Neural Networks

Associative Networks

Dr. Petr Musilek

Department of Electrical and Computer Engineering
University of Alberta

Fall 2019

Associative Networks

Hebbian Learning

Donald Hebb (1949)

As neuron *i* becomes more efficient in stimulating neuron *j* during training

- *i* sensitizes *j* to its stimulus
- Weight w_{ij} increases

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}} + \eta x_i o_j$$

Problems?

Modified Hebian learning (Grossberg)

Allows weights to decrease when not longer stimulated

$$\begin{aligned} w_{ij}^{\text{new}} &= w_{ij}^{\text{old}}(1-\alpha) &+ \quad \eta x_i o_j \\ \text{forgetting term} &&+ \quad \text{Hebbian term} \end{aligned}$$

$$\frac{w_{ij}^{\text{new}} - w_{ij}^{\text{old}}}{\Delta t} = \frac{\Delta w_{ij}}{\Delta t} \rightarrow \frac{dw_{ij}}{dt} = -\alpha w_{ij}^{\text{old}} + \eta x_i o_j$$

i.e. slow exponential weight decay with time constant α

Differential Hebian learning

Allows weights to decrease when the output decreases

$$w_{ij}^{\text{new}} = w_{ij}^{\text{old}}(1-\alpha) \quad + \quad \eta \frac{dx_i}{dt}\frac{do_j}{dt}$$

$$\text{forgetting term} \quad + \quad \text{differential term}$$

(rate of change used instead of value)

Note:

No target values used for training - this is an unsupervised learning.

Grossberg Learning

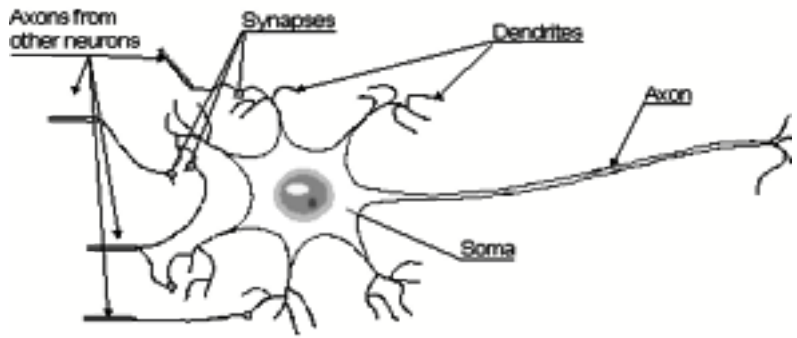
Attempt to mathematically explain psychological conditioning experiments

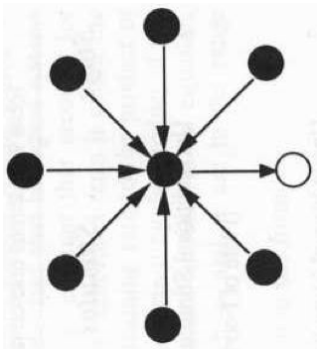
- Observational conditioning (monkeys)
- Operational conditioning (action/response)
- Classical conditioning (Pavlov)

Pavlov's experiment

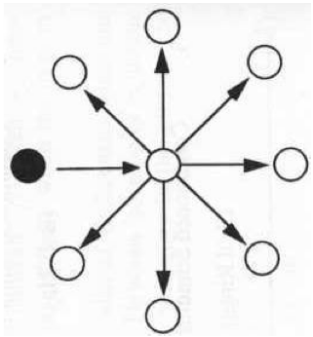
Stage	Stimulus	Response
1	Unconditional (food)	Unconditioned (dog salivates)
2	Unconditional (food) PLUS Conditioned (bell)	Conditioned (dog salivates)
3	Conditioned (bell)	Conditioned (dog salivates)

Concepts of instar and outstar





Instar



Outstar

Instar activity

1. The activity must grow when there is an external stimulus
2. It must rapidly decrease if it is no longer externally stimulated
3. It must respond to stimuli from other neurons in the network

