Memory Models

Memory Models

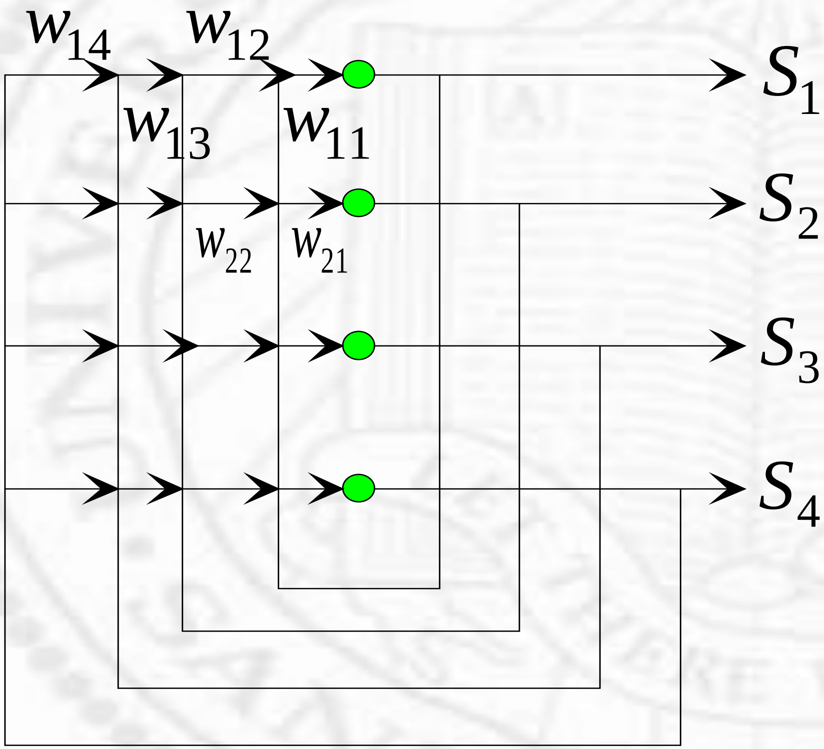
- There exits many other NN models/architectures to perform functions other than pattern recognition
- As an associative memory
  - content addressable
  - partial (noisy) information retrievable
- An optimization tool
  - minimize a cost function

Two Questions for Memory Models

- A learned ANN (fixed parameters)
  - Given some input (with error, missing data, etc.) how does it retrieve stored information?
  - Content-based retrieval
- An unlearned ANN (random parameters)
  - How to impose data and store the data?

Hopfield Net

- A completely connected graph with no hidden unit



$$S_i = \mathrm{sgn}\left(\sum_j w_{ij} S_j - \theta_i\right)$$

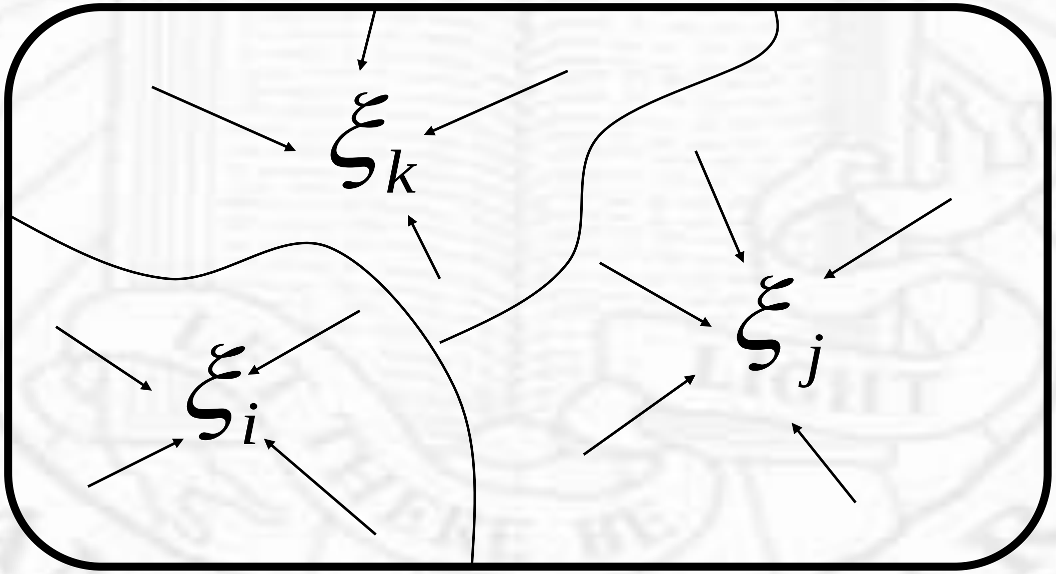
$$w_{ii} = 0 \ \text{(usually)}$$

Mathematical model

- A recurrent network (with feedback connections)
- Binary (1, -1) inputs
- Update can be either synchronous or asynchronous
- Synchronous
  - central clock
  - one-step
  - Not realistic for real NN
- Asynchronous
  - random update sequence
  - settle down “eventually”
- Continuously
  - in analog circuitry

Associate Memory (Learned)

- Pictorially, as an associate memory
  - tolerate certain imprecision



Learning Rule

- As an associate memory: one pattern
  - to force $$\xi_i = \text{sgn}\left(\sum_j w_{ij}\xi_j\right)$$
  - we have $$\boxed{w_{ij} = \frac{1}{N}\xi_i\xi_j}$$
  - Hebbian rule: Neurons that fire together, wire together. Neurons that fire out of sync, fail to link
  - If w_{ij} is positive, neuron j will attract neuron i close. Otherwise, neuron j will push neuron i away
  - Simple learning rule: both strength and weakness of the model

Associate Memory (cont.)

- Ideally, no error

$$S_k = \xi_k$$

$$\Rightarrow h_i = \mathrm{sgn}(\frac{1}{N}\sum_j w_{ij} S_j) = \mathrm{sgn}(\frac{1}{N}\sum_j \xi_i \xi_j \xi_j)$$

$$= \mathrm{sgn}(\frac{1}{N}\sum_j \xi_i) = \xi_i$$

With <50% error

$$h_i = \mathrm{sgn}(\frac{1}{N}\sum_j w_{ij} S_j) = \mathrm{sgn}(\frac{1}{N}\sum_j \xi_i \xi_j (\pm \xi_j))$$

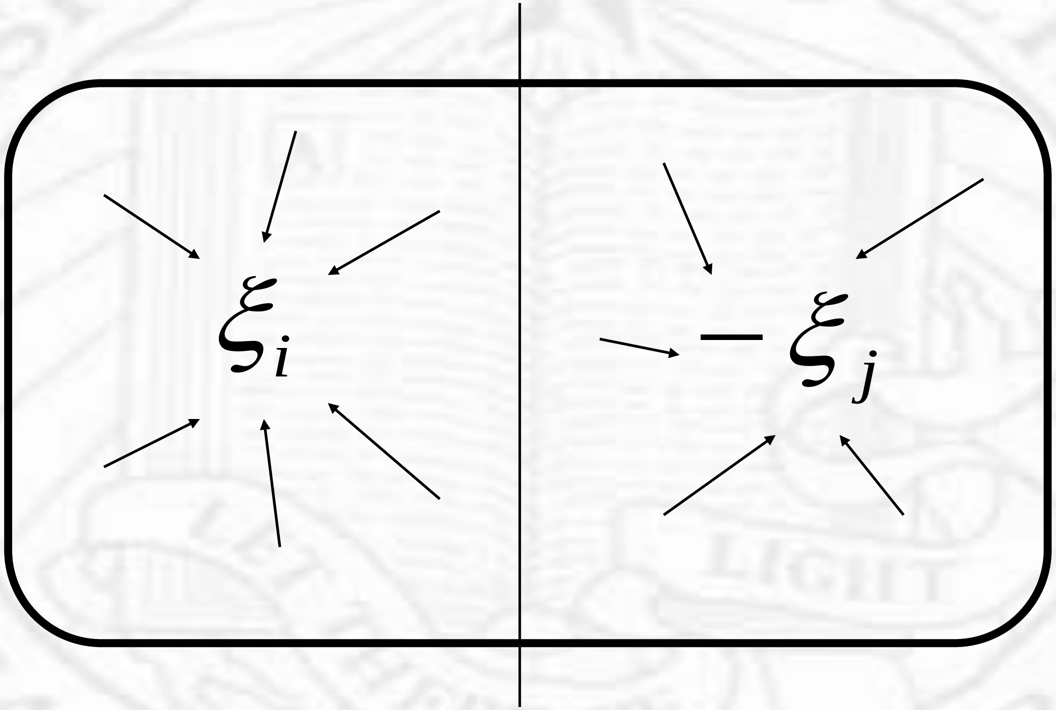
$$= \mathrm{sgn}(\frac{1}{N} \xi_i \alpha) = \xi_i \qquad \alpha > 0$$

Otherwise, end up at $-\xi_i$

two steady states

Associate Memory (cont.)

- Pictorially, as an associate memory with two states



More than one pattern

- Remember all of them (Hebb's rule or prescription)
- Can a stored pattern still be retrieved?

