

What is good about back-propagation

1. Discovering useful internal representations
 - Tradeoff between *memorization* and *generalization*
 - Networks generalize on the basis of *similarity*
 - Similarity in environmental input often does not indicate similarity in the appropriate behavioral (or internal) response
 - Networks must learn to re-represent inputs (over internal “hidden” units) in a way that corresponds more closely to output similarity
2. Temporal processing
 - Can learn to interpret time-varying input and produce time-varying output, including long-distance temporal dependencies
3. Efficiency
 - Relatively efficient in terms of amount of experience required for learning (well, not really, but it's the best we've got)

Alternatives to back-propagation

1. *Requires explicit targets (supervised)*
 - Unsupervised learning (e.g., **competitive learning**)
 - Incorporate general statistical biases/assumptions
 - Reinforcement learning
 - Environment does not specific appropriate actions but rather provides feedback on *efficacy* of actions
 - Feedback may be delayed and/or intermittent
 - Forward modeling
 - Learn internal model of how actions influence outcomes
 - Use model to determine how to modify actions to improve outcomes

What is wrong with back-propagation

1. Requires explicit targets for output units
2. Requires propagation of error signals backwards across units and connections
3. Requires real-valued (differentiable) unit functions
4. Requires storage of past state information (or strict control structure) to learn temporal behavior

Alternatives to back-propagation

1. *Requires explicit targets (supervised)*
 - Unsupervised learning (e.g., **competitive learning**)
 - Incorporate general statistical biases/assumptions
 - Reinforcement learning
 - Environment does not specific appropriate actions but rather provides feedback on *efficacy* of actions
 - Feedback may be delayed and/or intermittent
 - Forward modeling
 - Learn internal model of how actions influence outcomes
 - Use model to determine how to modify actions to improve outcomes
- 2-3. *Requires error propagation and real-valued units*
 - Contrastive Hebbian (Boltzmann) learning
 - Both forward (negative) and backward (positive) passes involve only propagation of activation
 - Still supervised; binary stochastic version is *slow*

Alternatives to back-propagation

1. Requires explicit targets (supervised)

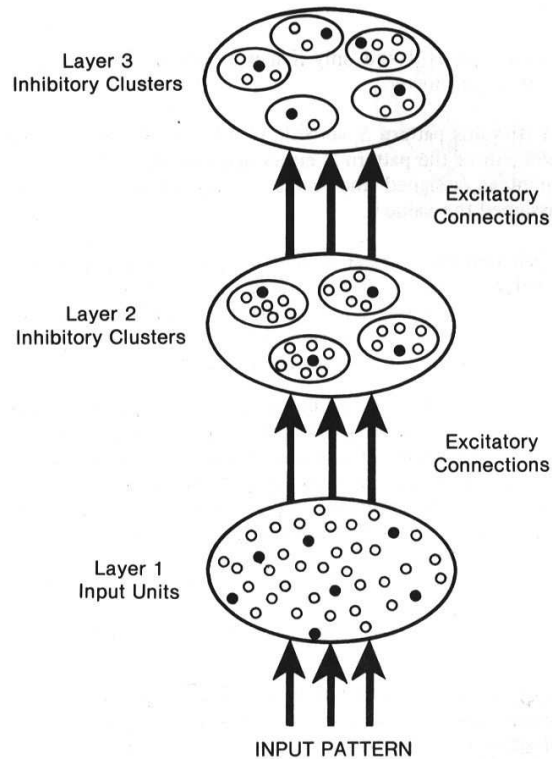
- Unsupervised learning (e.g., **competitive learning**)
 - Incorporate general statistical biases/assumptions
- Reinforcement learning
 - Environment does not specific appropriate actions but rather provides feedback on *efficacy* of actions
 - Feedback may be delayed and/or intermittent
- Forward modeling
 - Learn internal model of how actions influence outcomes
 - Use model to determine how to modify actions to improve outcomes

2-3. Requires error propagation and real-valued units

- Contrastive Hebbian (Boltzmann) learning
 - Both forward (negative) and backward (positive) passes involve only propagation of activation
 - Still supervised; binary stochastic version is *slow*

4. Requires storage of past states

- different integration rates, delay lines, ???



