Boltzmann Machines

Learning by Correlation
in a Hopfield-like Net

Boltzmann Machine

- Proposed by Ackley, Hinton, and Sejnowski, 1985.
- Extends Hopfield model with learning.
- Based on **probabilistic operation.**
- Learning is by **correlation** (sort of Hebbian in character).

Spin Glasses and the Ising Model

(useful for understanding Boltzmann machine)

Spin Glass

(http://www.tutorgig.com/ed/Spin_glass)

A *'spin glass*' is a disordered material exhibiting high magnetic **frustration** (inability to remain in a single lowest energy state).

The origin of the behavior can be either a disordered structure (such as that of a conventional, chemical glass) or a disordered magnetic doping in an otherwise regular structure.

Spin Glass

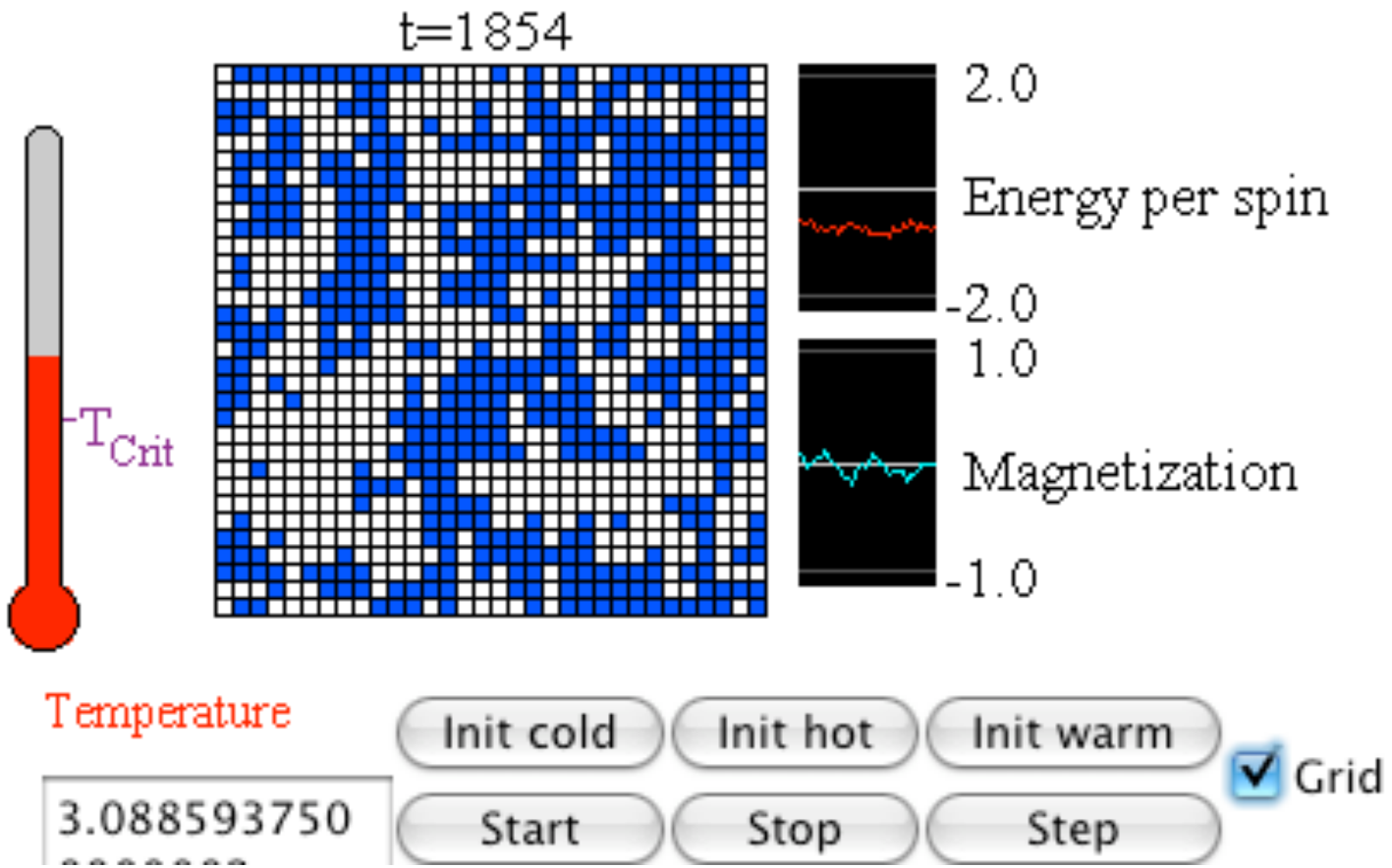
It is the time dependence which distinguishes spin glasses from other magnetic systems. Beginning **above** the spin glass transition temperature, **Tc**, where the spin glass exhibits more typical magnetic behavior, if an external magnetic field is applied and the magnetization is plotted versus temperature, it follows the typical Curie law (in which magnetization is inversely proportional to temperature) until Tc is reached, at which point the magnetization becomes virtually constant (this value is called the field cooled magnetization). This is the onset of the **spin glass phase**.