$$\boxed{w_{ij} = \frac{1}{N}\sum_{u=1}^{p} \xi_i^u \xi_j^u}$$

- Yes, if size of the second term is < 1

$$\operatorname{sgn}(h_i^v) = \xi_i^v \quad (\textit{for all } i)$$

$$h_i^v = \sum_j w_{ij}\xi_j^v = \frac{1}{N}\sum_j \sum_u \xi_i^u \xi_j^u \xi_j^v$$

$$= \xi_i^v + \frac{1}{N}\sum_j \sum_{u \neq v} \xi_i^u \xi_j^u \xi_j^v$$

- u: training patterns, v: test pattern

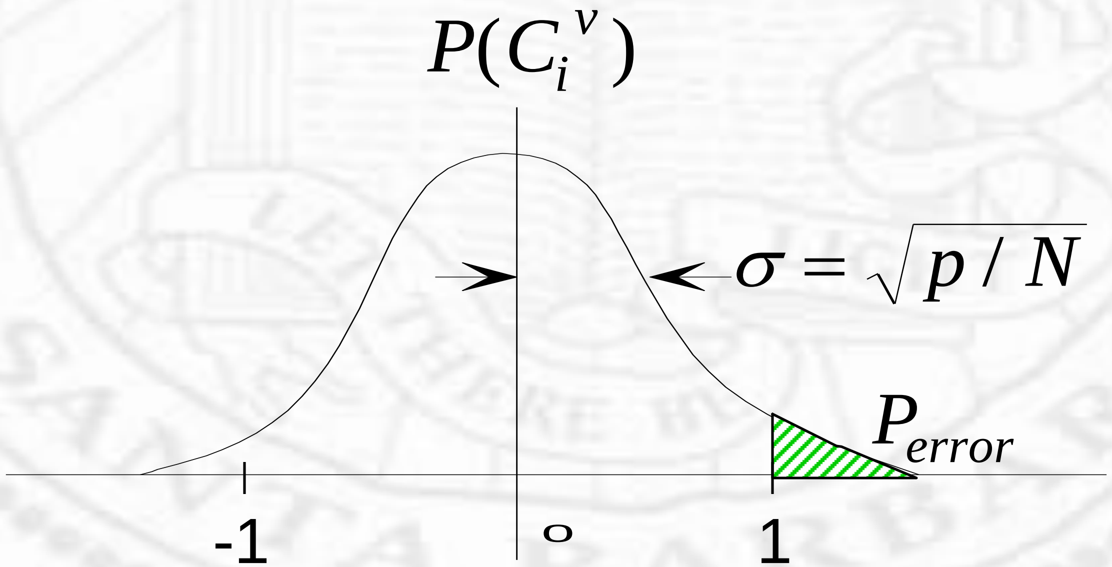
Storage capacity

- the cross-over term must be small

$$C_i^v = -\xi_i^v \frac{1}{N} \sum_j \sum_{u \neq v} \xi_i^u \xi_j^u \xi_j^v$$

- if

$C_i^v > 1$ then output will be incorrect

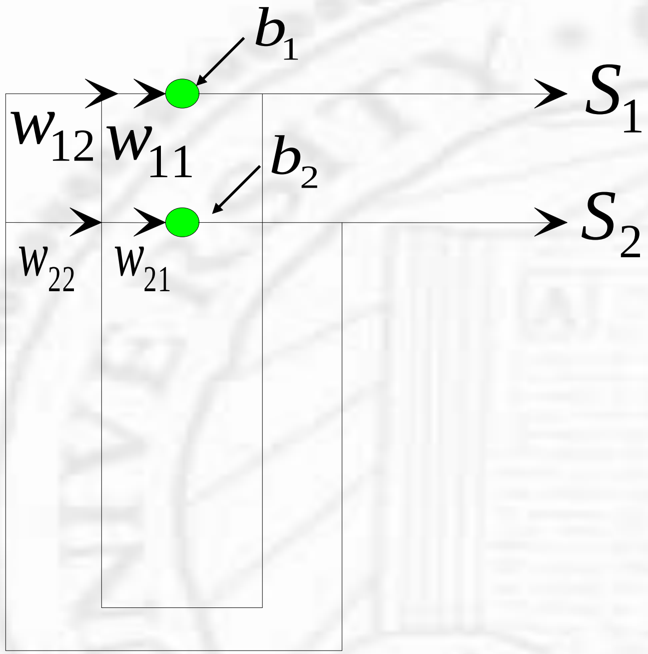


- If p>>1 & N>>1 & N>>p $$C_i^v = -\xi_i^v \frac{1}{N} \sum_j \sum_{u \neq v} \xi_i^u \xi_j^u \xi_j^v$$
  - p: # of patterns
  - N: length of the pattern
- If the p stored patterns are random
  - $$p(\xi_i^u = 1) = \frac{1}{2} \quad p(\xi_i^u = -1) = \frac{1}{2} \ \forall u, i$$
    $$\Rightarrow p(\xi_i^u \xi_j^u = 1) = \frac{1}{2} \quad p(\xi_i^u \xi_j^u = -1) = \frac{1}{2} \ \forall u, i \neq j$$
    $$\Rightarrow p(\xi_i^u \xi_j^u \xi_j^v = 1) = \frac{1}{2} \quad p(\xi_i^u \xi_j^u \xi_j^v = -1) = \frac{1}{2} \ \forall u \neq v, i \neq j$$

C_i^v binomial distribution with zero mean and variance p/N when p>>1 and N>>1 can be approximated by a Gaussian

Error rate dependence

P_{error}	$P_{\max} / N$
0.001	0.105
0.0036	0.138
0.01	0.185
0.05	0.37
0.1	0.61


b_1
b_2
w_{12}
w_{11}
w_{22}
w_{21}
S_1
S_2

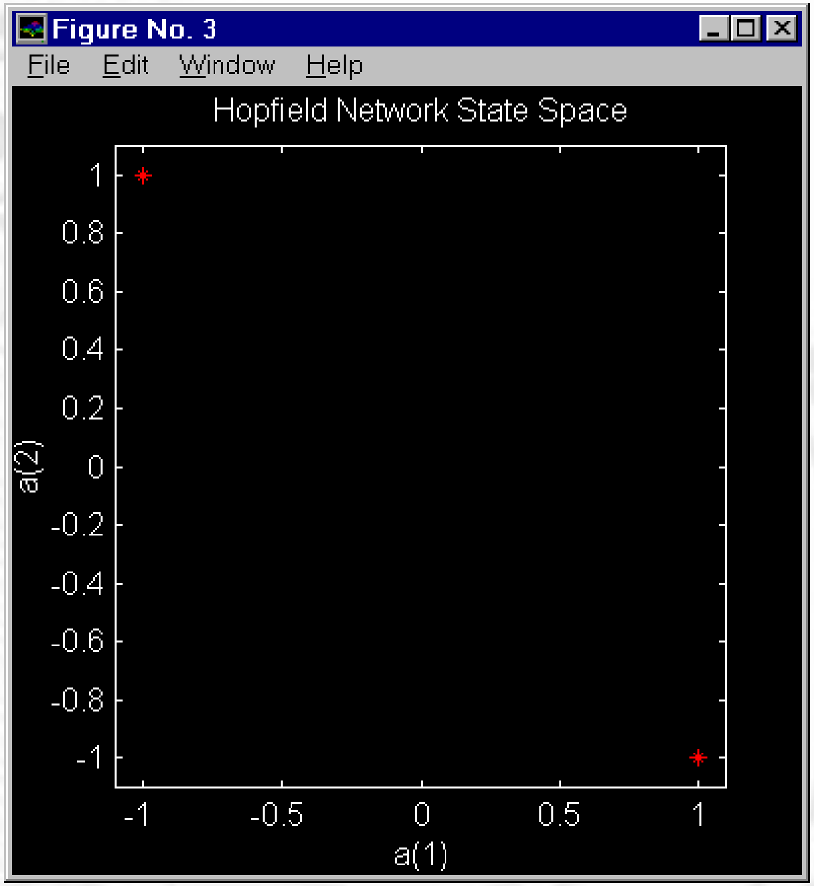


Figure No. 3
File Edit Window Help
Hopfield Network State Space
a(2)
a(1)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

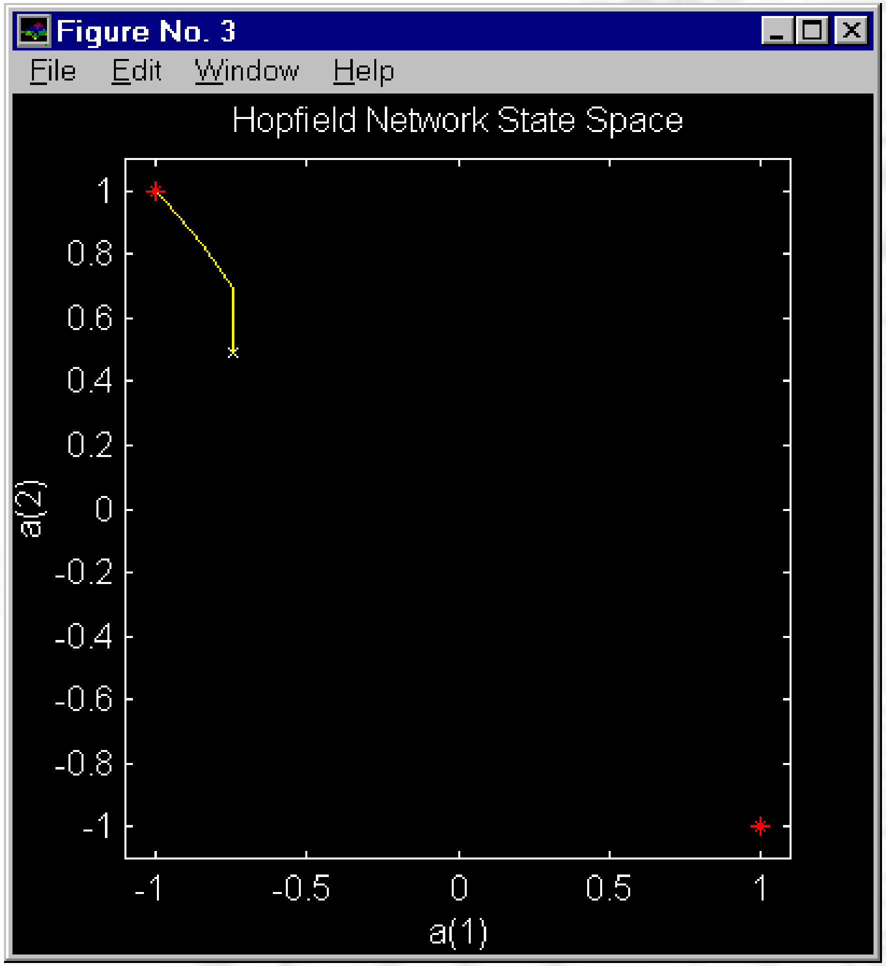


Figure No. 3
File Edit Window Help
Hopfield Network State Space
a(2)
a(1)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

