of ξ_n has zero mean and variance $\sigma^2 = 1/N^2$
- The crosstalk is then a sum of $N(p-1)$ IID random variables
  - IID - independent and identically distributed
- The *Central limit theorem* states that a sum of IID random variables with finite variance converges in distribution to a normal (or Gaussian) distribution
- Therefore the crosstalk may be modeled by a normal distribution having a zero mean and variance equal to $N(p-1) \cdot 1/N^2 = (p-1)/N$

Storage Capacity of the Hopfield Network

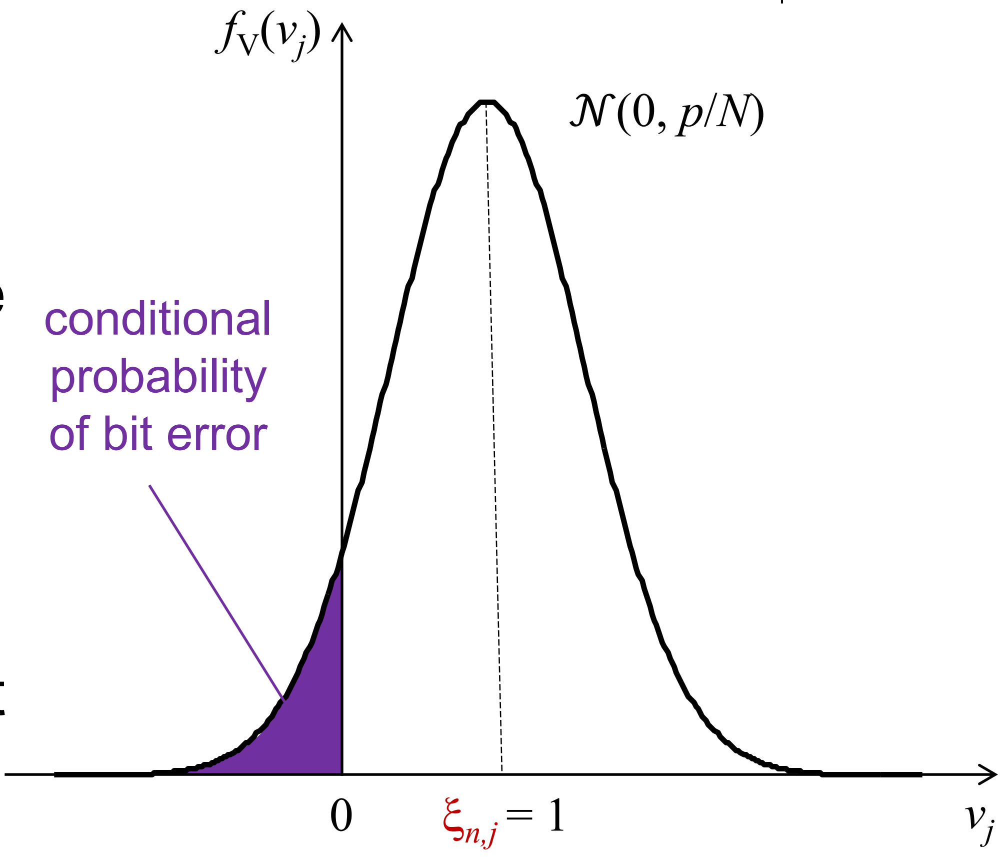
- We may now compute the signal-to-noise ratio or SNR:

$$\rho = \frac{\text{variance of signal}}{\text{variance of noise}} = \frac{1}{(p-1)/N} = \frac{N}{p-1} \approx \frac{N}{p}$$

- Stored components of a fundamental memory will be stable if SNR is high
  - in other words we must have $N >> p$
- The inverse of SNR is called load parameter α
  - lowering the load parameter $\alpha = p/N$ reduces the probability of error

Storage Capacity of the Hopfield Network

- There exists a critical value of the load parameter α_c after which the network performance breaks down
  - $\alpha_c = 0.138$
- Then storage capacity is $p_{\max} = 0.138\,N$ yielding 0.36% of unstable output bits
  - This is network capacity with errors



