When neural networks meet error correcting codes: towards new architectures for associative memories

Vincent Gripon
Joint work with Claude Berrou

Télécom Bretagne

April 7th, 2013

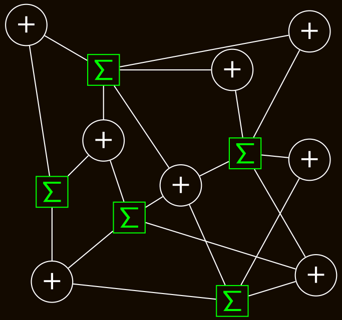
Plan

1. Biology and error correction codes
   - Starting point
   - From the neocortex to recurrent graphs : our three hypotheses
   - A few words about error correcting codes
2. A new architecture of associative memory
   - Storing
   - Retrieving
   - Performance
3. Developments
   - Blurred messages
   - Correlated sources
   - Learning sequences
4. Current and future work
   - Storing hierarchical messages
   - Combining associative memories and classifiers
   - Towards new computation models based on information

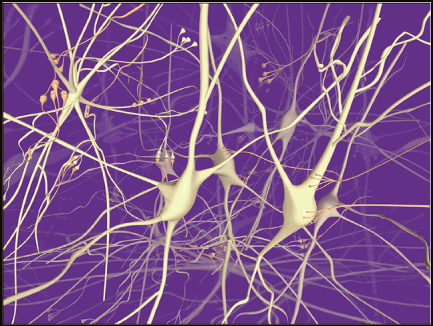
Plan

1. Biology and error correction codes
   - Starting point
   - From the neocortex to recurrent graphs : our three hypotheses
   - A few words about error correcting codes
2. A new architecture of associative memory
   - Storing
   - Retrieving
   - Performance
3. Developments
   - Blurred messages
   - Correlated sources
   - Learning sequences
4. Current and future work
   - Storing hierarchical messages
   - Combining associative memories and classifiers
   - Towards new computation models based on information

LDPC decoder

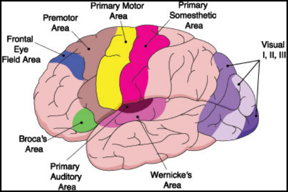


Neocortical "decoder"

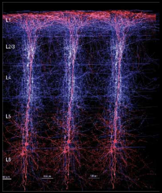


Both systems aim at retrieving a previously stored piece of information given part of its content.

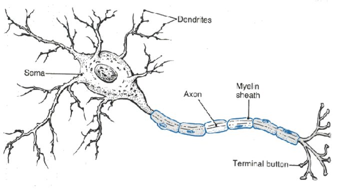
First hypothesis: the information scale



Macroscopic scale



Mesoscopic scale



Microscopic scale

Second hypothesis: redundancy

Illustration

02 29 00 12 77 12 77

02 29 00 1- 77 12 -7

Redundancy

- We lose approximately one neuron per second,
- But we remember our phone number,
- Mental information is robust,
- Therefore redundant.

Second hypothesis: redundancy

Illustration

02 29 00 12 77   12 77

02 29 00 1- 77   12 -7

Redundancy

- We lose approximately one neuron per second,
- But we remember our phone number,
- Mental information is robust,
- Therefore redundant.

Second hypothesis: redundancy

Illustration

02 29 00 12 77 12 77

02 29 00 1- 77 12 -7

Redundancy

- We lose approximately one neuron per second,
- But we remember our phone number,
- Mental information is robust,
- Therefore redundant.

Second hypothesis: redundancy

Illustration

02 29 00 12 77 12 77

02 29 00 1- 77 12 -7

Redundancy

- We lose approximately one neuron per second,
- But we remember our phone number,
- Mental information is robust,
- Therefore redundant.

Second hypothesis: redundancy

Illustration

02 29 00 12 77 12 77

02 29 00 1- 77 12 -7

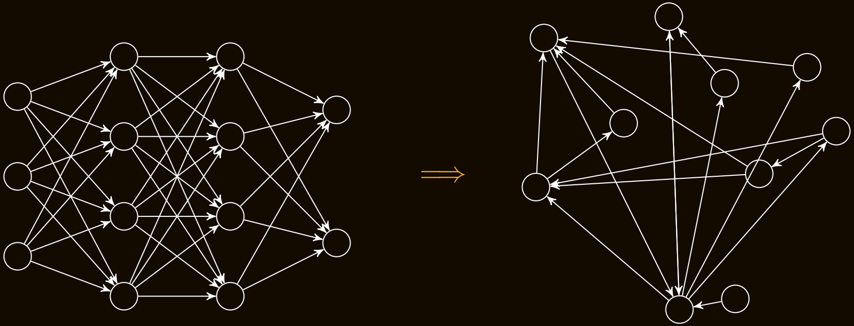
Redundancy

- We lose approximately one neuron per second,
- But we remember our phone number,
- Mental information is robust,
- Therefore redundant.