Ising Model

- The Ising model is a mathematical model that attempts to explain the behavior of spin glasses.
- An array of spin values is assumed, each +1 or -1.
- **An update consists of summing the neighboring spins and setting this spin to some function of that sum.**
- Generally this spin will tend to take on the value of the predominant spin of neighbors.

Ising Model Applet

(http://physics.ucsc.edu/~peter/ising/ising.html)



Ising Demo Information

- The energy is calculated from the formula E= - Sum<i,j> Si Sj where <i,j> symbolizes all pairs of nearest neighbors on the lattice.
- At infinite temperature the energy per spin (E/N, where N = L^2 is the number of spins) is zero. At zero temperature, all the spins are parallel and the energy per spin is -2.
- The critical temperature of the two dimensional Ising model is Tcrit=2/ln(1+sqrt2) = 2.269 . Initially the temperature is set to this value.
- The magnetization is simply the mean of all spins.

Ising Demo Information

- At temperatures well above the critical temperatures, the spin arrangement converges to a nearly random arrangement, independent of the starting state: "Init cold", "Init warm" or "Init hot", and fluctuates quickly. We say that, above the critical temperature, there is a single thermodynamic state and this has zero magnetization. The spin arrangement is truly random at infinite temperature.

- If you start below the critical temperature with "Init cold" (i.e. all the Si=-1) you will see that just a few small cluster of blue (i.e. Si=1) spins appear, and there is a non-zero (negative) magnetization. If we had started the simulation with all the Si 1 (blue) then there would have been a net positive magnetization. We see that, below the critical temperature, there are two thermodynamic states (the "up spin" state with positive magnetization and the "down spin" state negative magnetization) and the system stays in one or the other depending on how the spins are initialized.

Ising Demo Information

- If you start below the critical temperature with "Init hot" or "Init warm" then you see that the system initially cannot make up its mind whether to go into the "up spin" or "down spin" state. Large clusters of each spin form. Eventually, if you let the simulation run for a long time, one of the states will win. Which one wins depends on the random thermal fluctuations. There is equal probability for it to be the "up spin" or "down spin" state.

