Boltzmann Machines

Learning by Correlation

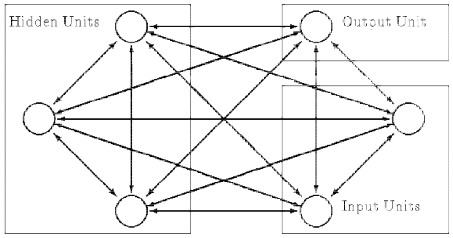
Boltzmann Machine

- Proposed by Ackley, Hinton, and Sejnowski, 1985
- Extends Hopfield model with learning
- Based on probabilitstic operation during training, correlations.
- Deterministic operation once weights determined.

Boltzmann Machine Structure

- The Boltzmann Machine is a Hopfield network, in which
- the neurons are divided into two subsets:
  - **visible**, which are further divided into:
    - input
    - output
  - **hidden**

Boltzmann Machine Structure



Boltzmann Machine Operation

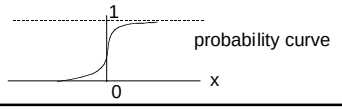
- There are two modes of operation:
  - clamped mode
  - free mode
- In **clamped** mode, the input and output of visible neurons are held fixed, while the hidden neurons are allowed to vary.
- In **free** mode, **only the inputs are held fixed** and other neurons are allowed to vary.

Stochastic Operation (used in Training)

- All neurons have output in {+1, -1}.
- The activation function determines not the exact next input, but rather the **probability** of the neuron's output being set to 1:
  - f(net) = probability that output is set to 1
- where $f(x) = 1/(1 + \exp(-2\beta x))$
  - where β is a parameter to be determined.

Stochastic Operation

- $f(x) = 1/(1 + \exp(-2\beta x))$
- Obviously this is a sigmoid:
  - With $\beta = 0$, the probability of setting to 1 is 0.5, i.e. total randomness.
  - As β increases, the probability of setting to 1 is > 0.5 if $x > 0$, and < 0.5 if $x < 0$.



Controlling β

- In order to stabilize the network, β is gradually increased from 0 over time.
- For a given input, initially changes in neuron state will be random, but eventually it will be determined *only* by the activation value (weighted sum of inputs) without randomness.
- At this point, the network will then stabilize.

Controlling β

- We can think of the increase in β as "cooling" the network.
- Thus, we can govern beta as $\beta = 1/T$ where T is the "temperature", which decreases with time (according to a *schedule*).
- The overall process is known as "**simulated annealing**".

Annealing Schedule

- The annealing schedule determines the temperature T as a function of the step of the algorithm.
- Example:
  $T = T_0/(1 + \log k)$
  where k is the step number and T_0 is an initial temperature.

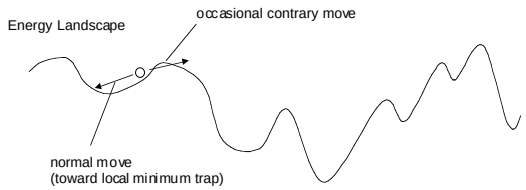
History of Simulated Annealing

- SA was first proposed in 1983 as a method for optimizing wire-routing on VLSI chips (an NP-hard problem)
- by Kirkpatrick, Gelatt, and Vecchi.
- This was a widely-celebrated result.
- SA is now used as a way to avoid local minima in a number of computational problems.

Role of Annealing in Stabilization

- Generally, move in direction of decreasing energy.
- **Occasionally, accept a move that increases energy.**
- This will be done with high probability at first, but lower probability as annealing progresses.

Role of Annealing in Stabilization

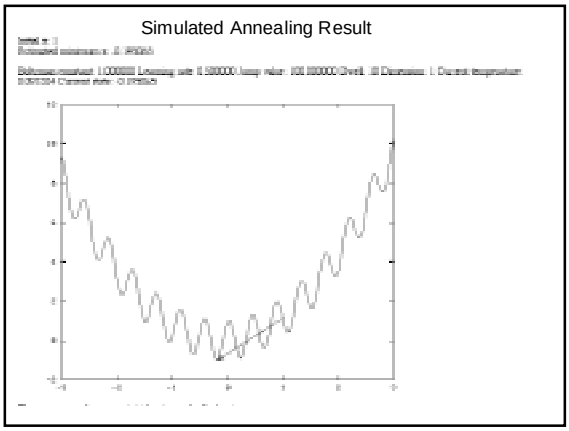


The probability of making a contrary move is inversely proportional to the energy increase and to the temperature (higher probability earlier in the annealing schedule).

Stand-alone demo of Simulated Annealing

http://www.taygeta.com/annealing/demo1.html

Simulated Annealing Result



Energy-Based Simulation

- As we know from Hopfield theory, making a single transition according to the activation function will decrease the energy.
- So we can simply decide to "flip" a neuron based on whether the flip lowers the energy (defined as $-\Sigma \Sigma w_{ij} y_i y_j$).