Instar Learning rule

If inputs and weights are normalized, *tot* is largest when weights are identical to input values

- weights would be changed only if they are different from the inputs

$$\Delta w_i = \eta(x_i - w_i)$$

- incoming weights of neuron converge to input pattern (previous layer)
- unsupervised learning (no targets available)

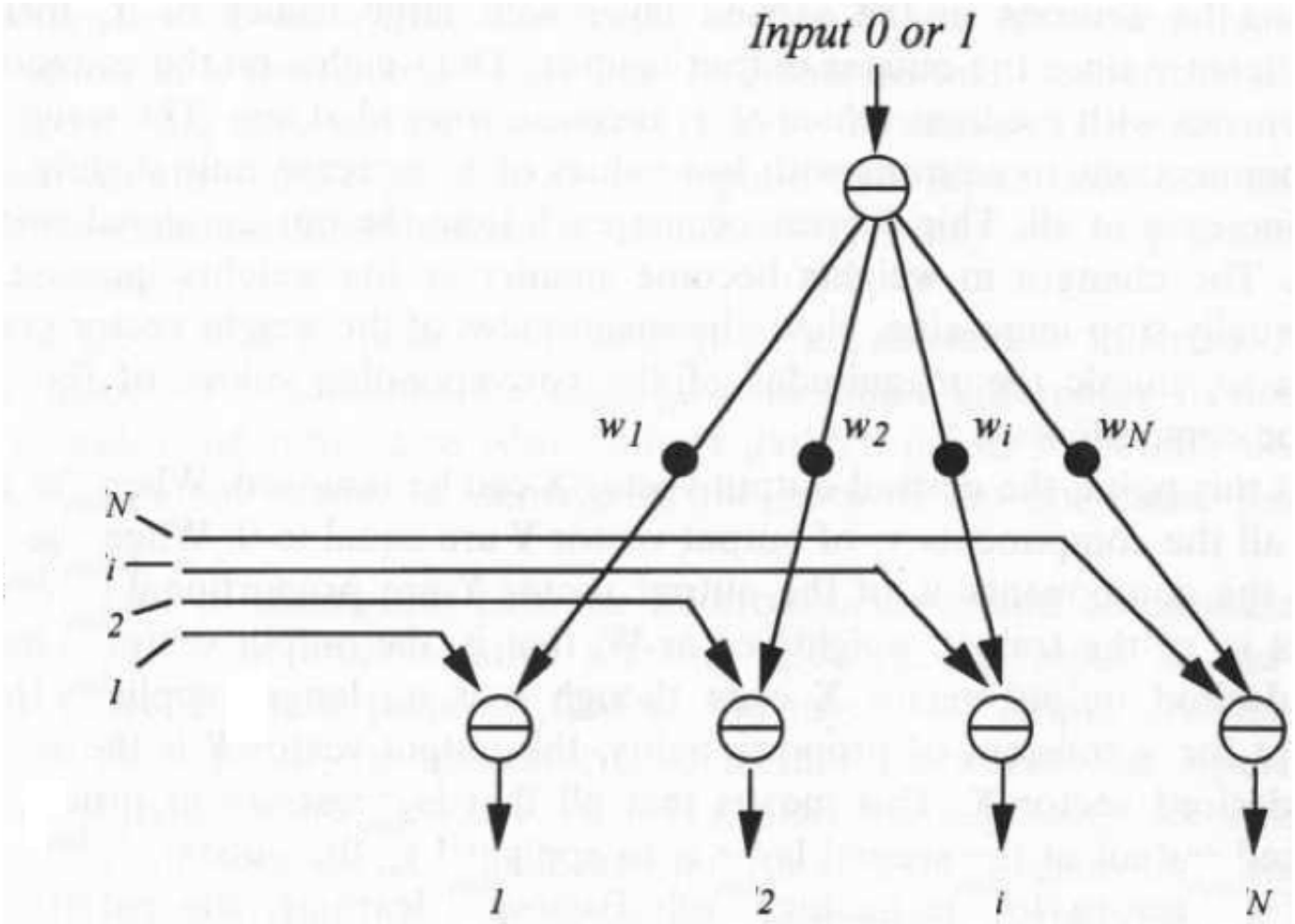
Outstar Learning rule

Weights connected to certain node should be equal to the desired outputs for the neurons they connect

$$\Delta w_{ij} = \eta(t_j - w_{ij})$$

- outgoing weights of neuron converge to output pattern (next layer)
- neuron learns to recall pattern when stimulated
- supervised learning