Competitive learning

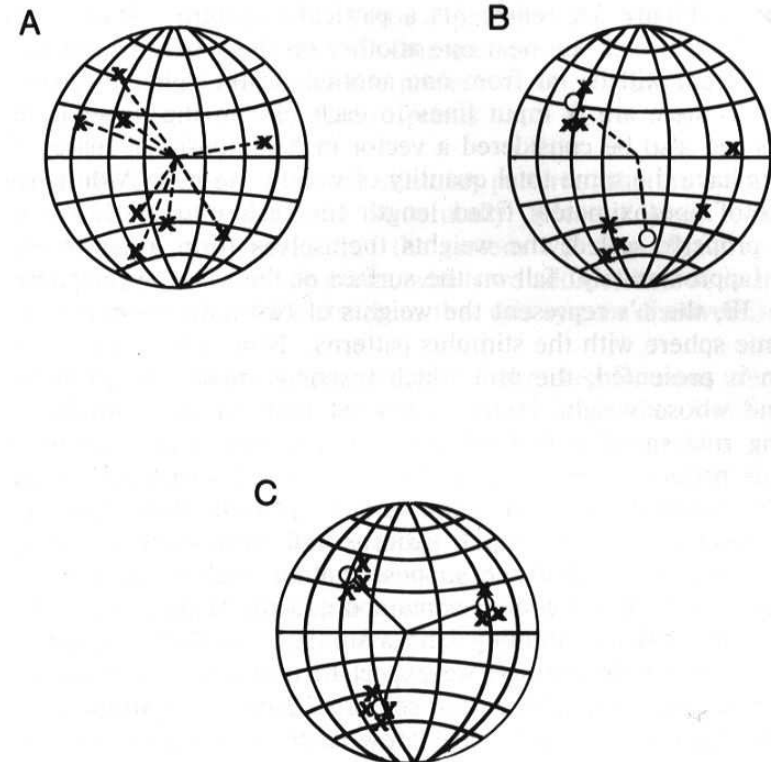
- Units in a layer are organized into non-overlapping cluster of competing units
- Each unit has a fixed amount of total weight to distribute among its input lines (usually $\sum_i w_{ij} = 1$)
- All units in a cluster receive the same (binary) input pattern
- The most active unit in a cluster shifts weight from inactive to active input lines:

$$\Delta w_{ij} = \begin{cases} 0 & \text{if unit } j \text{ loses on stimulus } k \\ g \frac{c_{ik}}{n_k} - gw_{ij} & \text{if unit } j \text{ wins on stimulus } k \end{cases}$$

where c_{ik} is equal to 1 if in stimulus pattern S_k , unit i in the lower layer is active and zero otherwise, and n_k is the number of active units in pattern S_k (thus $n_k = \sum_i c_{ik}$).

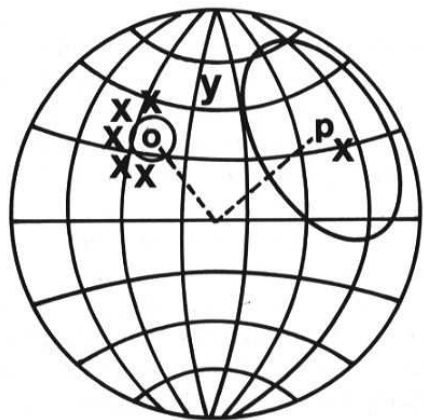
$$\Delta w_{ij} = \epsilon \left(\frac{a_i}{\sum_k a_k} - w_{ij} \right)$$