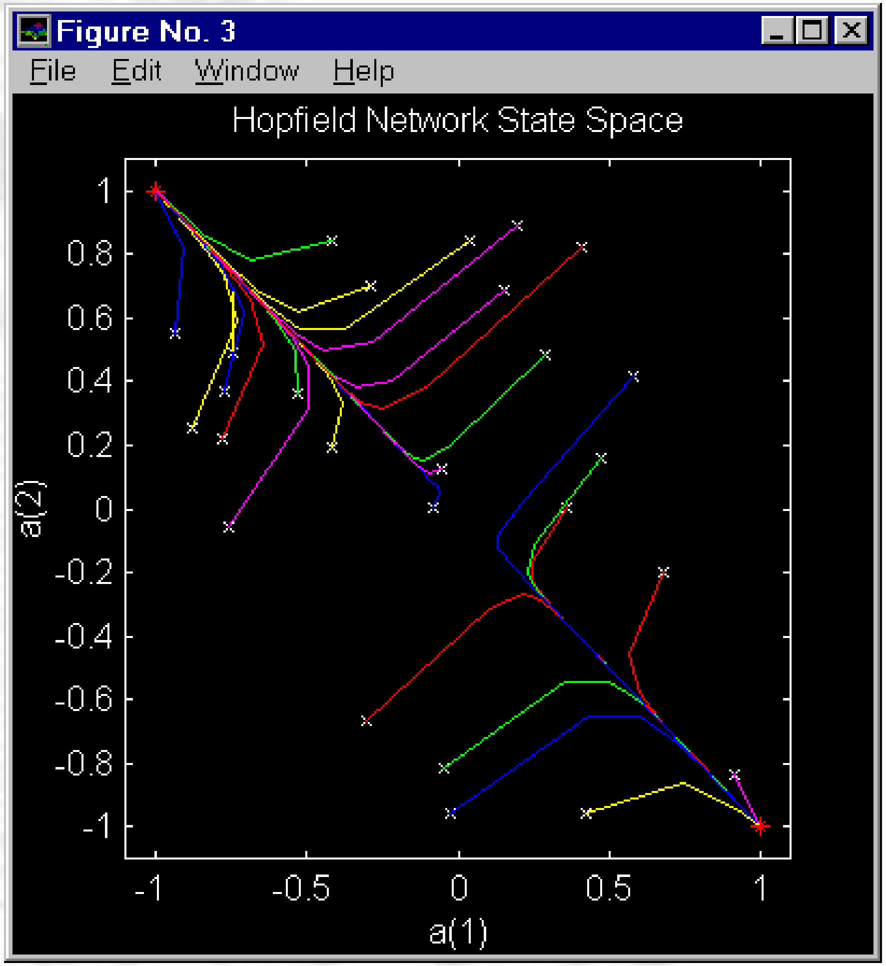


Figure No. 3
File Edit Window Help
Hopfield Network State Space
a(2)
a(1)
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
-1
-0.5
0
0.5
1

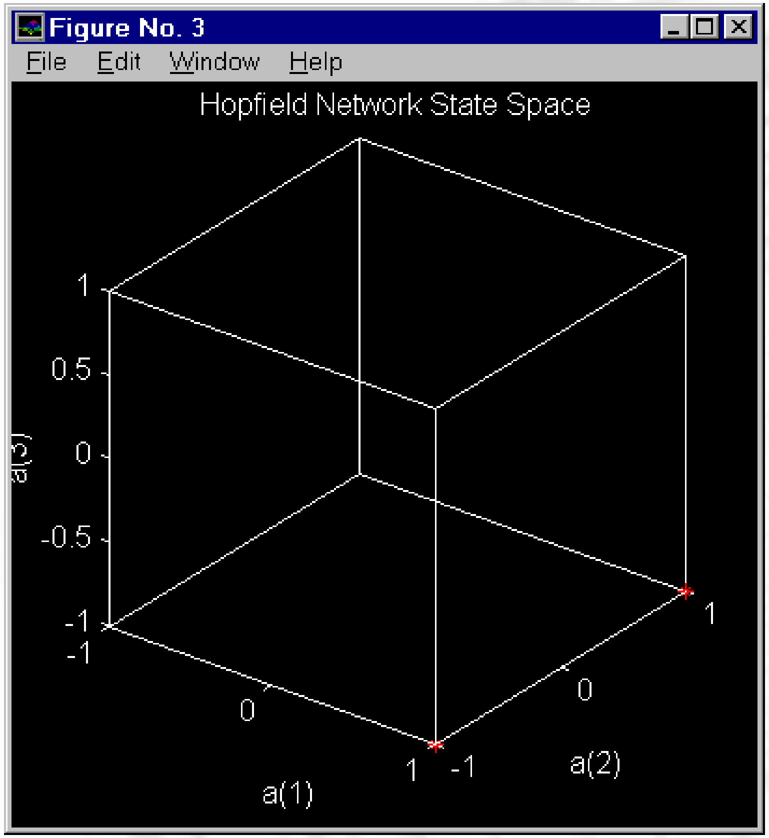
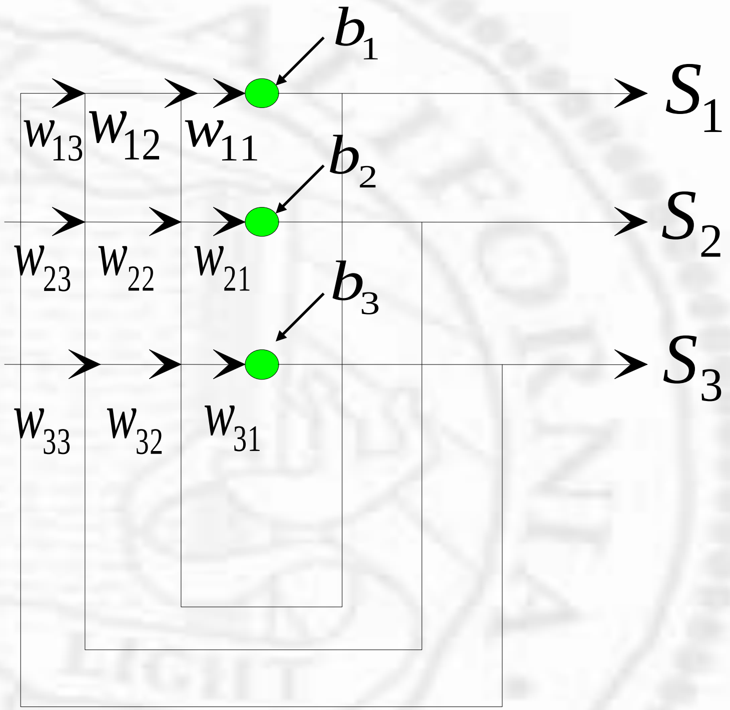


Figure No. 3
File Edit Window Help
Hopfield Network State Space
1
0.5
0
-0.5
-1
-1
0
1
-1
0
1
a(1)
a(2)
a(3)



b_1
b_2
b_3
S_1
S_2
S_3
w_{13} w_{12} w_{11}
w_{23} w_{22} w_{21}
w_{33} w_{32} w_{31}

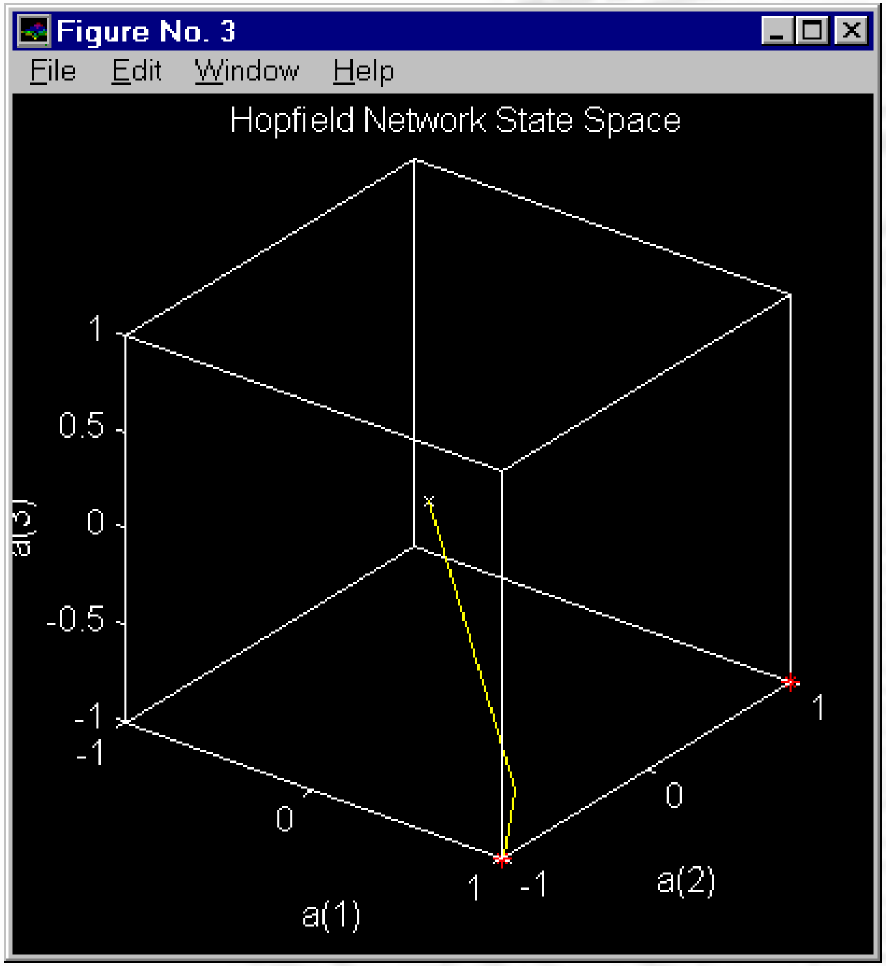


Figure No. 3
File Edit Window Help
Hopfield Network State Space
a(3)
a(1)
a(2)
1
0.5
0
-0.5
-1
-1
0
1
-1
0
1