Storage Capacity of the Hopfield Network

- We may require that the conditional probability of error is smaller than $1/N$
- Then the maximum capacity is $p_{\max} \approx N/(2 \log_e N)$
  - This is storage capacity *almost without errors*
- Finally, we may require that all fundamental memories are retrieved correctly
- Then the maximum capacity is $p_{\max} \approx N/(4 \log_e N)$
- Capacities listed here are derived from the asymptotic behaviour of the conditional probability of bit error

Stochastic Machines

- Stochastic machines introduce random (or stochastic) variations into the network
  - stochastic activation functions
  - stochastic synaptic weights
- Are well suited for solving optimization tasks
  - search problems such as constraint-satisfaction searches
  - learning problems such as data generation
- Often can be described via an energy function which must be minimized
  - we want to find the global minimum instead of a local one
  - stochastic nature of the machine enables us to avoid local minima

Boltzmann Machine

- Are named after the Boltzmann distribution which is used in physics (statistical mechanics)
- Are comprised of many identical neurons that are fully connected to each other
  - each neuron has two states, on or off
  - the state is a probabilistic function of the states of all other neurons and depends on the synaptic weights, i.e. the activation function is stochastic
  - all connections are bidirectional and symmetric
- There exists a choice of an activation function which makes the individual neurons act so to minimize the global energy

Boltzmann Machine and Hopfield Network

- Boltzmann machine can be thought of as a stochastic Hopfield network
  - synaptic weights are deterministic
  - activation functions are stochastic
- However, the purpose of the networks is different
  - Hopfield network stores each pattern in a local minima of its energy function, hence the network must find the local minima
  - Boltzmann machine is intended to find the global minimum of its energy function

Boltzmann Machine: Stochastic Activation Function

- The stochastic activation function of neurons in Boltzmann machine is inspired by the Metropolis algorithm
  - It is closely connected with the simulated annealing technique
- The stochastic activation function of *i*th neuron is

$$s_{i,\text{next}} = \begin{cases} 1, & \text{with probability } p = \dfrac{1}{1+e^{-\Delta E_i/T}} \\ -1, & \text{with probability } p = \dfrac{e^{-\Delta E_i/T}}{1+e^{-\Delta E_i/T}} \end{cases}$$

  where

$$\Delta E = -2 s_{i,\text{previous}} \sum_{j=1}^{N} w_{ij} s_{j,\text{previous}}$$

  is the energy change resulting from *i*th neuron flip
- T is the temperature

Boltzmann Machine: Stochastic Activation Function

- As in the Hopfield network the neuron to be updated is selected randomly (asynchronous update)
- The activation function has the property that the neuron is set to locally non-optimal state with some probability
  - In other words *we sometimes increase the energy of the network*
  - The chance of picking non-optimal states decreases with temperature
  - At temperature $T = 0$ the activation function becomes deterministic and Boltzmann machine turns into a Hopfield network
- When the temperature T is fixed the whole network eventually reaches thermal equilibrium
- The relative probability of two energy states A and B of the network in thermal equilibrium then follows the Boltzmann distribution

$$\frac{P(A)}{P(B)} = e^{-(E_A - E_B)/T}$$

Boltzmann Machine: Annealing Schedule

- At low temperatures there is a strong bias in favor of states with low energy, but the time required to reach the equilibrium may be long
- At high temperatures equilibrium is reached faster, but the bias is not in favor of states with low energy
- Start at high-temperature and then gradually reduce it
  - At high temperatures we perform a coarse search over the space of global states
  - Lowering the temperature finds a better minima within the coarse-scale minima selected at previous higher temperature
- The rate of lowering the temperature is called annealing schedule