Outstar learning



Associative Memories (AM)

Systems that store information by associating each data item with one or more other stored data items, (usually) in distributed form.

- Content addressable (CAM)
- Robust
  - Noisy data
  - Incomplete data
  - Failed elements

Associative Memories ...

- Concept: object or pattern $\boldsymbol{x}$ (input) reminds the network of object or pattern $\boldsymbol{o}$ (output)
- Many biological neural nets are associative memories
- Heteroassociative Vs. autoassociative memories
  - If $\boldsymbol{x}$ and $\boldsymbol{o}$ are different, the system is called *heteroassociative* network
  - If $\boldsymbol{x}$ and $\boldsymbol{o}$ are the same, the system is called *autoassociative* nettwork

Associative Memories ...

- Unidirectional vs. bidirectional memories
  - Unidirectional: ***x*** reminds you of ***o***
  - Bidirectional: ***x*** reminds you of ***o*** and ***o*** reminds you of ***x***
- Recognizing new or incomplete patterns
  - Recognizing patterns that are similar to one of the patterns stored in memory (generalization)
  - Recognizing incomplete or noisy patterns whose complete (correct) forms were previously stored in memory

Bidirectional Associative Memory

BAM is a matrix representing a crossbar network with symmetric weights

- the network has two layers
- each neuron in a layer has one output from the outside, and
- inputs from all neurons of the other layer

BAM can store pattern pairs $[\boldsymbol{x}(k), \boldsymbol{o}(k)]$; , $k = 1, \ldots, n$ and recall

- $\boldsymbol{o}$ corresponding to particular $\boldsymbol{x}$, or
- $\boldsymbol{x}$ corresponding to particular $\boldsymbol{o}$
- i.e. it works in both directions $\rightarrow$ *bidirectional*

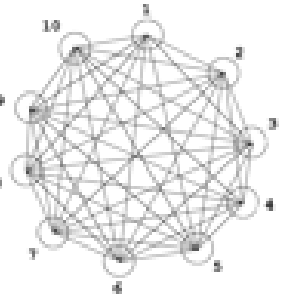
Setting of BAM weights

Weights are not obtained by training process but directly constructed from input–output pairs, e.g. binary to grey code

$$\begin{array}{lccccccccccccl}
\boldsymbol{x}(1): & (& 0 & 0 & 1 & 0 &) & \Longleftrightarrow & (& 0 & 0 & 1 & 1 &) & :\boldsymbol{o}(1) \\
\boldsymbol{x}(2): & (& 0 & 0 & 1 & 1 &) & \Longleftrightarrow & (& 0 & 0 & 0 & 0 &) & :\boldsymbol{o}(2) \\
\boldsymbol{x}(3): & (& 0 & 1 & 0 & 0 &) & \Longleftrightarrow & (& 0 & 1 & 1 & 0 &) & :\boldsymbol{o}(3)
\end{array}$$

- Before proceeding, the values must be converted to bipolar (0s become -1s while 1s remain)
- From each pair, a mapping matrix $\boldsymbol{M}_k = \boldsymbol{x}(k) \times \boldsymbol{o}(k)$ is constructed
- The matrices are then combined into a master matrix $\boldsymbol{M} = \boldsymbol{M}_1 + \boldsymbol{M}_2 + \cdots + \boldsymbol{M}_n$