Energy-Based Simulation

- To include the annealing temperature:
  - If a flip lowers the energy, do it.
  - If a flip raises the energy by ΔE, do it with probability $\exp(-\Delta E/T)$.
    - In other words, generate a random number $\rho \in [0, 1]$ and if $\rho < \exp(-\Delta E/T)$, flip the neuron, otherwise leave it as is.

Energy-Based Simulation

- This technique was originally used in the famous Metropolis, Rosenbluth, Teller equation of state calculations in statistical mechanics (Ising or "spin-glass" model).

Energy-Based Simulation

- In order to compute ΔE, it is not necessary to fully compute the energy before and after. Instead could just use $\Delta E = \Sigma\, w_{ij}\, y_j$ where i is the neuron being flipped.

Boltzmann Distribution

- The name of the machine derives from the fact that, at steady state, if s and t are two states with energies E_s and E_t respectively, then the probabilities of being in those states P[s] vs. P[t] satisfy

  $$P[s]/P[t] = \exp(E_t - E_s)/T$$

  where T is the temperature. This is known as the "Boltzmann distribution" or "Boltzmann-Gibbs distribution".

Multiple Simulations Per Sample

- Suppose we set the input and output neurons according to a specific **sample**.
- We then anneal the network.
  The final state reached is not necessarily unique, due to the probabilistic moves made along the way.
- We can observe, over **several** such simulations, which neurons' outputs are **correlated** at the end, represented as a **correlation** $\rho_{ij} = E[y_i\, y_j]$, the average value of the product of the outputs of neurons i and j.

Learning Rule

- Let ρ_{ij}^{+} be the correlation value when the network is run in **clamped** mode, and ρ_{ij}^{-} be the correlation value when the network is run in **free** mode.
- The Boltzmann learning rule is

  $$\Delta w_{ij} = \eta(\rho_{ij}^{+} - \rho_{ij}^{-})$$

  where η is the learning rate.
- In other words, whether weights are changed depends on the difference between the correlations in clamped vs. free mode.

Summary of Boltzmann Learning

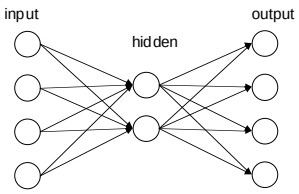
- Repeat until weights stabilize
  - For each sample:
    - Run the sample multiple times in both clamped and free mode.
      - In each run, annealing takes place.
      - Accumulate statistics from each run.
    - Compute the weight update for the sample based on the statistics and the learning rule.

Conclusion of Learning

- At the conclusion of a number of weight-adjustment cycles, we have a Hopfield net that can be operated deterministically.

Boltzman Example

- Ackley, Hinton, and Sejnowski, 1985 present the following example:
- 4 line one-hot encoder-decoder, 2 hidden units



Boltzman Example

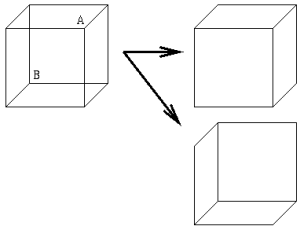
- To prevent weights from growing too large, used a "noisy" **clamping** technique: each *on* bit of a clamped vector is set to *off* with prob. 0.15 and each *off* bit set to *on* with prob. 0.05.
- Network was **unclamped** and allowed to reach equilibrium. Statistics were gathered for the same number of annealings as in the clamped case.
- Annealing schedule: (time units @ temperature) 2@20, 2@15, 2@12, 4@10.
- 1 time unit = interval giving each neuron a chance to flip.

Additional Examples

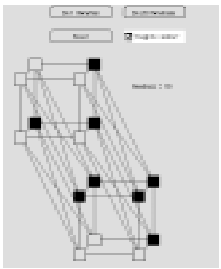
- 4-3-4 encoder/decoder converged quickly
- 8-3-8: more difficult
- 40-10-40: converged in 98.6% of runs.

Necker Cube Demo

(http://www2.psy.uq.edu.au/~mav/java/Necker.html)



Necker Cube Demo



Neurons representing whether vertex is shown in front or back

Blue weights are positive, red are negative.

Constraints Represented

- Each vertex can have only one interpretation. Therefore there are negative weights connecting units representing different interpretations of the same vertex.
- The same interpretation cannot be given to more than one vertex - so units representing the same interpretation are connected with negative weights.
- Units from the same interpretation should be on together, so locally consistent units are connected with positive weights.

Speedup Possibility

- Training of a Boltzmann machine is extremely slow.
- A possible speedup is to use the "mean-field" approximation to get the correlation values.
- This approach is due to Peterson and Anderson, 1987.