- Units gradually come to respond to clusters of similar inputs

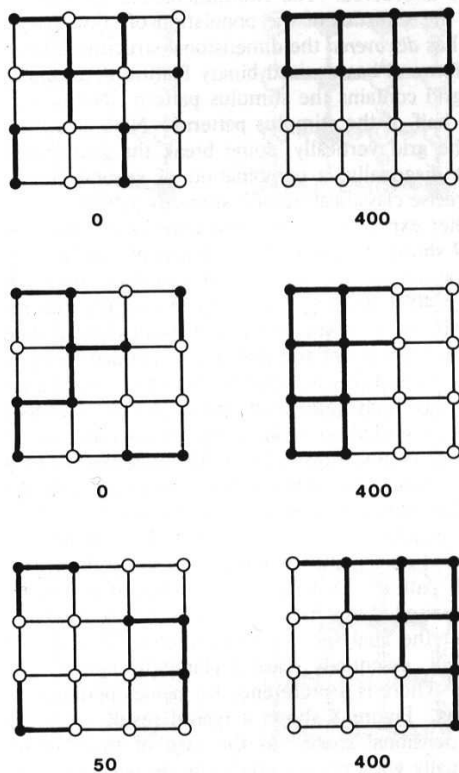
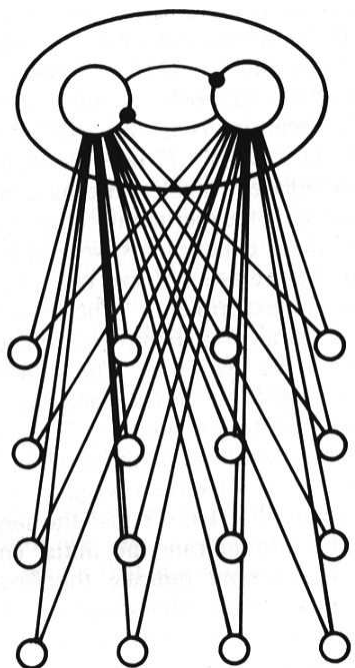
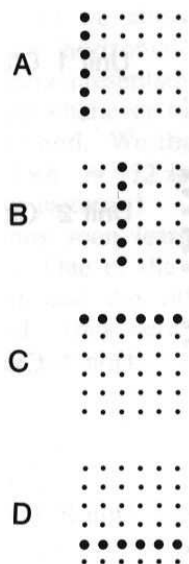


$$\Delta w_{ij} = \begin{cases} g_l \frac{c_{ik}}{n_k} - g_l w_{ij} & \text{if unit } j \text{ loses on stimulus } k \\ g_w \frac{c_{ik}}{n_k} - g_w w_{ij} & \text{if unit } j \text{ wins on stimulus } k \end{cases}$$

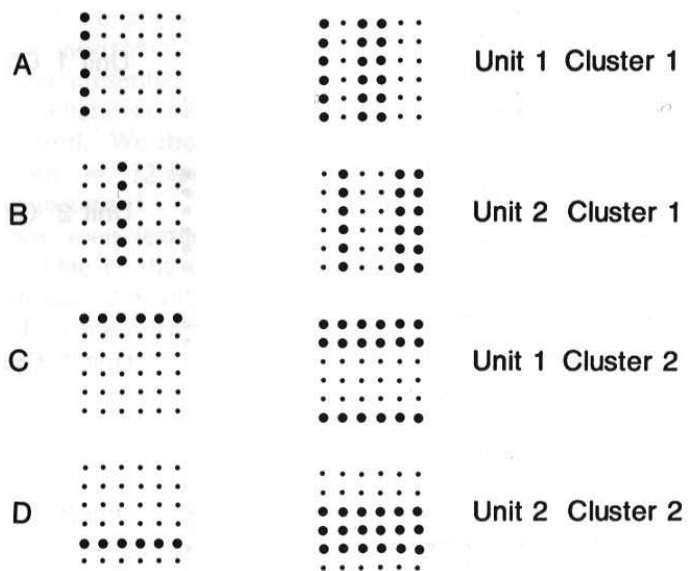
where g_l is the learning rate for the losing units, g_w is the learning rate for the winning unit, and where $g_l \ll g_w$. In our experiments we made