Boltzmann Machine: The Issue of Convergence

- The Hopfield network is deterministic and always converges into the closest local minimum
- The Boltzmann Machine is stochastic and converges to global minimum in expectation
  - There is no guarantee that the found minimum is indeed the global one
  - We can only estimate the probability that it is
  - That probability is higher for slower annealing schedules and longer simulation times
- In practice the machine is usually simplified by imposing connectivity constraints
  - Restricted Boltzmann machine (*Smolensky 1986*)

Neural Networks: Boltzmann Machine

SIMULATED ANNEALING

Simulated Annealing

- Proposed by Kirkpatrick et al. in 1983
- Simulated annealing is a probabilistic algorithm for approximating the global optimum of some function
  - It is considered a *meta-heuristic*
  - Has strong foundations in *thermodynamics* and *statistical mechanics*
- Simulated annealing searches for the global extrema of a *bounded objective function* over some *configuration space*
  - It can be used to solve many combinatorial optimization problems

Simulated Annealing

- Both name and notion come from annealing in metallurgy
  - a metal is heated and is then cooled in a controlled manner to achieve desirable properties
  - slow cooling promotes diffusion making the metal progress toward its thermal equilibrium state
  - rapid cooling (quenching) prevents phase transformations and leaves the metal in a metastable state
  - therefore, if done properly annealing results in the metal reaching its minimum energy state

Simulated Annealing: Metropolis Algorithm

- Metroplis et al. in 1953 described how to simulate a collection of molecules in thermal equilibrium at a given fixed temperature T $[K]$
- In simulation a randomly selected molecule is randomly displaced after which the change in energy ΔE [J] of the ensemble is computed
- The key result is Metropolis Acceptance Criterion which defines the probability $\Pr$ to accept the change

$$\Pr\{\text{Accept } \Delta E\} = \begin{cases} 1, & \Delta E \le 0 \\ e^{-\Delta E / kT}, & \Delta E > 0 \end{cases}$$

- In other words
  - If the change lowers the total energy of the system then it is accepted
  - If the change increases the total energy of the system then it is accepted with the probability $\exp(-\Delta E/kT)$, where k is the Boltzmann constant

Simulated Annealing: Metropolis Algorithm

- If the simulation is run for a sufficient number of changes (steps) then the final configuration of molecules is close to the thermal equilibrium or the steady state of a system of molecules
  - It is a global minimum at fixed temperature $T\,[K]$
- A formal proof of the convergence models the simulation process as a homogeneous *Markov chain* and shows the chain’s steady state is the thermal equilibrium
  - In theory convergence happens when the number of steps reaches infinity

Simulated Annealing: Metropolis Algorithm

- Three assumptions for the Metropolis algorithm to produce the thermal equilibrium are:

1) Reversibility (symmetry, detailed balance)
   - the probability of selecting the next state is the same as the probability of returning from that next state to the current state

2) Ergodicity
   - random displacement of molecules is such that molecules may reach any position in configuration space

3) Convergence to canonical distribution
   - Probabilities in the acceptance criterion are such that on average the ensemble tends to a Boltzmann (or Gibbs) distribution, i.e.

$$F(\text{final state}) \propto \exp\left(-\frac{E_{\min}}{kT}\right)$$

Simulated Annealing: Metropolis Algorithm

- The Metropolis algorithm therefore produces a thermal equilibrium or steady state solution for some fixed temperature
- The simulated annealing process runs the Metropolis algorithm iteratively (*Kirkpatrick 1983*):
  - We first melt the system being optimized at high effective temperature
  - At high temperatures all energy states are equally probable
  - Then we lower the temperature in slow stages until the system freezes and no further changes occur
  - At each temperature the Metropolis simulation must proceed long enough for the system to reach steady state at that temperature