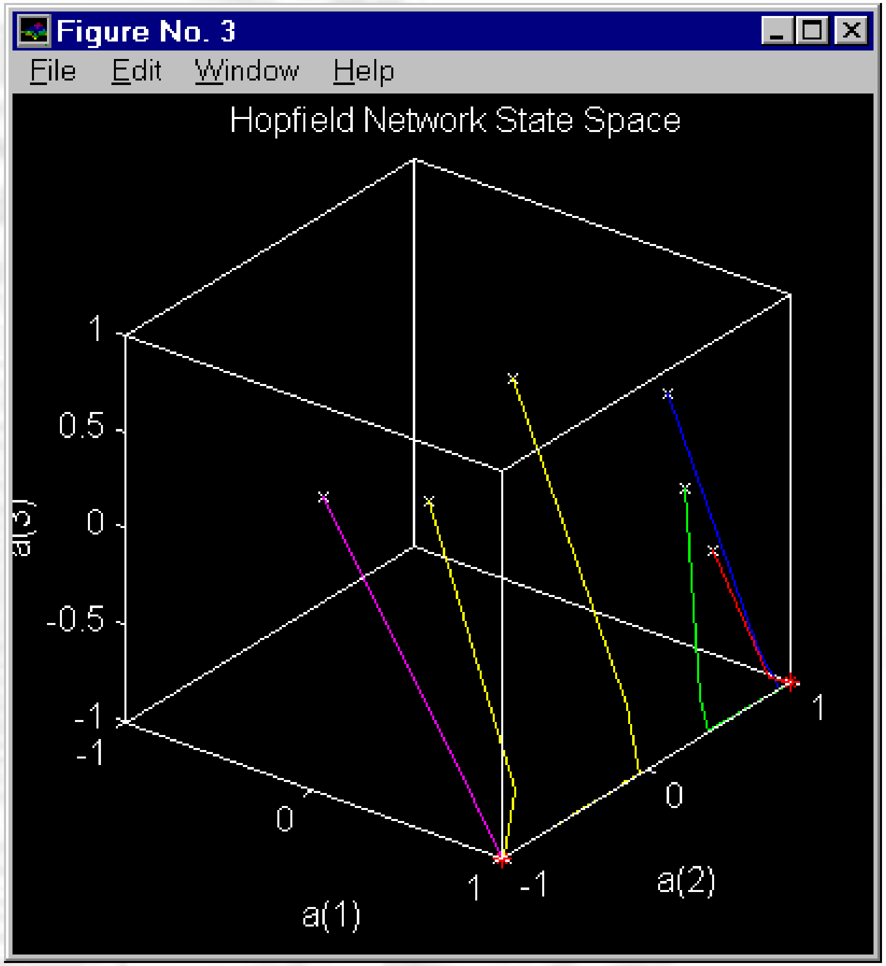
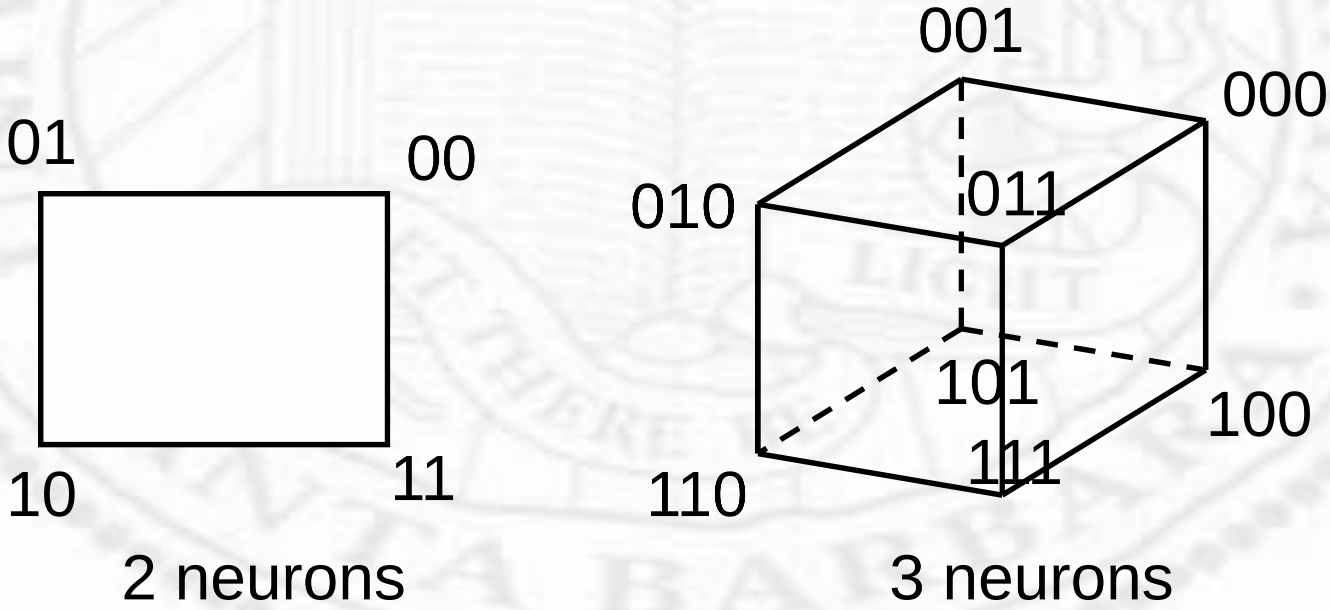


Figure No. 3
File Edit Window Help
Hopfield Network State Space
a(3)
a(1)
a(2)
1
0.5
0
-0.5
-1
-1
0
1
-1
0
1

Optimization Tool

- 2^n distinct states
- p stored values
- network moves from vertex to vertex until stabilization



Hopfield's Contribution

- Define an energy function (E) over the landscape
- E is non-increasing as the system evolves
- Stored patterns are local minimums
- E evolves according to Hebb's rule

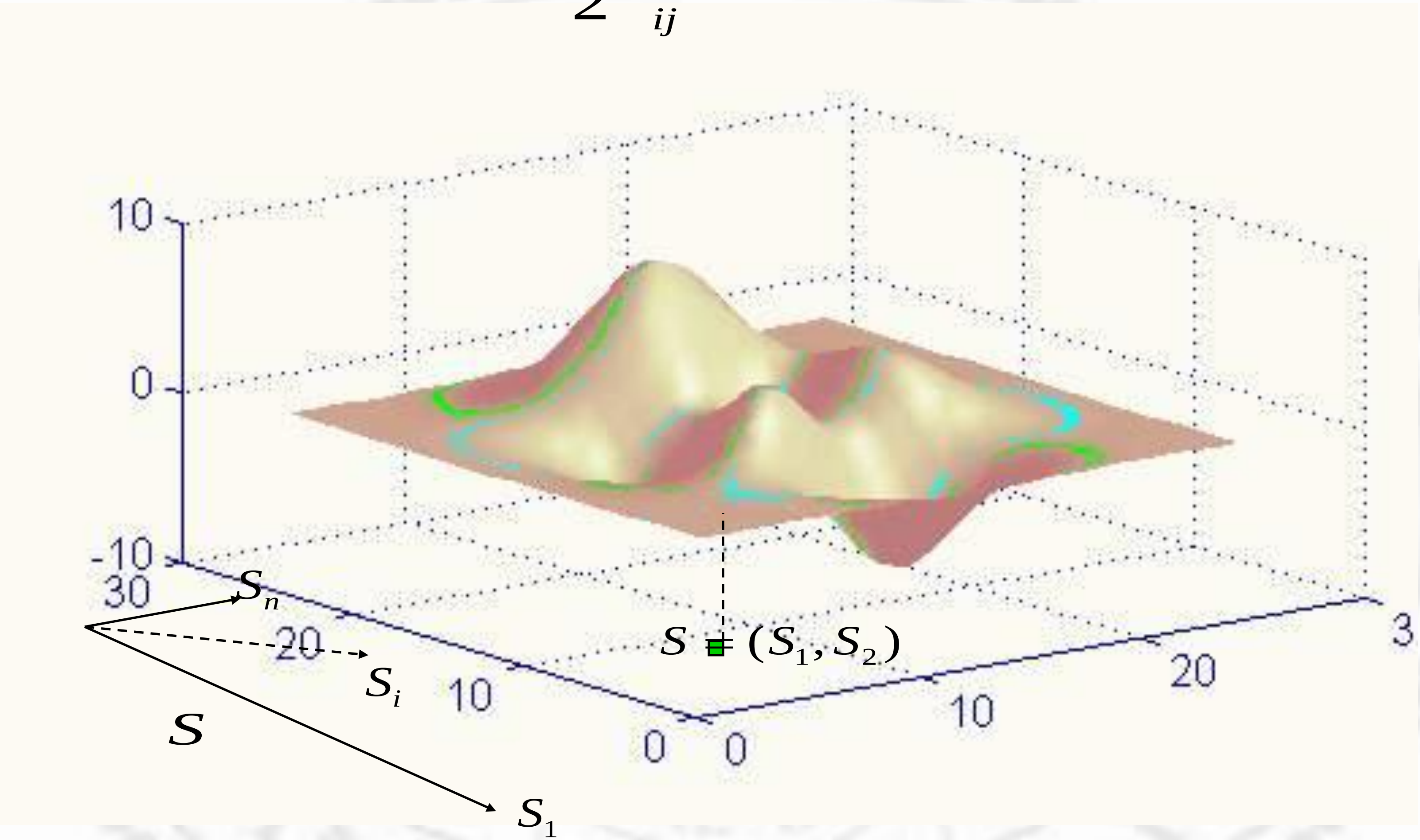
$$E = -\frac{1}{2}\sum_{ij} w_{ij} S_i S_j = -\frac{1}{2}\sum_{j} h_j S_j = -\frac{1}{2}\sum_{i} h_i S_i$$

Row sum

Column sum

$$E = -\frac{1}{2}\sum_{ij} w_{ij} S_i S_j + \sum_{i} \theta_i S_i$$

$$E = -\frac{1}{2}\sum_{ij} w_{ij} S_i S_j$$



Energy Function (cont.)

- ***E*** evolves according to Hebb's rule

$$\frac{\partial E}{\partial S_i} = -\sum_j w_{ij} S_j$$

- or Hebb's rule is simple gradient descent

- identify an energy function

- extract and store w_{ij} terms

- given input will relax to a local minimum

Energy Function

- ***E*** is ***non-increasing*** as the system evolves
  - Caveats: energy function exists if *w* is symmetric (e.g., by Hebb)
  - Sequential update model, neuron *p* update while all others held steady
- Before update: x_p

$$E(t) = -\frac{1}{2}\sum w_{ij}x_ix_j = -\frac{1}{2}\sum_{ij,i\neq p,j\neq p} w_{ij}x_ix_j - \frac{1}{2}\sum_j w_{pj}x_px_j - \frac{1}{2}\sum_i w_{ip}x_ix_p$$

- After update: X^*_p

$$E(t) = -\frac{1}{2}\sum w_{ij}x_ix_j = -\frac{1}{2}\sum_{ij,i\neq p,j\neq p} w_{ij}x_ix_j - \frac{1}{2}\sum_j w_{pj}x_p^*x_j - \frac{1}{2}\sum_i w_{ip}x_ix_p^*$$

Energy Function

- ***E*** change will only depends on terms with x_p and x^*_p

$$\Delta E = -\frac{1}{2}\sum_j w_{pj} x_p^* x_j - \frac{1}{2}\sum_i w_{ip} x_i x_p^* + \frac{1}{2}\sum_j w_{pj} x_p x_j + \frac{1}{2}\sum_i w_{ip} x_i x_p$$

- Remember that $w_{ij} = w_{ji}$,

$$\Delta E = \sum_i w_{pi} x_i (x_p - x_p^*)$$

- -1 to 1, $(x_p - x^*_p) = -2$, $\text{sum}(w_{pi} * x_i) > 0$ (accumulated input must be +)
- 1 to -1, $(x_p - x^*_p) = 2$, $\text{sum}(w_{pi} * x_i) < 0$ (accumulated input must be -)
- In either case, $\Delta E < 0$

Stored patterns as attractors (local minimums)