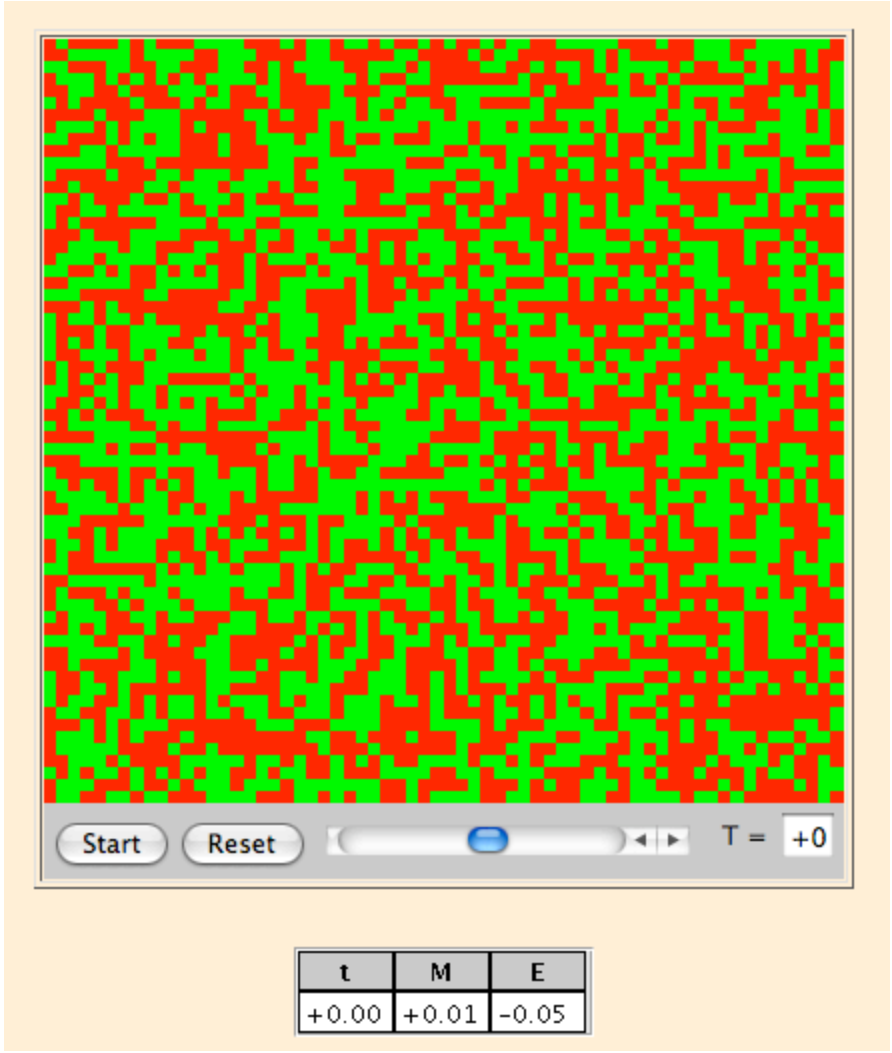
- For temperatures near the transition temperature, there are large clusters of spins with the same orientation, which fluctuate only very slowly. This is because the "correlation length" of an infinitely large system diverges at the critical point.

Ising Model Applet 2

(http://www.phy.syr.edu/courses/ijmp_c/Ising.html)

Ising Model Applet 3

(http://webphysics.davidson.edu/applets/ising/default.html)



Boltzmann Machine Structure

- The Boltzmann Machine is like a Hopfield network, in which
- the neurons are divided into two subsets:
  - **visible**, which are further divided into:
    - input
    - output
  - **hidden**
- As with the Hopfield model, the weights are symmetric.

Boltzmann Machine Structure

Note on Structure

- Different structures have appeared in the literature.
- For example, another one is a layered structure, in which there is no direct connection between input and output (called RBM).

Boltzmann Machine Operation

- There are two modes of operation:
  - clamped mode (used only for learning)
  - 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 all other neurons are allowed to vary.

Weights & Energy

$$E = -\sum_{i<j} w_{ij}\, s_i\, s_j + \sum_i \theta_i\, s_i$$

Where:

- w_{ij} is the connection strength between unit j and unit i.
- s_i is the state, $s_i \in \{0, 1\}$, of unit i.
- θ_i is the threshold of unit i.

The connections in a Boltzmann machine have two restrictions:

- $w_{ii} = 0 \qquad \forall i$. (No unit has a connection with itself.)
- $w_{ij} = w_{ji} \qquad \forall i, j$. (All connections are symmetric.)