Simulated Annealing: Formal Problem Description

- Let Ω be the configuration space
- Let $f:\Omega \to \mathbb{R}$ be an objective (or energy) function
  - i.e. the quantity $f(\omega)$ is the cost of some solution $\omega \in \Omega$
- Let $N(\omega)$ be the neighborhood function for $\omega \in \Omega$
  - i.e. $N(\omega)$ are neighboring solutions that can be reached in a single iteration of a local search algorithm from the solution $\omega \in \Omega$
- Let T_n be *pseudo-temperature* of the system at iteration n
  - i.e. the Boltzmann constant is absorbed into T_n
- We want to find $\omega^* \in \Omega$ such that

$$f(\omega^*) \leq f(\omega)$$

for all $\omega \in \Omega$

Simulated Annealing: Inner Loop at Fixed Temperature

- Start with an initial solution $\omega_{\text{steady}} \in \Omega$ from the higher temperature T_{n-1}
  - If ω_{steady} is not available then randomly select one
- Then randomly generate a neighboring solution $\omega' \in N(\omega)$
- Accept the candidate solution ω' if it satisfies the Metropolis acceptance criterion:

$$\Pr\{\text{Accept}\,\omega'\} = \begin{cases} 1, & \text{if } f(\omega') - f(\omega) \le 0 \\ e^{-\left(f(\omega') - f(\omega)\right)/T_n}, & \text{if } f(\omega') - f(\omega) > 0 \end{cases}$$

- Repeat this random selection of candidates until a steady state ω_{steady} at temperature T_n is reached
  - In practice we simply repeat the procedure M_n times

Simulated Annealing: Outer Loop and Annealing Schedule

- The outer loop simply selects the next temperature T_{n+1} and repeats the inner loop
  - T_n must be a decreasing sequence in n whose limit is zero

- An annealing schedule defines
  - How many steps M_n are required for the Metropolis algorithm to run at temperature T_n
  - What is the sequence of temperatures T_n

The Algorithm in Pseudo-Code

```
Select an initial solution $\omega \in \Omega$
Select a temperature cooling schedule $T_n$
Select a repetition schedule $M_n$

Set outer loop repetition counter $n = 0$
Repeat
        Set inner loop repetition counter $m = 0$
        Repeat
                Generate a random solution $\omega' \in N(\omega)$
                Calculate $\Delta = f(\omega') - f(\omega)$
                If $\Delta \leq 0$ then $\omega \leftarrow \omega'$
                If $\Delta > 0$ then
                        Pick a random $\xi$ from $U[0,1]$ distribution
                        If $\xi < \exp(-\Delta / T_n)$ then $\omega \leftarrow \omega'$
                $m \leftarrow m + 1$
        Until $m = M_n$
        $n \leftarrow n + 1$
Until the stopping criterion is met
```

Simulated Annealing: Convergence

- Convergence to the global minimum is proven using the theory of Markov chains
  - In essence all proofs show that *global minimum* coincides with the *steady state* of a Markov chain which models the simulated annealing process
- Such formal analysis shows there exists limitations on the annealing process, e.g. on temperature cooling schedule (*Mitra, 1986*)

$$T_n \geq \frac{\beta}{\log(n)}$$

where β is a problem-dependent constant

Simulated Annealing: Discussion

- A key property of simulated annealing is that there is a non-zero probability of locally selecting a worse solution
  - That makes it different from simple hill-climbing and gradient descent or ascent algorithms which in each iteration always select a better solution
- This key property also makes the algorithm "dumb" (Fleischer 1995):
  - *"…it is blind to the global optimum and can find it only to leave it in the next iteration searching the entire configuration space in total ignorance of the quality of the solution it left…"*

Simulated Annealing: Practical Considerations

- In practice we want both the convergence property to hold and the running time to be short
  - These are opposing requirements
- Three key issues which must be considered for a practical application of simulated annealing are
  1. An appropriate cooling schedule
  2. A suitable objective function
  3. A well-defined neighborhood structure
- Unfortunatley, answers are mostly problem specific

Travelling Salesman Problem

- The travelling salesman problem is:
  - *Given a list of cities and the cost of travel between each pair of them find the cheapest route that visits each city once and returns to your starting city*
- The problem is NP-hard
- For $N \geq 3$ cities there are exactly $(N - 1)!/2$ possible undirected tours
  - Hence the complexity of brute force approach is $O(n!)$
  - Current exact solutions algorithms have complexity $O(n^2 2^n)$
- *Simulated annealing* can find an approximate solution
  - In many applications a near-optimal solution is good enough