Hopfield Network

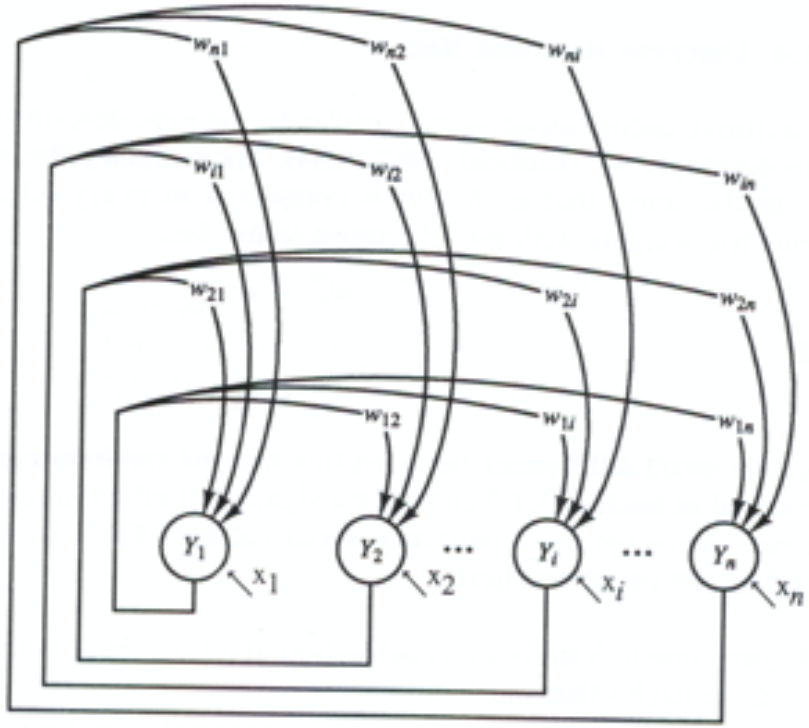


- A more advanced type of autoassociative memory
- Almost fully connected

- Architecture
  - symmetric weights

$$w_{ij} = w_{ji}$$

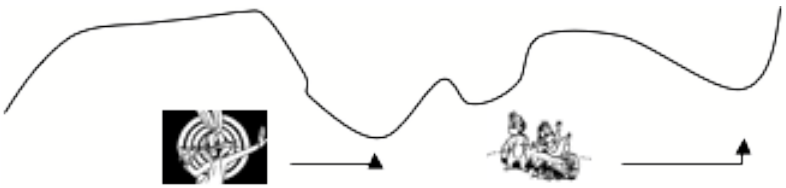
  - notice the “feedback” in the network structure
  - no feedback from a cell to itself $w_{ii} = 0$