Probabilistic Firing

Thus, the difference in the global energy that results from a single unit i being 0 versus 1, written ΔE_i, is given

$$\Delta E_i = \sum_j w_{ij} s_j - \theta_i$$

A Boltzmann machine is made up of stochastic units. The probability, p_i of the i-th unit being on is given by:

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

where the scalar T is referred to as the temperature of the system.

Controlling T

- We can think of the decrease in T as “cooling” the network.
- The exact values of T can be controlled by a schedule or a function of “time”.
- 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.

Stand-alone demo of Simulated Annealing: Minimizing a function (currently disable, but see similar matlab demo)

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

1-D Function, called **func** to find minimum of (in ANSI Forth):

```
: func ( -- ) ( F: x -- z )          \ lots of local minima
                                     \ f[x] = cos( 14.5 x - 0.3 )
                                     \          + (x + 0.2) * x
        FDUP 14.5E0 F* 0.3E0 F- FCOS
        FSWAP
        FDUP 0.2E0 F+ F*
        F+
;
```

You are encouraged to try your own function!

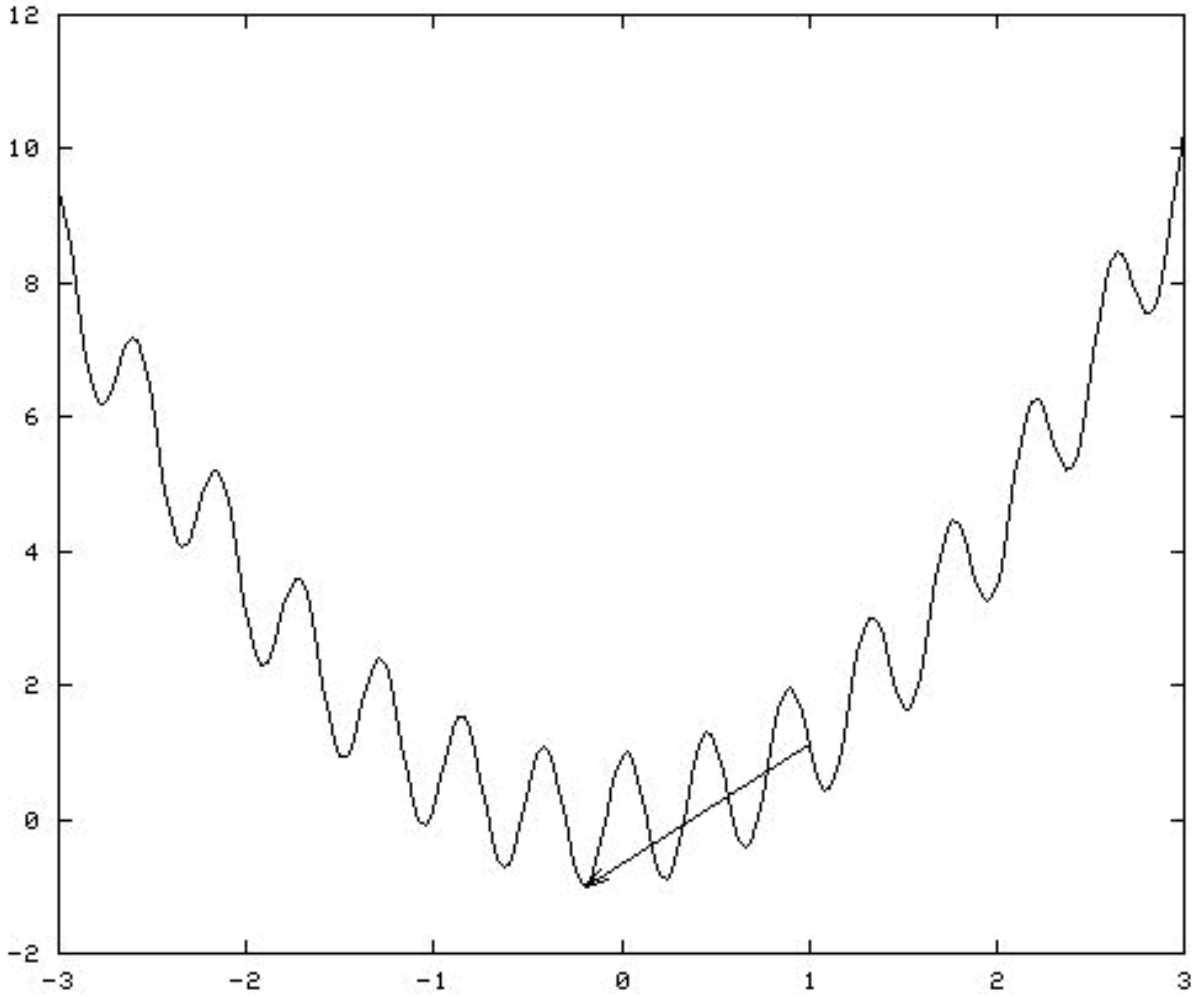
X Range, from: -3.0 to: 3.0

Initial X location: 1.0

Simulated Annealing Result

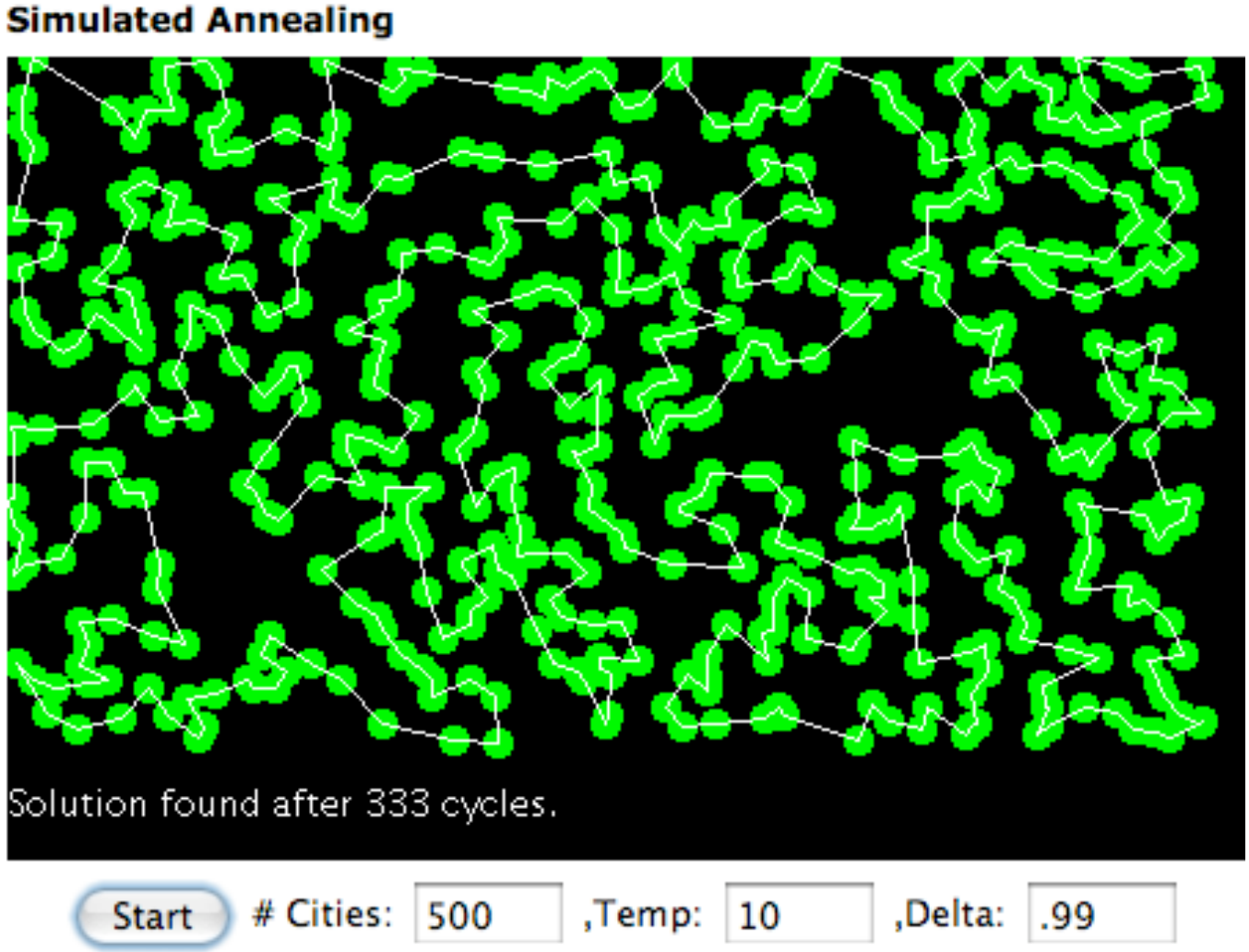
Initial x: 1
Estimated minimum x: -0.195065