Neural Networks: Recurrent Networks

SELECTED PROBLEMS

Problems

Problem 14.4 from (Haykin, 1998) – Hopfield network

Consider a Hopfield network made up of five neurons which is required to store the following three fundamental memories:

$\xi_1 = [1, 1, 1, 1, 1]^T$, $\xi_2 = [1, -1, -1, 1, -1]^T$ and $\xi_3 = [-1, 1, -1, 1, 1]^T$.

The bias applied to each neuron is zero.

(a) Compute the 5×5 synaptic weight matrix of the network.

(b) Using the asynchronous updating rule demonstrate that all three fundamental memories ξ_1, ξ_2 and ξ_3 satisfy the stability condition.

(c) Investigate the retrieval performance of the network when it is presented with a noisy version of ξ_1 in which the second element is reveresed in polarity, i.e. probe is $x_{\text{probe}} = [1, -1, 1, 1, 1]^T$.

Problems

Problem 14.7 from (Haykin, 1998) – Hopfield network

Consider a Hopfield network made up of two neurons whose synaptic weight matrix is

$$\mathbf{W} = \begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

The bias applied to each neuron is zero.

The four possible states of the network are
$\mathbf{s}_1 = [1, 1]^T$, $\mathbf{s}_2 = [-1, 1]^T$, $\mathbf{s}_3 = [-1, -1]^T$ and $\mathbf{s}_4 = [1, -1]^T$.

(a) Using the stability condition show that states $\mathbf{s}_2$ and $\mathbf{s}_4$ are stable.

(b) Using the energy function show that states $\mathbf{s}_2$ and $\mathbf{s}_4$ are stable.

(c) Show that states $\mathbf{s}_1$ and $\mathbf{s}_3$ exhibit a limit cycle. What is the length of the limit cycle?

Problems

Boltzmann machine

Consider a Boltzmann machine that operates at temperature $T = 1$.

Let the energy of two states labeled A and B be $E_{\mathrm{A}} = -25$ and $E_{\mathrm{B}} = -22$.

We also know that the system has chosen one of these two configurations among all possible configurations, but we do not know which one.

(1) What is $P(\mathrm{A})/P(\mathrm{B})$ (this ratio is also called *Boltzmann factor*)?

(2) Compute the conditional probability $P(\mathrm{A}|\mathrm{A} \text{ or } \mathrm{B})$, i.e. compute the probability of the machine being in state A if we know that machine is in one of A or B states.

(3) Express the conditional probability $P(\mathrm{A}|\mathrm{A} \text{ or } \mathrm{B})$ in terms of energies and temperature.

References

1. Haykin, Simon. *"Neural Networks: A Comprehensive Foundation" (2nd Edition)*. Prentice Hall, 1998.
2. John A., Anders S. Krogh, and Richard G. Palmer. *"Introduction to the theory of neural computation"*. Westview Press. June 1991.
3. Hopfield, John J. *"Neurons with graded response have collective computational properties like those of two-state neurons."* Proceedings of the National Academy of Sciences 81.10 (1984): 3088-3092.
4. Hinton, Geoffrey E., Terrence J. Sejnowski, and David H. Ackley. *"Boltzmann machines: Constraint satisfaction networks that learn" (Technical Report CMU-C5-84-119)*. Carnegie-Mellon University. 1984.
5. Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E. "*Equation of state calculations by fast computing machines*". The journal of chemical physics, 21(6), 1087-1092. (1953)
6. Kirkpatrick, Scott, C. Daniel Gelatt, and Mario P. Vecchi. "*Optimization by simulated annealing*." Science 220.4598 (1983): 671-680.
7. Henderson, Darrall, Sheldon H. Jacobson, and Alan W. Johnson. "*The theory and practice of simulated annealing*." Handbook of metaheuristics. Springer US, 2003. 287-319.