Third hypothesis: recurrent graph

The neocortex can be essentially regarded as a distributed recurrent graph.

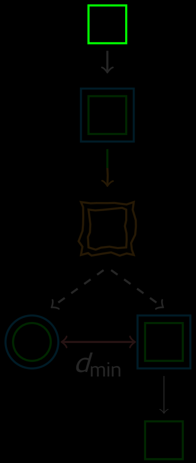
illustration



In one sentence

The neocortex is a recurrent, distributed graph of neocortical columns (fanals) that is able to store redundant pieces of information.

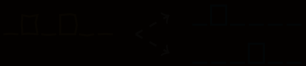
Error correcting codes



Example: the thrifty code

- Code containing only binary words with a single "1":

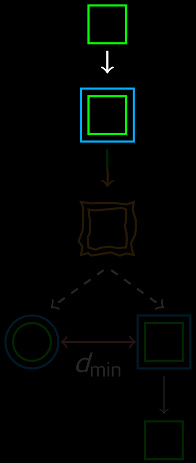
- Drawback: $d_{min} = 2$:



- But easy to decode and minimize the energy:

winner-take-all

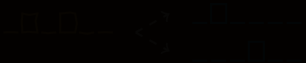
- These codes can be associated like the distributed codes. . .



Example: the thrifty code

- Code containing only binary words with a single "1":

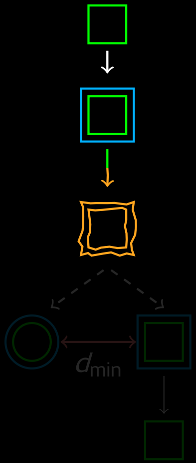
- Drawback: $d_{min} = 2$:



- But easy to decode and minimize the energy:

- These codes can be associated like the distributed codes. . .

Error correcting codes



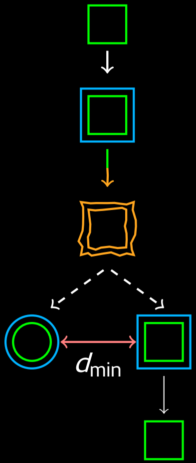
Example: the thrifty code

- Code containing only binary words with a single "1":
- Drawback: $d_{\min} = 2$:
- But easy to decode and minimize the energy:

winner-take-all

- These codes can be associated like the distributed codes. . .

Error correcting codes



Example: the thrifty code

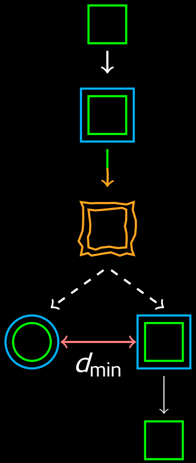
- Code containing only binary words with a single "1":

- Drawback: $d_{min} = 2$:

- But easy to decode and minimize the energy:

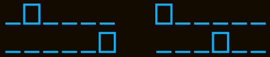
- These codes can be associated like the distributed codes...

Error correcting codes



Example: the thrifty code

- Code containing only binary words with a single "1":



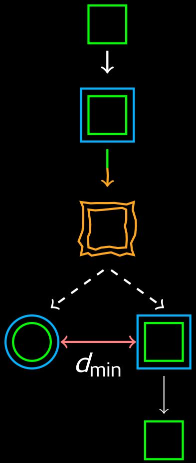
- Drawback: $d_{min} = 2$:

- But easy to decode and minimize the energy:

winner-take-all

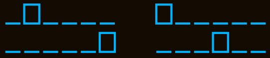
- These codes can be associated like the distributed codes...

Error correcting codes



Example: the thrifty code

- Code containing only binary words with a single "1":

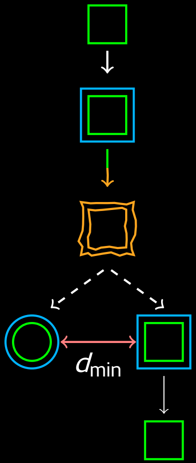


- Drawback: $d_{\min} = 2$:

- But easy to decode and minimize the energy:

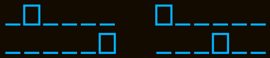
- These codes can be associated like the distributed codes...

Error correcting codes

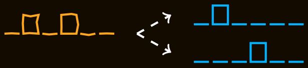


Example: the thrifty code

- Code containing only binary words with a single "1":



- Drawback: $d_{\min} = 2$:

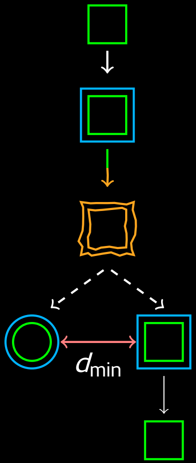


- But easy to decode and minimize the energy:

winner-take-all

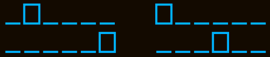
- These codes can be associated like the distributed codes...