Boltzman constant: 1.000000 Learning rate: 0.500000 Jump value: 100.000000 Dwell: 10 Dimension: 1 Current temperature: 0.093204 Current state: -0.195065



TSP via Simulated Annealing

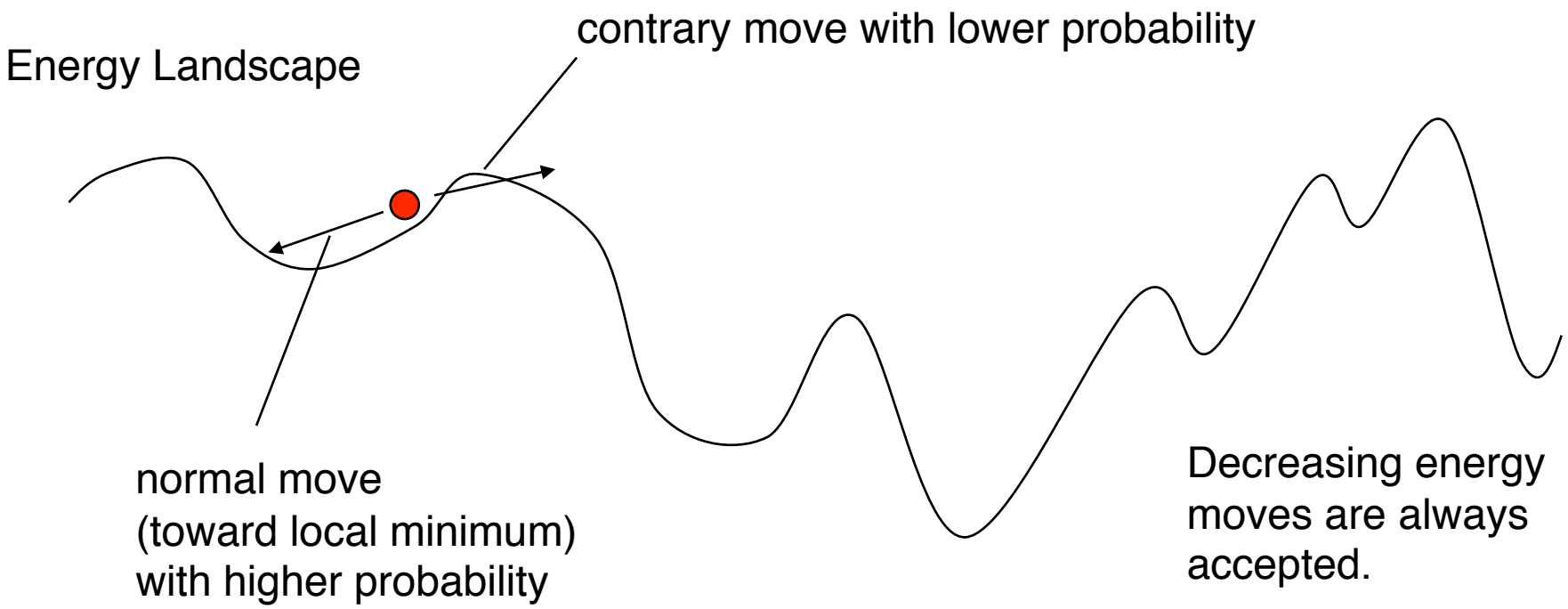
http://www.heatonresearch.com/articles/64/page1.html