Mean-Field Theory

- If f is a function of two variables, then the expectation
  E[f(x, y)]
  can be *approximated* by
  f(E[x], E[y])
- This idea can be applied to the weight change rule of the Boltzmann machine, which entails computing

$$\rho_{ij} = E[y_i y_j] \approx E[y_i] E[y_j]$$

Mean-Field Theory

- For the Boltzmann distribution, the probability that node i takes value 1 at temperature T can be shown to be:

$$p_1 = 1/(1+\exp(-\Sigma w_{ij} E[y_j]/T))$$

- So the *expected* output of node i is

$$E[y_i] = 1*p_1 + (-1)*(1- p_1)$$

$$= \tanh(\Sigma w_{ij} E[y_j] / 2T)$$

Mean-Field Theory

- We now have n non-linear equations in n unknowns $E[y_j]$ which can be solved ***deterministically*** by using successive approximations (without simulation!, but we still have to anneal).
- We can then use these approximations to update the weights:

$$\Delta w_{ij} = \eta(E^+[y_i] E^+[y_j] - E^-[y_i] E^-[y_j])$$

where + and - designate clamped vs. free as before.

Cauchy Machine (Szu, 1986)

- Same topology as Boltzmann machine
- Learns arbitrary spatial patterns by Hebbian encoding and fast simulated annealing, claimed to find min. energy with probability 1.
- Probability of neuron being 1 is $p_1 = T/(T+(\Delta E)^2)$ vs. $p_1 = 1/(1+\exp(-\Delta E/T))$ for Boltzmann.
- Annealing schedule is $T = T_0/(1 + k)$ vs. $T = T_0/(1 + \log k)$ for Boltzmann.

Constraint-Satisfaction Problems

- The Necker cube example illustrates how minimum-energy seeking networks can be used to find solutions to constraint-satisfaction problems (problems specified by constraints that can't be violated, rather than as computing a function).
- A number of other such problems have been studied in this context.

Traveling Salesperson Problem

- The problem is: given a set of n nodes ("cities") with a specified minimum cost between each pair of nodes, find a permutation ("tour") of the nodes that minimizes the summed costs between the nodes in the permutation sequence.
- The costs are symmetric, and the general problem does not require that there be any Euclidean relationship among the nodes.

Finding Solutions to the TSP using a Hopfield Net

- Global minimum is not necessarily found, although this might be doable with a Boltzmann style algorithm instead.
- The difficulty is encoding the instance of the TSP as a net:

  minimal cost $\leftrightarrow$ minimal energy

TSP Formulation

- Represent a given problem as a matrix:
  - Cities correspond to rows.
  - Positions on the tour correspond to columns.
  - Example:

	1	2	3	4	5
A	0	0	1	0	0
B	1	0	0	0	0
C	0	0	0	0	1
D	0	1	0	0	0
E	0	0	0	1	0

  means B occurs first on the tour, D occurs second, A third, E fourth, C fifth.

TSP Formulation

- Assume {0, 1} values rather than {-1, 1}.
- The neurons correspond to entries in the matrix (n^2 neurons for n cities).
- Neurons in a row have inhibitory connections from other neurons in same row:
  - If one neuron is on, then others tend to be off, especially in minimum energy state.
- Similarly for neurons in the same column

TSP Formulation

- Need to represent costs between cities as neural weights:
  - Want to inhibit selection of adjacent cities in proportion to the cost between those cities.
  - Let X and Y be rows and i and j be columns.
  - Let w_{XYij}, the weight between neurons Xi and Yj, in different columns be $-Dc_{XY}(\delta_{j,i+1} + \delta_{j,i-1})$ where
    - -D is a constant to be chosen
    - c_{XY} is the direct cost from city X to city Y
    - $\delta_{j,i}$ is the Kronecker delta (1 if i = j, 0 otherwise)

TSP Formulation

- Finally, need to favor tours that include *all* n cities, as opposed to just a subset of them. This is done by adding a sufficiently-large global inhibition to each neuron.
- In total, we choose the weight from Xi to Yj as

  $w_{XiYj} = -A\delta_{XY}(1- \delta_{i\,j}) -B\,\delta_{i\,j}(1- \delta_{XY})-C-Dc_{XY}(\delta_{j,i+1} + \delta_{j,i-1})$

  for appropriate constants A, B, C, D.

TSP Formulation

- The corresponding energy is the computed to to have the form

$$A\Sigma_X\Sigma_i\Sigma_{j\neq i}\, y_{Xi}y_{Xj}$$

$$+ B\,\Sigma_X\Sigma_i\Sigma_{Y\neq X}\, y_{Xi}y_{Yi}$$

$$+ C(\Sigma_X\Sigma_i\, y_{Xi} - n)^2$$

$$+ D\Sigma_X\Sigma_{Y\neq X}\Sigma_i\, c_{XY}y_{Xi}\,(y_{Y,i+1} + y_{Y,i-1})^2$$

Other Hopfield Machines

- BAM (Bi-directional Associative Memory) (Bart Kosko, USC)
- Stores input-output patterns
- Can retrieve either direction
  - Given input, find output
  - Given output, find input
- Hopfield net divided into two tiers with behavior activating one tier then the other.