Principle

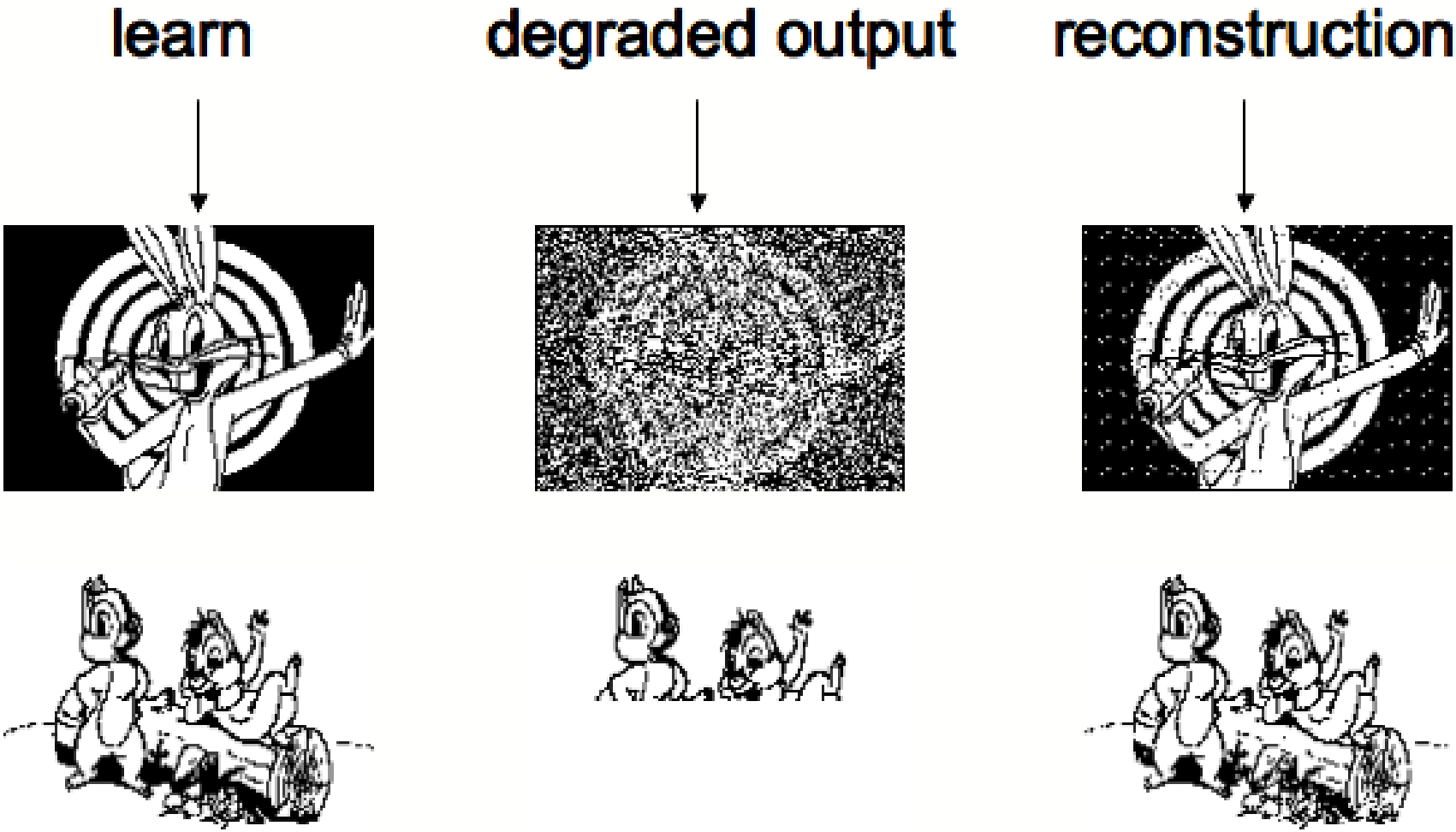
Based on the physical principle that every object seek a low energy static position.

During learning, the network uses input to define the low energy points.



When a similar to a learned position is encountered, some neurons will drag the other neurons to the static position.

Illustration



Operation

- Task: Store images with resolution 20×20 pixels (need a network with 400 nodes)
- Learning:
  1. Present image
  2. Apply Hebb rule: cells that fire together, wire together (increase weight between two nodes if both have same activity, otherwise decrease)
  3. Go back to 1
- Recall:
  1. Present incomplete pattern
  2. Pick random node, update
  3. Repeat 2 until settled

Hebbian rule for Hopfield network

Changes are proportional to the correlation between the firing (activity) of the pre- and post-synaptic neurons.

Technically:

- Consider
  $T = \{x(k)|x(k) = (x_1(k), \ldots, x_m(k)) \in \{-1, 1\}^m, k = 1, \ldots, n\}$
- Start with $w_{ji} = 0$ $(j = 1, \ldots, m;\ i = 1, \ldots, m)$
- For given training set T do