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



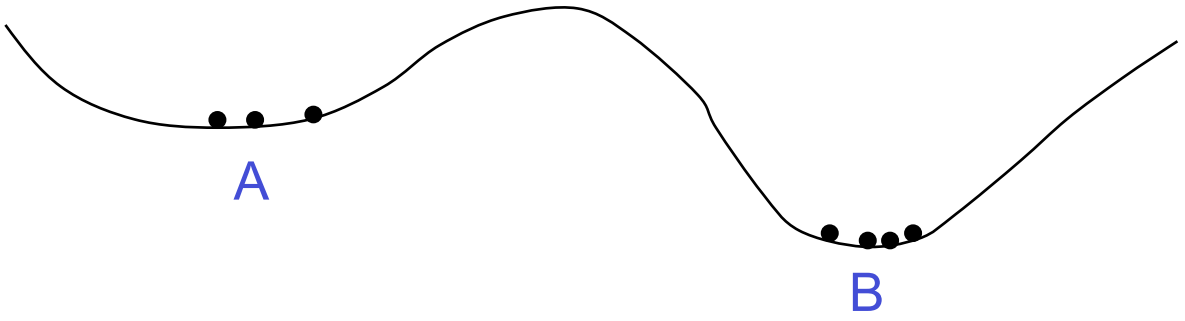
The probability of making a contrary move is inversely proportional to the energy increase and to the temperature (higher probability earlier in the annealing schedule).

How temperature affects transition probabilities (slide from Hinton)

High temperature transition probabilities

$$p(A \rightarrow B) = 0.2$$

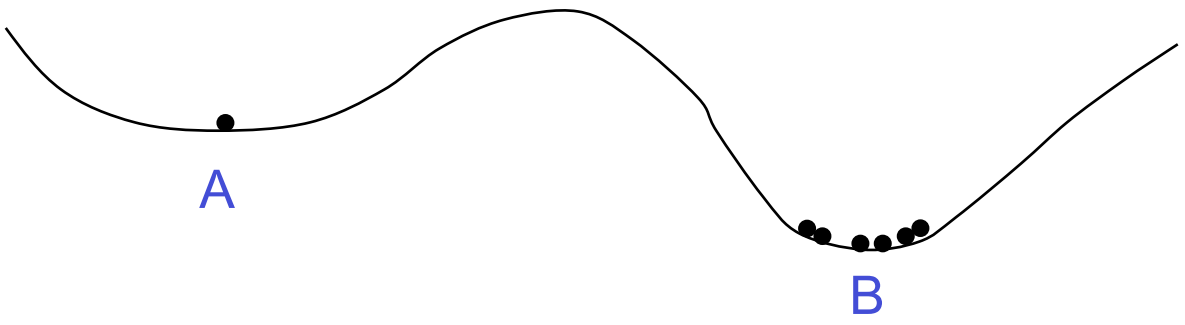
$$p(A \leftarrow B) = 0.1$$



Low temperature transition probabilities

$$p(A \rightarrow B) = 0.001$$

$$p(A \leftarrow B) = 0.000001$$



Annealing Advantage

- At high temperature the ***transition*** probabilities for uphill jumps are much greater.
- At low temperature the ***equilibrium*** probabilities of low energy states are much higher than the equilibrium probabilities of high energy ones.

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 $-\Sigma \Sigma w_{ij} y_i y_j$).
- The decrease in energy for flipping the i^{th} neuron is equivalent to:

$$\Delta E_i = E_i(-1) - E_i(+1) = \Sigma_j w_{ij} y_j$$

Energy-Based Simulation

- To include the annealing temperature:
  - If a flip **lowers** the energy, do it.
  - If a flip **raises** the energy by ΔE, flip the output probabilistically.