- Minimize when $S_i = \varepsilon_i$

$$E = -\frac{1}{2N}\left(\sum_i S_i \xi_i\right)^2$$

one pattern

$$E = -\frac{1}{2N}\sum_{u=1}^{p}\left(\sum_i S_i \xi_i^{u}\right)^2$$

p patterns

$$E = -\frac{1}{2N}\sum_{u=1}^{p}\left(\sum_i S_i \xi_i^{u}\right)\left(\sum_j S_j \xi_j^{u}\right)$$

$$= -\frac{1}{2}\sum_i \sum_j \left(\frac{1}{N}\sum_{u=1}^{p} \xi_i^{u} \xi_j^{u}\right) S_i S_j$$

$$= -\frac{1}{2}\sum_i \sum_j w_{ij} S_i S_j$$

Energy expression

Spurious states (attractors)

$$-\xi^u \because E = -\frac{1}{2N}\left(\sum_i S_i \xi_i\right)^2 = -\frac{1}{2N}\left[\sum_i S_i(-\xi_i)\right]^2$$

$$\because h_i^v = \sum_j w_{ij}(-\xi_j^v) = \frac{1}{N}\sum_j \sum_u \xi_i^u \xi_j^u (-\xi_j^v)$$

$$= -\xi_i^v + \frac{1}{N}\sum_j \sum_{u \neq v} \xi_i^u \xi_j^u \xi_j^v$$

$$\Rightarrow \operatorname{sgn}(h_i^v) = -\xi_i^v$$

$$\xi_i^{mix} = \operatorname{sgn}(\pm \xi_i^{u_1} \pm \xi_i^{u_2} \pm \xi_i^{u_3})$$

$$h_i^{mix} = \frac{1}{N}\sum_{j,u} \xi_i^u \xi_j^u \xi_j^{mix} = \frac{1}{2}\xi_i^{u_1} + \frac{1}{2}\xi_i^{u_2} + \frac{1}{2}\xi_i^{u_3}$$

$$+ cross - terms$$

Spurious states (attractors)

$$-\xi^u \because E = -\frac{1}{2N}(\sum_i S_i \xi_i)^2 = -\frac{1}{2N}[\sum_i S_i(-\xi_i)]^2$$

$$\because h_i^v = \sum_j w_{ij}(-\xi_j^v) = \frac{1}{N}\sum_j \sum_u \xi_i^u \xi_j^u (-\xi_j^v)$$

$$= -\xi_i^v + \frac{1}{N}\sum_j \sum_{u \neq v} \xi_i^u \xi_j^u \xi_j^v$$

$$\Rightarrow \mathrm{sgn}(h_i^v) = -\xi_i^v$$

$$\xi_i^{mix} = \mathrm{sgn}(\pm \xi_i^{u_1} \pm \xi_i^{u_2} \pm \xi_i^{u_3})$$

$$h_i^{mix} = \frac{1}{N}\sum_{j,u} \xi_i^u \xi_j^u \xi_j^{mix} = \frac{1}{2}\xi_i^{u_1} + \frac{1}{2}\xi_i^{u_2} + \frac{1}{2}\xi_i^{u_3}$$

$$+ cross - terms$$

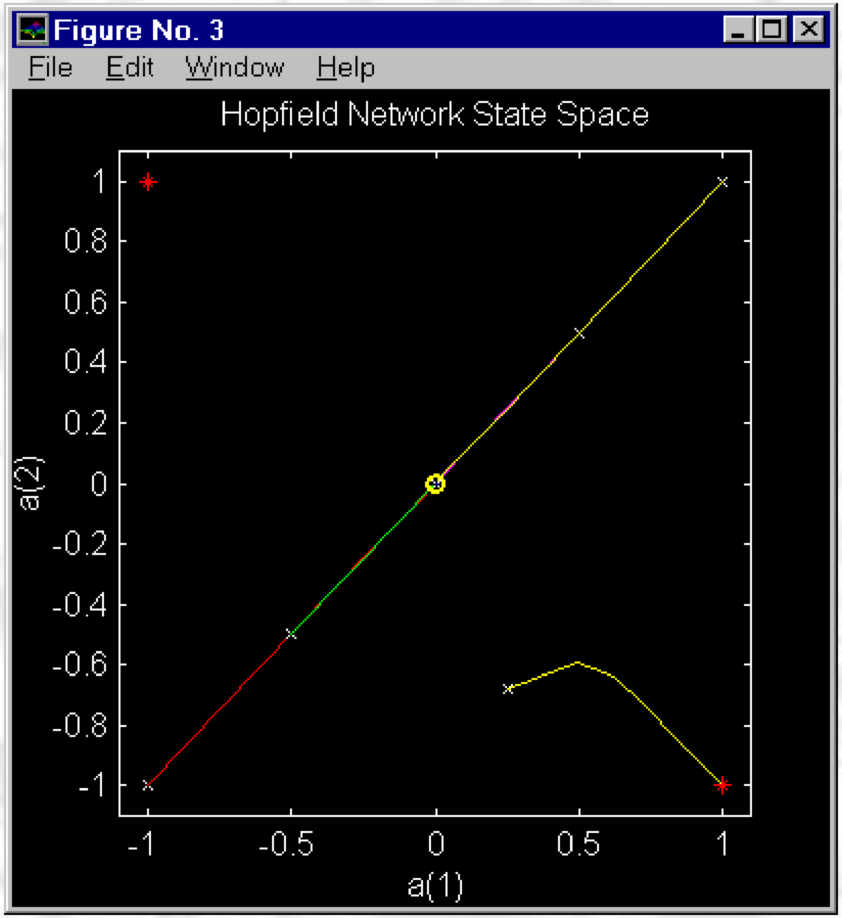


Figure No. 3
File
Edit
Window
Help
Hopfield Network State Space
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
-1
a(2)
-1
-0.5
0
0.5
1
a(1)

Caveats

- As associate memory, local minimum might be ok (the corrupted patterns are not far from the correct ones)
- As an optimization tool, it might *not* be ok to get stuck at local minimum
- However, Hopfield net using Hebb learning performs a *deterministic*, gradient descent search
- Other search techniques, more *stochastic* in nature, are needed for global minimum

Caveats (cont.)

- Techniques such as simulated annealing and ANN like Boltzman machine are needed for global minimum search

Simulated Annealing

- Randomness in search to jump out of local minimum
- Rely on an analogy with statistical mechanics

Simulated Annealing (cont.)

- Consider a system of a large number of particles and configurations (e.g., a bucket of water)
- An energy function is defined for each possible configuration of particles
- The likelihood of a particular configuration in thermal equilibrium is given by the Boltzmann-Gibbs distribution

$$P_{\alpha_i} = \frac{1}{Z} e^{-\frac{E_{\alpha_i}}{kT}} \qquad Z = \sum_i e^{-\frac{E_{\alpha_i}}{kT}}$$

k : *Boltzmann* constant

T : *temperature*

Simulated Annealing (cont.)

- At high temperature, all configurations are (*almost*) equally likely
  - The system can transit from low to high as easily it can from high to low
  - This corresponds to a global, coarse search
- At low temperature, configurations with small energy are preferred
  - The system transitions are mostly from high to low
  - this corresponds to a local, fine search

$$\frac{P_{\alpha_i}}{P_{\alpha_j}} = e^{-\frac{E_{\alpha_i} - E_{\alpha_j}}{kT}}$$

Simulated Annealing Procedure

- Start from high temperature and *gradually* lower the temperature
- Allow enough time for evolution at *each* temperature setting for equilibrium
- At each temperature setting, the system can evolve either by increasing or decreasing energy
- The probability of *increasing* system energy is controlled by temperature (the higher (lower) the temperature, the more (less) likely system will increase its energy)

- The transition probability is

$$P(\alpha_i \rightarrow \alpha_j) = \begin{cases} 1 & \Delta E = E_{\alpha_j} - E_{\alpha_i} < 0 \\ e^{-\frac{E_{\alpha_j} - E_{\alpha_i}}{kT}} & otherwise \end{cases}$$

- Can lead to
  - equilibrium
  - limit cycle
  - chaos
- Equilibrium requires

$$P_{\alpha_i} P(\alpha_i \rightarrow \alpha_j) = P_{\alpha_j} P(\alpha_j \rightarrow \alpha_i)$$

SA in Hopfield Networks

- Analogy: consider *S* forms a system with a large number of states
- Instead of using Hebb's rule which is gradient descent, the system is allowed to *increase* energy based on current *temperature*

SA in Hopfield Networks

- Recall that Hopfield energy definition is

$$E = -\frac{1}{2}\sum_{ij} w_{ij} S_i S_j$$

- If a change is made to, S_j, energy is going to change

$$\Delta E = E' - E = -\frac{1}{2}\sum_{ij} w_{ij} S_i S_j{}' - \frac{1}{2}\sum_{ij} w_{ij} S_i S_j = \sum_{i} w_{ij} S_i$$

SA in Hopfield Networks

- This can lead to an increase or a decrease in system energy
  - if energy decreases, great!, let it happen
  - if energy increases, not so great, let it happen by probability

$$P(S_j \rightarrow S_j^{'}) = e^{-\frac{\Delta E}{kT}}$$

An example - weight matching

- A set of N points
- with a known distance between each pair
- link points together in pairs
- each point is linked to exactly one other
- minimize total length of the link

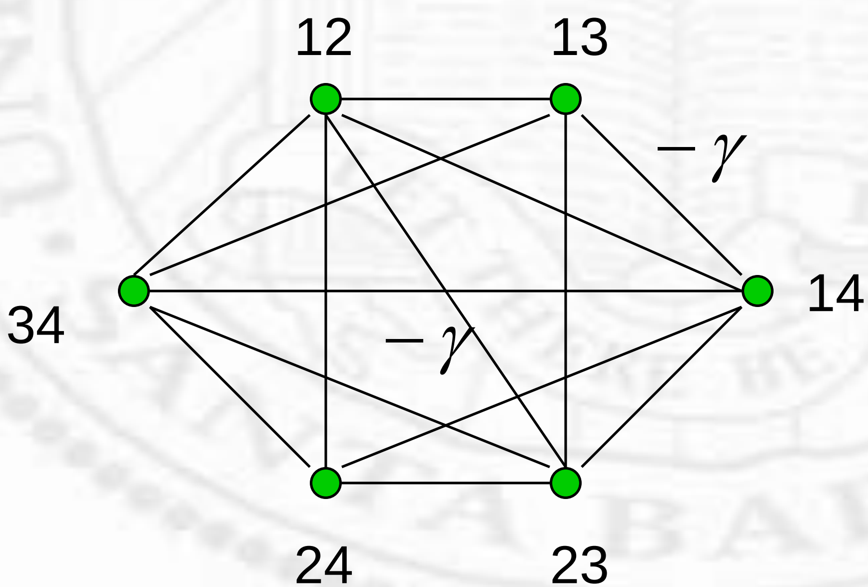
minimize $L = \sum_{i<j} d_{ij} n_{ij} \quad with \quad \sum_{j} n_{ij} = 1 \,(\,for\ all\ i\,)$

