Recurrence networks
Full matrix of connections (bidirectional)
\( w_{ij} \): weight from node \( j \) to node \( i \)
\( b_i \): bias for node \( i \)
Network state evolves over time, to some equilibrium state or stationary distribution

Hopfield network
Recurrent network model of pattern completion
- Attractor network: settles to nearest stored state
- Content-addressable memory: retrieves a pattern from partial information

Architecture
- Symmetric connections, \( w_{ij} = w_{ji} \)
- No self-connections, \( w_{ii} = 0 \)
- Binary activation, \( \pm 1 \)

Dynamics
- Asynchronous deterministic updating
  \[ a_i \leftarrow \text{sign} \left( \sum_j w_{ij} a_j + b_i \right) \]
  with \( i \) chosen randomly on each step

Energy function
- Measure of discordance of a network state (cf. Smolenky’s harmony)
  \[ E(a) = -\sum_{ij} a_i a_j w_{ij} - \sum_i a_i b_i = -a^T W a - b^T a \]
- Update rule always reduces energy
  \[ \Delta E = \Delta a_i \cdot \left( -\sum_j w_{ij} a_j - b_i \right) \]
  negative if \( a_i \) changed
- Attractors are local minima

Hebbian learning
- Set network to some state \( a \)
  \[ \Delta w_{ij} = \frac{1}{N} a_i a_j \]
- Trained patterns become attractors

Stability of trained patterns
- Train on patterns \( a^k \) for \( k \in \{1, \ldots, K\} \)
  \[ w_{ij} = \frac{1}{N} \sum_k a^k_i a^k_j \]
- Consider network in state \( a^l \)
  \[ a^l_i = \sum_j w_{ij} a^l_j + \frac{1}{N} \sum_k \sum_{j \neq l} a^k_i a^k_j a^l_j + \frac{1}{N} \sum_k \sum_j a^k_j a^k_i a^l_j \]

Crossstalk term
- Interference among patterns
  - Determines storage capacity of network, i.e. before trained patterns are no longer attractors
  - Random patterns, large \( N \): phase transition at \( K \approx 0.138 \cdot N \)

Boltzmann machine
Probabilistic updating
- \( a \in \{0,1\} \)
  \[ \text{Pr}[a_i = 1|a_{-i}] = \text{logistic} \left( \frac{a^l_i}{T} \right) = \frac{1}{1 + e^{-\left( \sum_j w_{ij} a_j + b_i \right)/T}} \]
  \[ \text{Pr}[a_i|a_{-i}] \propto e^{\left( \sum_j w_{ij} a_j + b_i \right)a_i/T} \]

Boltzmann distribution
- From statistical mechanics (thermodynamics)
  \[ \text{Pr}[a] \propto e^{-E(a)/T} \]
  - Same equation as softmax

Boltzmann machine and Gibbs sampling
- Hopfield energy \( E(a) = -a^T W a - b^T a \)
  \[ \text{Pr}[a_i = 1|a_{-i}] = \frac{e^{-E(1,a_{-i}) - E(0,a_{-i})}/T}{e^{-E(1,a_{-i}) - E(0,a_{-i})}/T} \]
  \[ = e^{-\left( \sum_j w_{ij} a_j + b_i \right)/T} \]
  - Thus update rule converges to Boltzmann distribution
- Logistic (and linear transformations thereof) is only function where conditional probs are mutually consistent
Stationary distribution under Hebbian learning

\[ E(a) = -a^T W a = -\frac{1}{N} a^T \left( \sum_k a^k a^{k^T} \right) a = -\frac{1}{N} \sum_k (a \cdot a^k)^2 \]

(complications around \{0,1\} vs. \{-1,1\} representation)

Concentrated around attractors
Extremely slow mixing
Simulated annealing: reduce temperature over time

Restricted Boltzmann machine

Bipartite, input and hidden
Visible units \( v_i \), hidden units \( h_j \)
Symmetric weights \( w_{ij} \)
No intralayer connections: faster convergence
Conditionally independent sampling: \( p_{\text{net}}(h|v) \propto e^{v^T W h + b^T h} \), \( p_{\text{net}}(v|h) \propto e^{v^T W h + b^T v} \)

Training
Input patterns \( v \sim p_{\text{in}} \) (training distribution)
Goal: network reproduces input distribution
Gibbs/Boltzmann distribution marginalized over \( h \)
\( p_{\text{net}}(v) = \sum_h p_{\text{net}}(v, h) \approx p_{\text{in}}(v) \)
Hidden layer learns latent features

Boltzmann machine learning rule
Maximum likelihood of training patterns, by gradient descent
\[ \prod_k p_{\text{net}}(v^k) \to \sum_k \ln p_{\text{net}}(v^k) \]
Training set as proxy for true generating distribution: \( \sum_v p_{\text{in}}(v) \ln p_{\text{net}}(v) \)
Equivalent to maximizing \( \sum_v p_{\text{in}}(v) (\ln p_{\text{net}}(v) - \ln p_{\text{in}}(v)) = KL(p_{\text{in}}|p_{\text{net}}) \)

\[ \frac{\partial}{\partial w_{ij}} \ln p_{\text{net}}(v^k) = \frac{\partial}{\partial w_{ij}} \ln \sum_h e^{-E(v^k, h)} = \frac{1}{\sum_h e^{-E(v^k, h)}} \sum_h \frac{\partial}{\partial w_{ij}} e^{-E(v^k, h)} \]

Contrastive divergence algorithm
Small number of steps for negative phase (e.g., 1 step) rather than convergence
- Sample \( v^+ \) from training set
- Sample \( h^+ \) from \( p_{\text{net}}(h|v^+) \)
- Calculate \( W^+ = v^T v^+ \)
- Gibbs update: Sample \( v^- \) from \( p_{\text{net}}(v|h^+) \), and sample \( h^- \) from \( p_{\text{net}}(h|v^-) \)
- Calculate \( W^- = v^- T h^- \)
- Update \( \Delta W = \epsilon (W^+ - W^-) \), \( \Delta b_v = \epsilon (v^+ - v^-) \), \( \Delta b_h = \epsilon (h^+ - h^-) \)

Hierarchical Boltzmann machine
Stacked RBMs
Trained sequentially
Hierarchy of features of increasing complexity
Often back-propagation training for fine-tuning