Energy-Based Simulation

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

- This is generally called the "**Metropolis algorithm**".

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".

Learning in the Boltzmann Machine

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 are made along the way.
- We can observe, over **several** simulation steps, which neurons' outputs are **correlated** at the ends, represented as a **correlation** $\rho_{ij} = E[y_i\, y_j]$, the expected (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.

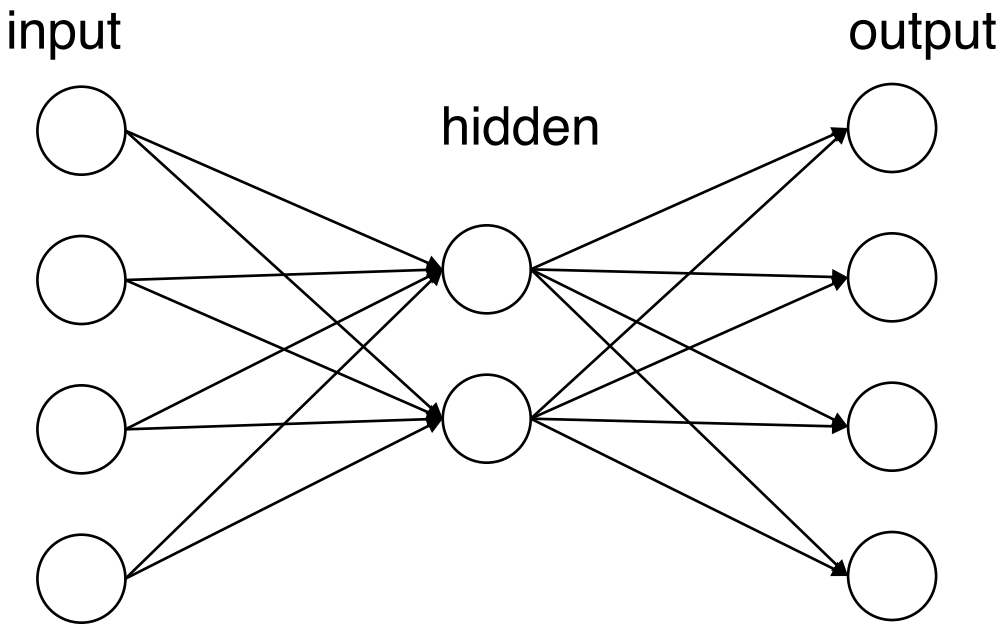
- In other words, whether weights are changed depends on the difference between the correlations in clamped vs. free mode.

Batch Learning Algorithm

- Clamped phase
  - For each data vector in the training set:
    - Clamp the data vector on the visible units.
    - Let the hidden units reach thermal equilibrium at a temperature of 1 (may use annealing to speed this up).
    - Accumulate ρ_{ij}^{+} by sampling $y_i\, y_j$ for all pairs of units.
- Free phase
  - Repeat many times to get good estimates
    For each data vector in the training set:
    - Do not clamp any of the [output?] units.
    - Let the whole network reach thermal equilibrium at a temperature of 1.
    - Accumulate ρ_{ij}^{-} by sampling $y_i\, y_j$ for all pairs of units.
- Weight updates
  - Update each weight by adding $\eta(\rho_{ij}^{+} - \rho_{ij}^{-})$.

Boltzman Example

- Ackley, Hinton, and Sejnowski, 1985 presented 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.

Boltzmann Simulators

- /cs/cs152/boltzmann
  - Simulates only the distribution, not learning.
  - It is analogous to a spin-glass simulation.

- /cs/cs152/boltz
  - Learning, weight-saving, etc.
  - An example, bxor, provides a demo of training a Boltzmann machine to implement xor. Run the shell script bxor.run (may have to run more than once for convergence).

- A new matlab version in progress.

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 Machine

- 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.
  $T = T_0/(1 + \log k)$ (typical) for Boltzmann.