Energy function

$$H(n) = \sum_{i<j} d_{ij} n_{ij} + \frac{\gamma}{2} \sum_{i} (1 - \sum_{j} n_{ij})^2$$

$$= \frac{\gamma}{2} \sum_{i \neq j} {n_{ij}}^2 + \gamma \sum_{i} \sum_{\substack{j \neq i \\ k \neq i \\ j \neq k}} n_{ij} n_{ik} + \sum_{i<j} d_{ij} n_{ij} + \gamma \frac{N}{2}$$

$$= \frac{\gamma}{2} \sum_{k} S_k S_k + \gamma \sum_{k \neq l} S_k S_l + \sum_{k} d_k S_k + \gamma \frac{N}{2}$$

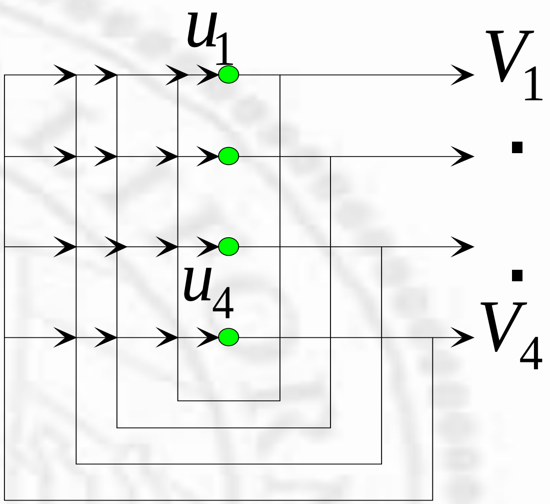


Extension - continuous inputs

$$V_i = g(u_i) = g(\sum_j w_{ij} V_j)$$

$$-1 \le V_i \le 1 \quad g(x) = \tanh(\beta x)$$

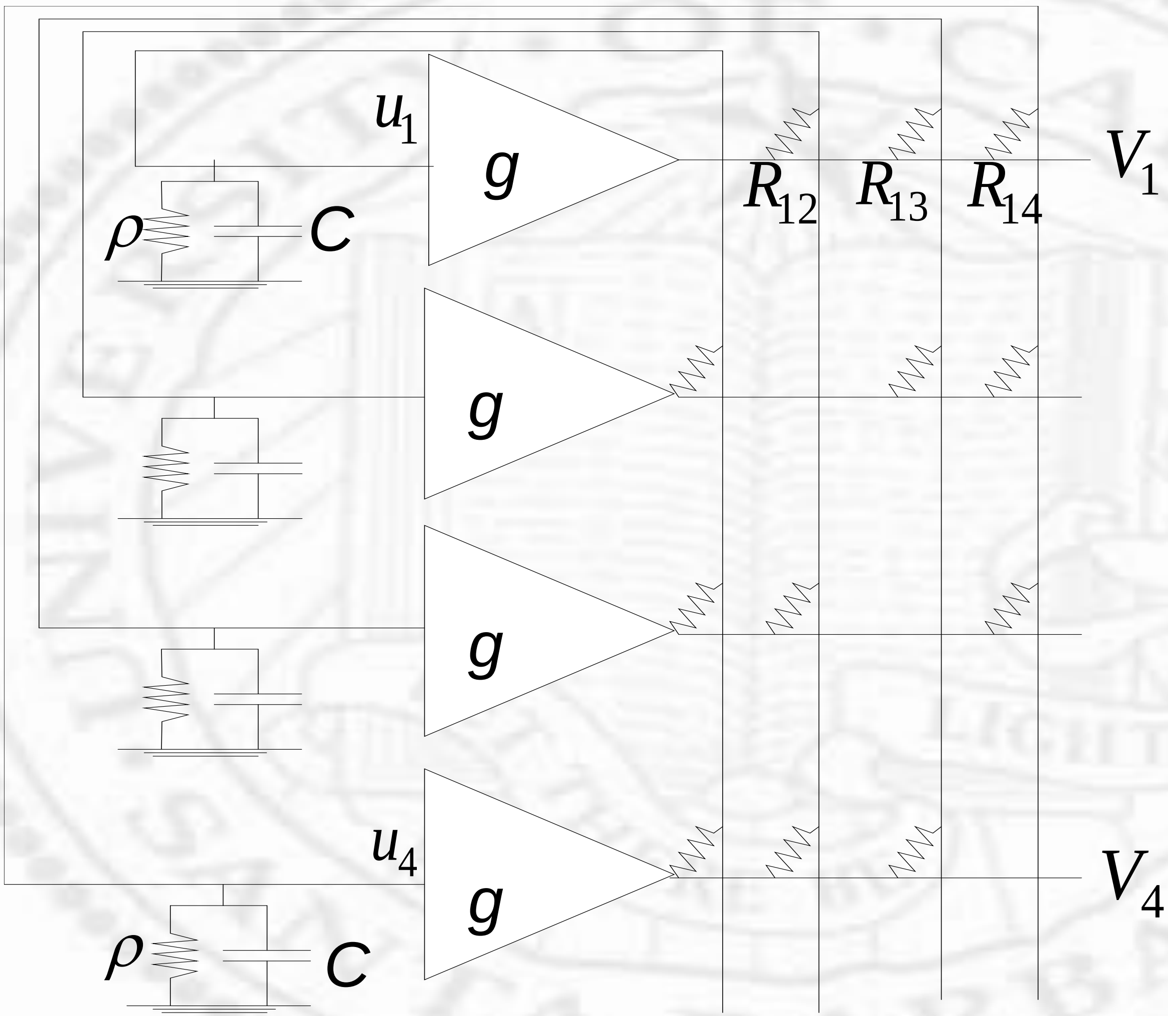
$$0 \le V_i \le 1 \quad g(x) = \frac{1}{1+e^{-2\beta x}}$$



$$\tau_i \frac{dV_i}{dt} = -V_i + g(u_i) = -V_i + g(\sum_j w_{ij} V_j)$$

$$\tau_i \frac{du_i}{dt} = -u_i + \sum_j w_{ij} V_j = -u_i + \sum_j w_{ij} g(u_j)$$

Hardware implementation



parameters

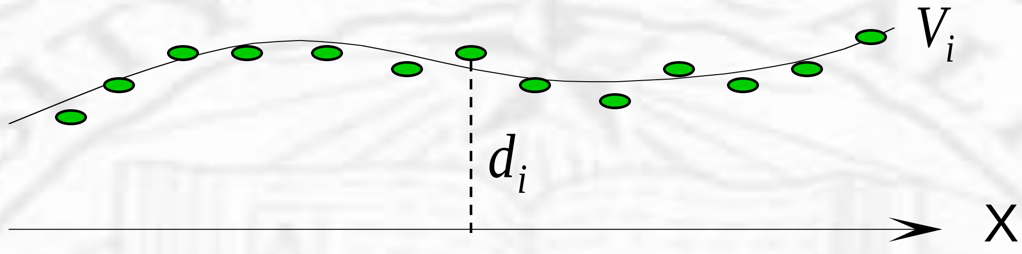
$$C\frac{du_i}{dt}+\frac{u_i}{\rho}=\sum_j \frac{1}{R_{ij}}(V_j-u_i)$$

$$\tau_i\frac{du_i}{dt}=-u_i+\sum_j w_{ij}g(u_j)$$

$$\tau_i=R_iC \qquad \frac{1}{R_i}=\frac{1}{\rho}+\sum_j\frac{1}{R_{ij}} \qquad w_{ij}=\frac{R_i}{R_{ij}}$$

$$\text{if } R_i\approx\rho \text{ then } w_{ij}=\frac{\rho}{R_{ij}}$$

An application - curve fitting



$$H = \frac{1}{2}\kappa\sum_i (V_i - V_{i+1})^2 + \frac{1}{2}\lambda\sum_i (V_i - d_i)^2$$

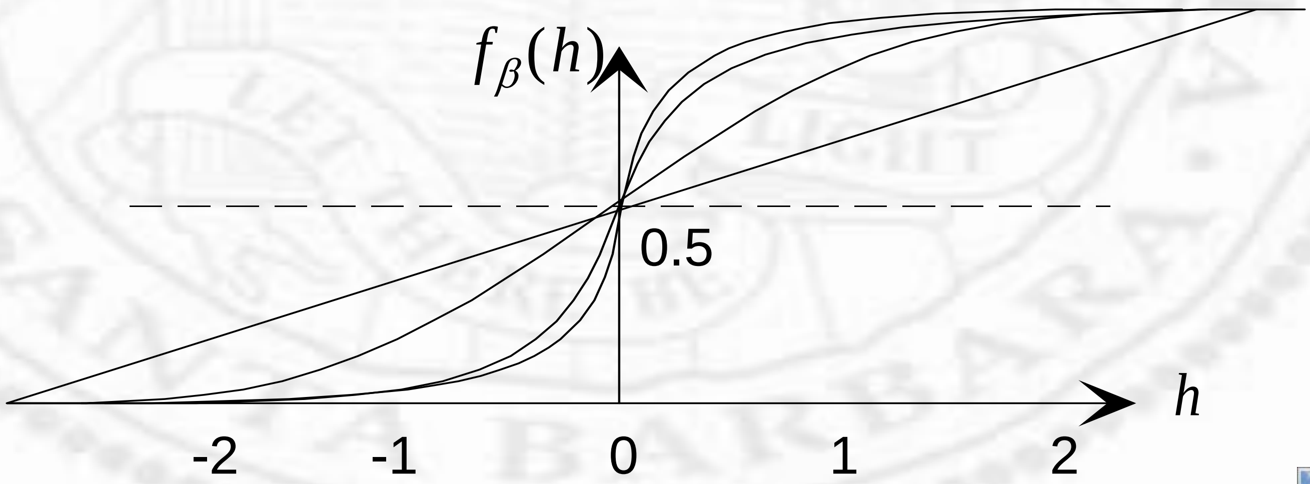
$$-\frac{\partial H}{\partial V_i} = \kappa(V_{i+1} - 2V_i + V_{i-1}) + \lambda(d_i - V_i)$$

$$\kappa\tau\frac{dV_i}{dt} = \kappa(V_{i+1} - 2V_i + V_{i-1}) + \lambda(d_i - V_i)$$