$$w_{ij} = \sum_{k=1}^{n} x_j(k) x_i(k), 1 \leq j \neq i \leq m$$

Active Mode of Hopfield Network

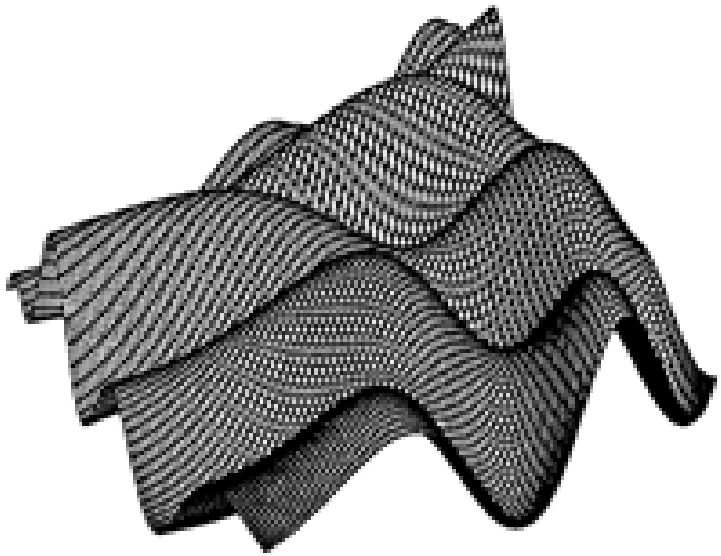
1. Set $o_i = x_i (i = 1, \ldots, m)$
2. Go through all neurons and at each time step select one neuron j to be updated according the following rule:
   - compute its internal potential $tot_j = \sum_{i=1}^{m} w_{ij} o_i$
   - set its new state $o_j = \begin{cases} 1 & \text{if } tot_j > 0, \\ o_i & \text{if } tot_j = 0, \\ -1 & \text{if } tot_j < 0. \end{cases}$
3. IF not stable configuration THEN go to step 2
   ELSE end - output of the net is determined by the current state of neurons.

Energy Function and Landscape

Energy function $E(o) = -\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} w_{ij} o_i o_j$

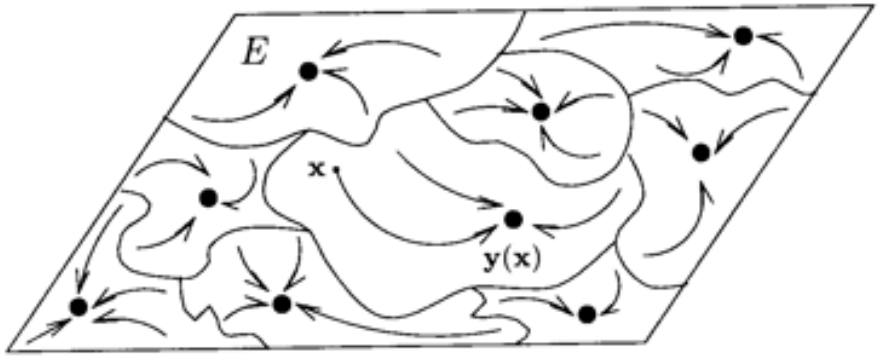
Energy Landscape

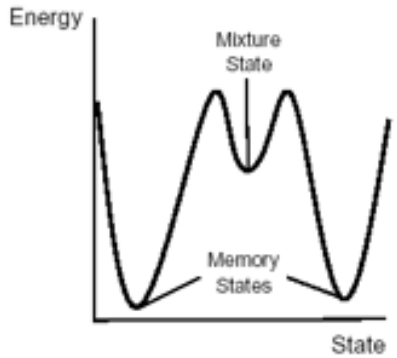
- high energy - unstable states
- low energy - more stable states
- energy always decreases (or remain constant) as the system evolves according to its dynamical rule



Attractors and Phantoms

- Local minima of the energy function represent stored examples - attractors
- Basins of attraction - catchment areas around each minimum
- False local optima - phantoms





Storage Capacity of Hopfield Network

- Capacity of the network - maximum number of patterns that can be stored without unacceptable errors.
- Empirical results
  - $n \leq 0.138m$ - training examples as local minima of $E(o)$
  - $n < 0.05m$ - training examples as global minima of $E(o)$, deeper minima than those corresponding to phantoms
- Example: 10 tr. examples, 200 neurons $\rightarrow$ 40000 weights

Example

Pattern recognition

- 8 examples, matrix 12×10 pixels → 120 neurons
- input pattern with 25% bits incorrect