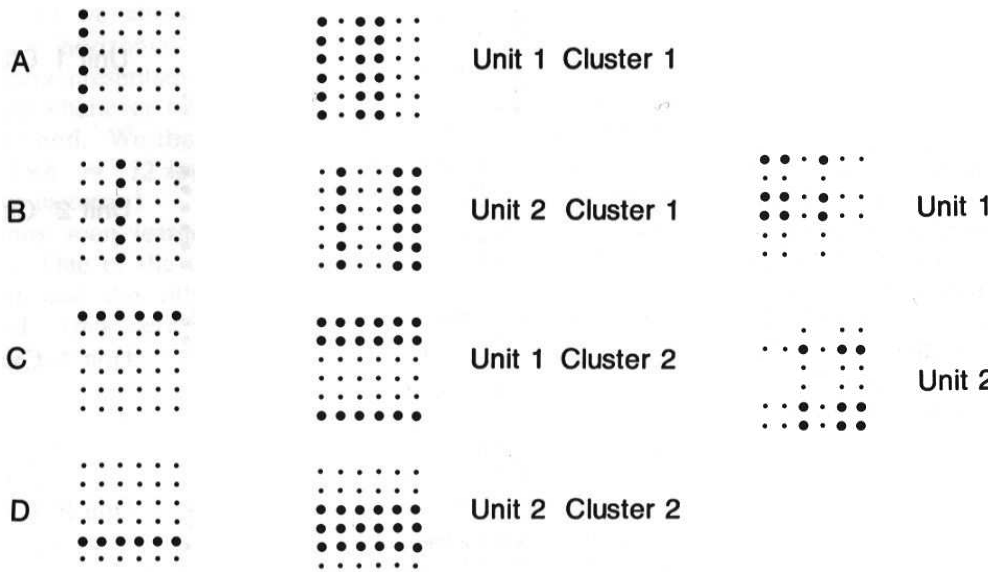
Horizontal/vertical bars



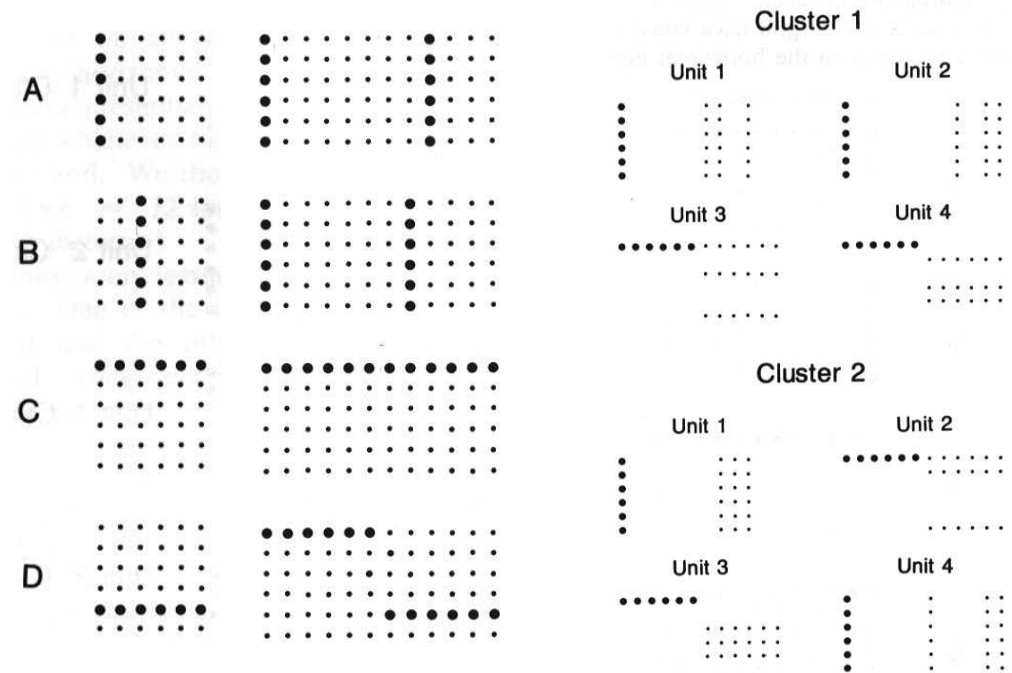
Horizontal/vertical bars



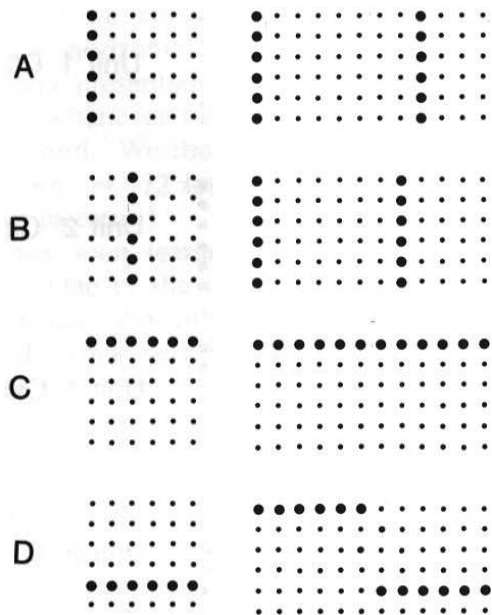
Horizontal/vertical bars



Horizontal/vertical bars: Teaching input



Horizontal/vertical bars: Teaching input



Self-Organizing Maps/Kohonen Networks

- Extension of competitive learning in which competing units are topographically organized (usually 2D)
- Neighbors of "winner" also update their weights (usually to a lesser extent), and thereby become more likely to respond to similar inputs
- Similarity of input space gets mapped onto (2D) topographic unit space