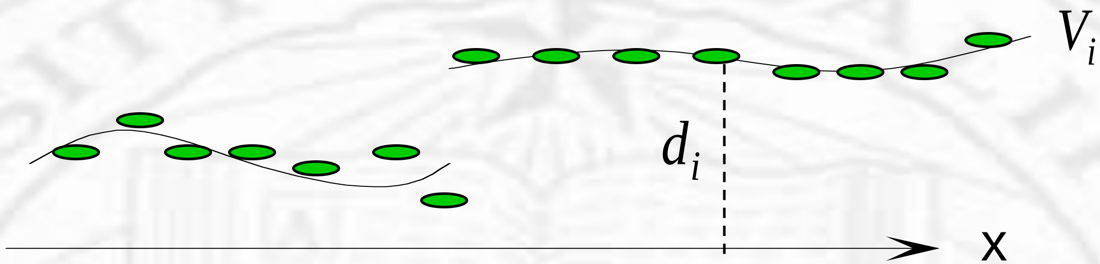
Extension - stochastic networks

- Analogy of statistical mechanics of magnetic systems
- Spin orientation as a probabilistic function of the temperature

$$P(S_i = \pm 1) = f_\beta(\pm h_i) = \frac{1}{1 + e^{\mp 2\beta h_i}} \qquad h_i = \sum_j w_{ij} S_j$$



An application - curve fitting with discontinuity



$$H = \frac{1}{2}\kappa\sum_i (1-S_i)(V_i - V_{i+1})^2 + \frac{1}{2}\lambda\sum_i (V_i - d_i)^2 + \mu\sum_i S_i$$

$$S_i \begin{cases} 1 & (\textit{line process}) \textit{ in between } V_i \textit{ and } V_{i+1} \\ -1 & \end{cases}$$

General Energy-Based Models

- Many (n) trapped particles in a container
- State (configuration) space (X) comprises locations $S_i = (x_i, y_i, z_i)$ of all these particles
- Each configuration has an energy value capturing the interactions (w_{ij}) of these particles ($E(X) = -\sum_{i,j} w_{ij} S_i S_j$)
- Likelihood (probability) of a state ∝ -energy and form a Boltzmann distribution (Z: partition function)

$$p(x) = \frac{e^{-E(x)}}{Z} \qquad Z = \sum_x e^{-E(x)}$$

General Energy-Based Models

- Binary, nearest neighbor interaction gives rise to Ising model explaining ferromagneism
  - An *n*-d lattice structure
  - Each particle spins up or down
  - Neighboring particles interact with each other
  - All particles subject to an environmental field

Energy: Hamiltonian function

$$H(\sigma) = -\sum_{\langle i\ j\rangle} J_{ij}\sigma_i\sigma_j - \mu\sum_j h_j\sigma_j$$

Probability: Boltzmann distribution

$$P_\beta(\sigma) = \frac{e^{-\beta H(\sigma)}}{Z_\beta} \qquad Z_\beta = \sum_\sigma e^{-\beta H(\sigma)}$$

In ANN

- A similar energy expression $E(X) = -\sum_{i,j} w_{ij} S_i S_j$ is often used (e.g., in Hopfield net)
- Particles (neurons): can be either visible (v) and clamped or hidden (h, latent)
- Two questions:
  - How to store
  - How to retrieve
- Both are more complicated with hidden

Caveats

- Reproduce a probability distribution that matches input
  - Using KL divergence as error (cost) function
- Generally, not possible to examine every location in the probability state space (even with binary neurons, *n* such neurons means 2^n state space)
  - Sampling (e.g., MCMC, Gibbs) is a must

KL Divergency

- Discrepancy (increase in code length) of using a code book tuned for one distribution for another
- *P*(i): base (observed) distribution with entropy (code length) $-\sum_i P(i) log P(i)$
- *Q*(i): test (recovered) distribution with entropy $(code\ length) - \sum_i P(i) log Q(i)$
- Increase in code length = $-\sum_i P(i) log Q(i)$-$(-\sum_i P(i) log P(i))$

$$D_{\mathrm{KL}}(P\|Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

KL Divergence

- Always positive, zero if $P=Q$
- Not symmetrical so not strictly a distance measurement
- Useful for BM for cost function: how observed distribution (P) differs from recovered distribution (Q)

Energy-Based Models

- Without hidden units (e.g., Hopfield)

$$p(x) = \frac{e^{-E(x)}}{Z}. \qquad Z = \sum_x e^{-E(x)}$$

- Likelihood $\prod_i p(x^{(i)})$
- L: log-likelihood, l: loss

$$\mathcal{L}(\theta, \mathcal{D}) = \frac{1}{N} \sum_{x^{(i)} \in \mathcal{D}} \log\ p(x^{(i)})$$

$$\ell(\theta, \mathcal{D}) = -\mathcal{L}(\theta, \mathcal{D})$$

- Minimize loss $-\frac{\partial \log p(x^{(i)})}{\partial \theta},$

- With hidden units (e.g., Boltzmann)

$$P(x) = \frac{e^{-\mathcal{F}(x)}}{Z} \qquad Z = \sum_x e^{-\mathcal{F}(x)}.$$

$$\mathcal{F}(x) = -\log \sum_h e^{-E(x,h)}$$

- Minimize loss

$$\boxed{-\frac{\partial \log p(x)}{\partial \theta} = \frac{\partial \mathcal{F}(x)}{\partial \theta} - \sum_{\tilde{x}} p(\tilde{x}) \frac{\partial \mathcal{F}(\tilde{x})}{\partial \theta}.}$$

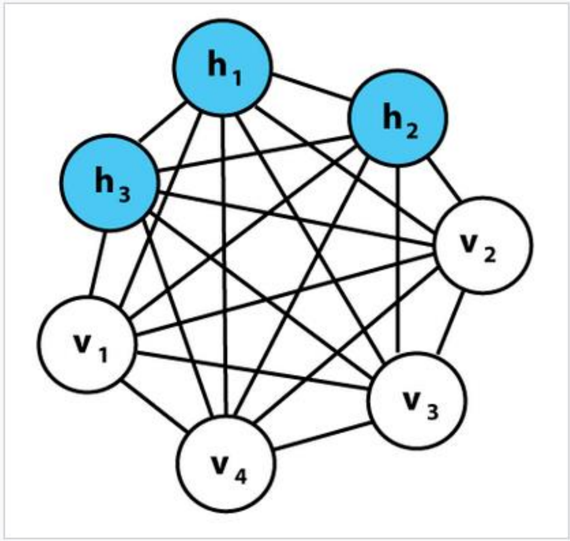
- Positive vs. negative phases
  - Increase p(samples) decrease p(samples from models)

Boltzmann Machine

- Stochastic, generative, recurrent neural network
- Maintain an internal representation (Hopfield is all external)
- Binary states (on or off)

$$S_i = \sigma\left(\sum_j w_{ij} S_j + b_i\right)$$

- Allow unconstrained connectivity
  - Between hidden and visible units
  - Between hidden units
  - Between visible units



Two Questions

- A learned Boltzmann machine (w_{ij} fixed)
  - Given some input (with error, missing data, etc.) how does it retrieve stored information?
  - Content-based retrieval: similar to Hopfield network but with hidden unit to "memorize" or "organize" information
- An unlearned Boltzmann machine (w_{ij} random)
  - How to impose v (visible) data and learn h (latent) variables?

Stochastic State Change

- Energy the same as Hopfield Net

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

- Change of energy from flipping a state

$$\Delta E_i = E_{\text{i=off}} - E_{\text{i=on}}$$

$$\Delta E_i = \sum_{j>i} w_{ij}\, s_j + \sum_{j<i} w_{ji}\, s_j + \theta_i$$

- Energy is proportional to the negative log probability of the state (less likely <-> higher energy, or Boltzmann distribution)

$$\Delta E_i = -k_B\, T \ln(p_{\text{i=off}}) - (-k_B\, T \ln(p_{\text{i=on}}))$$

$$p(x) = \frac{e^{-E(x)}}{Z}$$

$$P(x) \propto e^{-\frac{E}{kT}}$$

Stochastic State Rep

- Probability of state transition
  - Lower (higher) energy <-> high (low) probability

$$\Delta E_i = -k_B T \ln(p_{\text{i=off}}) - (-k_B T \ln(p_{\text{i=on}}))$$

$$\frac{\Delta E_i}{T} = \ln(p_{\text{i=on}}) - \ln(p_{\text{i=off}})$$

$$\frac{\Delta E_i}{T} = \ln(p_{\text{i=on}}) - \ln(1 - p_{\text{i=on}})$$

$$\frac{\Delta E_i}{T} = \ln\left(\frac{p_{\text{i=on}}}{1 - p_{\text{i=on}}}\right)$$

$$-\frac{\Delta E_i}{T} = \ln\left(\frac{1 - p_{\text{i=on}}}{p_{\text{i=on}}}\right)$$

$$-\frac{\Delta E_i}{T} = \ln\left(\frac{1}{p_{\text{i=on}}} - 1\right)$$

$$\exp\left(-\frac{\Delta E_i}{T}\right) = \frac{1}{p_{\text{i=on}}} - 1$$

$$\Rightarrow \quad p_{\text{i=on}} = \frac{1}{1 + \exp(-\frac{\Delta E_i}{T})}$$

$\Downarrow$

NOT change probability

Stochastic State Evolution

- Choose a unit, flip or not flip based on T (temperature)
  - High T, both flip and not flip are likely
  - Low T
    - Lower energy, high chance of flipping
    - Higher energy, low chance of flipping
- Equilibrium state
  - Approach Boltzmann distribution
  - Depend on T, not on initial configuration
  - Attractors are the final equilibrium states

Specification of Attractors

- Similar to Hebbian rules (as in Hopfield network), but
  - Visible states, V (settable) $P^+(V)$
  - Hidden states, H (not settable)
- After running, $P^-(V)$
- Want + and – to be the same, using KL divergence (v: all possible states)

$$G = \sum_{v} P^+(v) \ln\left(\frac{P^+(v)}{P^-(v)}\right)$$

GD Operations

$$G = \sum_v P^+(v) \ln\left(\frac{P^+(v)}{P^-(v)}\right)$$

$$\frac{\partial G}{\partial w_{ij}} = -\frac{1}{R}[p_{ij}^+ - p_{ij}^-]$$

$$\frac{\partial G}{\partial \theta_i} = -\frac{1}{R}[p_i^+ - p_i^-]$$

- Positive clamping – visible unit clamped according to P^+
- Negative phase – no clamping
- P^+ij: i and j both on in positive phase
- P^-ij: i and j both on in negative phase

Details of GD

- X (state), V (visible), H (hidden): X = V+H
- Likelihood of Observing V=v : $L(\theta|v) = p(v|\theta)$, $\theta=\{\omega_{ij}\}$ (<= Bayes rule)
- Log likelihood

$$\ln \mathcal{L}(\theta \mid S) = \ln \prod_{i=1}^{\ell} p(x_i|\theta) = \sum_{i=1}^{\ell} \ln p(x_i|\theta)$$

Details of GD (cont.)

- q(x): the distribution underlying the observation (x_i)
- p(x): the distribution of the BM (based on parameters w_{ij})
- Minimize KL difference as error measurement (only 2nd term depends on BM)

$$\mathrm{KL}(q||p) = \sum_{x\in\Omega} q(x)\ln\frac{q(x)}{p(x)} = \sum_{x\in\Omega} q(x)\ln q(x) - \sum_{x\in\Omega} q(x)\ln p(x)$$

- Maximize log-likelihood ln(p(x))

Details of GD (cont.)

$$\theta^{(t+1)} = \theta^{(t)} + \underbrace{\boxed{\eta \frac{\partial}{\partial \theta^{(t)}} \left(\sum_{i=1}^{N} \ln \mathcal{L}(\theta^{(t)} | x_i) \right)} - \boxed{\lambda \theta^{(t)}} + \boxed{\nu \Delta \theta^{(t-1)}}}_{:= \Delta \theta^{(t)}}$$

- Red: vanilla gradient descent
- Green: regularization term (from θ^2)
- Blue: momentum term
- An added twist: there are both visible and hidden states

Gradient of Log likelihood

$$\ln \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{v}) = \ln p(\boldsymbol{v} \mid \boldsymbol{\theta}) = \ln \frac{1}{Z} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} = \ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} - \ln \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$$

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta} \mid \boldsymbol{v})}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \right) - \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \right)$$

$$= -\frac{1}{\sum\limits_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} + \frac{1}{\sum\limits_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$$

$$= -\sum_{\boldsymbol{h}} p(\boldsymbol{h} \mid \boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$$

$$\Downarrow$$

$$p(\boldsymbol{h} \mid \boldsymbol{v}) = \frac{p(\boldsymbol{v},\boldsymbol{h})}{p(\boldsymbol{v})} = \frac{\frac{1}{Z} e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\frac{1}{Z} \sum\limits_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} = \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\sum\limits_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}}$$

$$\frac{\partial lnL(w_{ij}|v)}{\partial w_{ij}} = -\sum_{h} p(h|v) S_i S_j + \sum_{v,h} p(v,h) S_i S_j$$

Details of GD

$$\frac{\partial lnL(w_{ij}|v)}{\partial w_{ij}} = -\sum_{h} p(h|v)S_iS_j + \sum_{v,h} p(v,h)S_iS_j$$

$$\rho_{ij}^{+} = \sum_{v}\sum_{h} p(h|v)S_iS_j$$

$$\rho_{ij}^{-} = \sum_{v}\sum_{h} p(v,h)S_iS_j$$

- +: correlation in the positive state (clamping v)
- -: correlation in the negative state (clamping nothing, day dreaming)

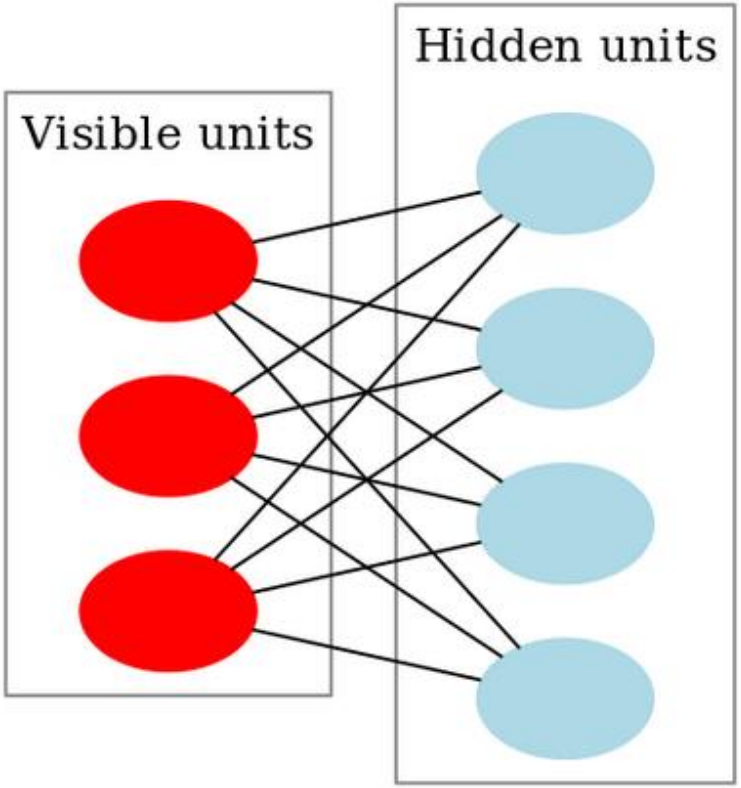
In Reality

$$\frac{\partial \ln L(w_{ij}|v)}{\partial w_{ij}} = -\sum_{h} \boxed{p(h|v)} S_i S_j + \sum_{v,h} \boxed{p(v,h)} S_i S_j$$

- The energy functions
  - Under model distribution of the hidden variables given training samples
  - Under pure model distribution
- Are exponential in the number of states
- MCMC (Gibbs) is used to obtain a sampling based estimate

Restricted Boltzmann Machine

- Does not allow unconstrained connectivity
  - Between hidden and visible units
  - Between hidden units (x)
  - Between visible units (x)



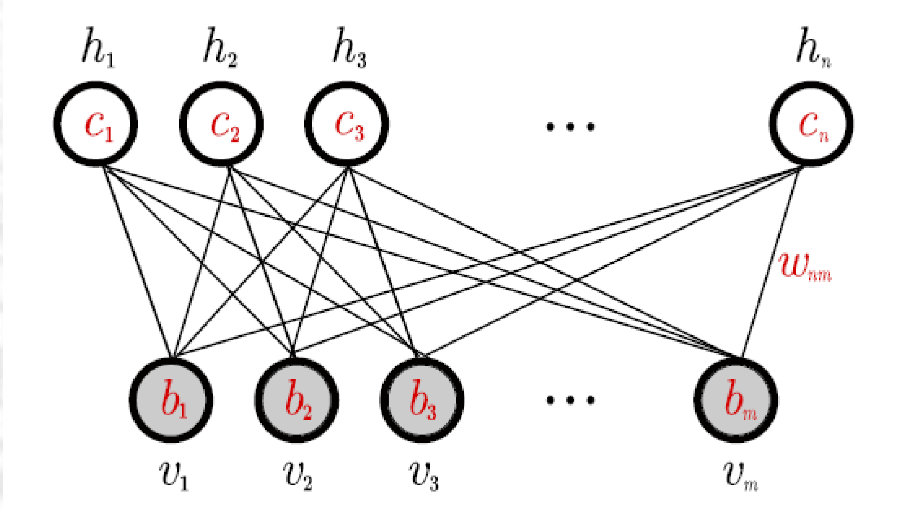
Training

- Think about Auto-encoder
  - Forward (from visible to hidden)
    - Clamp visible to input, compute hidden
  - Backward (from hidden to visible)
    - Nothing clamped
- Goal: Forward + backward should reproduce original pattern of probability
- Again, error is in KL divergence
  - Much faster with simplified structures

Conditional Independence

- A Markov Random Field property
  - Hidden units are independent given the visible unit they connect to
  - Visible units are independent give hidden unit they connect to

$$p(\boldsymbol{h} \mid \boldsymbol{v}) = \prod_{i=1}^{n} p(h_i \mid \boldsymbol{v}) \;\; \text{and} \;\; p(\boldsymbol{v} \mid \boldsymbol{h}) = \prod_{i=1}^{m} p(v_i \mid \boldsymbol{h})$$



$$E(\boldsymbol{v},\boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i$$

$$p(\boldsymbol{v}) = \frac{1}{Z}\sum_{\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) = \frac{1}{Z}\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$$

$$= \frac{1}{Z}\sum_{h_1}\sum_{h_2}\cdots\sum_{h_n} e^{\sum\limits_{j=1}^{m} b_j v_j} \prod_{i=1}^{n} e^{h_i\left(c_i + \sum\limits_{j=1}^{m} w_{ij} v_j\right)}$$

$$= \frac{1}{Z} e^{\sum\limits_{j=1}^{m} b_j v_j} \sum_{h_1} e^{h_1\left(c_1 + \sum\limits_{j=1}^{m} w_{1j} v_j\right)} \sum_{h_2} e^{h_2\left(c_2 + \sum\limits_{j=1}^{m} w_{2j} v_j\right)} \cdots \sum_{h_n} e^{h_n\left(c_n + \sum\limits_{j=1}^{m} w_{nj} v_j\right)}$$

$$= \frac{1}{Z} e^{\sum\limits_{j=1}^{m} b_j v_j} \prod_{i=1}^{n} \sum_{h_i} e^{h_i\left(c_i + \sum\limits_{j=1}^{m} w_{ij} v_j\right)}$$

$$= \frac{1}{Z} \prod_{j=1}^{m} e^{b_j v_j} \boxed{\prod_{i=1}^{n} \left(1 + e^{c_i + \sum\limits_{j=1}^{m} w_{ij} v_j}\right)} \qquad (22)$$

Product of experts

Faster Update – Contrastive Divergent (approximate GD)

- For each sample
  - "+" : set *v* to sample, for each hidden (*h*) state
    - Compute activation for h_i $\quad \text{sigmod}(\sum_j w_{ij} v_j)$
    - Turn h_i on with probability $\quad \text{sigmod}(\sum_j w_{ij} v_j)$
    - Compte $e_{ij}^{+} = h_i v_j$
  - "-": For each visible (*v*) state
    - Compute activation for v_j $\quad \text{sigmod}(\sum_i w_{ij} h_i)$
    - Turn v_j on with probability $\quad \text{sigmod}(\sum_i w_{ij} h_i)$
    - Compte $e_{ij}^{-} = h_i v_j$
  - Update with $w_{ij} = L(e_{ij}^{+} - e_{ij}^{-})$ (L: learning rate)

Deep Belief Network

- Think about Auto-encoder
  - Forward (from visible to hidden)
    - Clamp visible to input, compute hidden
  - Backward (from hidden to visible)
    - Nothing clamped
- Goal: Forward + backward should reproduce original pattern
- The hidden units become the visible units of the next layer
- Learned layer by layer with fine tuning at the end by backpropagation