Error correcting codes

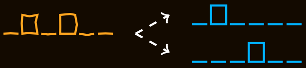


Example: the thrifty code

- Code containing only binary words with a single "1":



- Drawback: $d_{\min} = 2$:



- But easy to decode and minimize the energy:

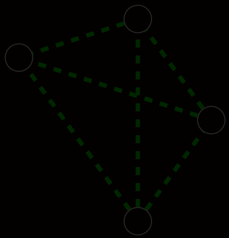
winner-take-all

- These codes can be associated like the distributed codes...

Clique

Set of nodes that are all connected one to another.

Example: codewords = 4 nodes cliques



2 distinct nodes
$\Rightarrow d_{\text{min}} = 6$ edges

Codes of cliques of size $c \ll n$

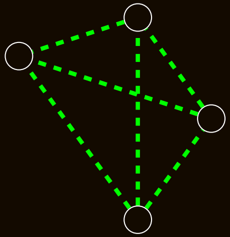
- $d_{\text{min}} = 2(c-1) \approx 2c$,
- $\Rightarrow F = rd_{\text{min}} \approx 2$,
- Cliques are codewords of a very interesting error correcting code...

Codes made of cliques of constant size

Clique

Set of nodes that are all connected one to another.

Example: codewords = 4 nodes cliques



Symbols = edges

2 distinct nodes $\Rightarrow d_{\min} = 6$ edges

Codes of cliques of size $c \ll n$

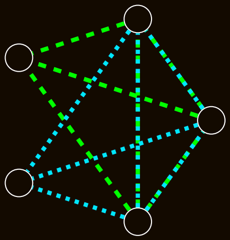
- $d_{\min} = 2(c-1) \approx 2c$,
- $\Rightarrow F = r d_{\min} \approx 2$,
- Cliques are codewords of a very interesting error correcting code. . . and they are nice!

Codes made of cliques of constant size

Clique

Set of nodes that are all connected one to another.

Example: codewords = 4 nodes cliques



Symbols = edges

2 distinct nodes
$\Rightarrow d_{\min} = 6$ edges

Codes of cliques of size $c \ll n$

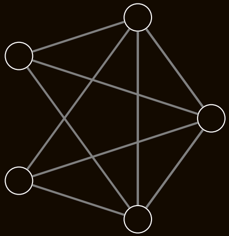
- $d_{\min} = 2(c-1) \approx 2c,$
- $\Rightarrow F = r d_{\min} \approx 2,$
- Cliques are codewords of a very interesting error correcting code. . . and they are free!

Codes made of cliques of constant size

Clique

Set of nodes that are all connected one to another.

Example: codewords = 4 nodes cliques



Symbols = edges

2 distinct nodes
$\Rightarrow d_{\min} = 6$ edges

Codes of cliques of size $c \ll n$

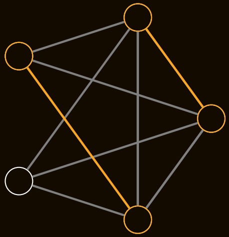
- $d_{\min} = 2(c-1) \approx 2c$, rate $r \approx \frac{c}{2}\binom{c}{2}^{-1}$
- $\Rightarrow F = r d_{\min} \approx 2$,
- Cliques are codewords of a very interesting error correcting code… and they are free!

Codes made of cliques of constant size

Clique

Set of nodes that are all connected one to another.

Example: codewords = 4 nodes cliques



Symbols = edges

2 distinct nodes
$\Rightarrow d_{min} = 6$ edges

Codes of cliques of size $c \ll n$

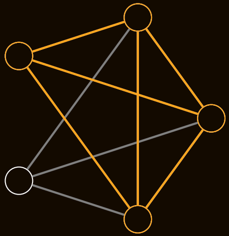
- $d_{min} = 2(c-1) \approx 2c$, rate $r \approx \frac{c}{2}\binom{c}{2}^{-1}$
- $\Rightarrow F = r d_{min} \approx 2$,
- Cliques are codewords of a very interesting error correcting code... and they are free!

Codes made of cliques of constant size

Clique

Set of nodes that are all connected one to another.

Example: codewords = 4 nodes cliques



Symbols = edges

2 distinct nodes
$\Rightarrow d_{\min} = 6$ edges

Codes of cliques of size $c \ll n$

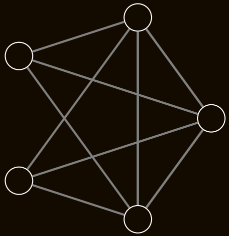
- $d_{\min} = 2(c-1) \approx 2c$, rate $r \approx \frac{c}{2}\binom{c}{2}^{-1}$
- $\Rightarrow F = rd_{\min} \approx 2$,
- Cliques are codewords of a very interesting error correcting code... and they are free!

Codes made of cliques of constant size

Clique

Set of nodes that are all connected one to another.

Example: codewords = 4 nodes cliques



Symbols = edges

2 distinct nodes
$\Rightarrow d_{\min} = 6$ edges

Codes of cliques of size $c \ll n$

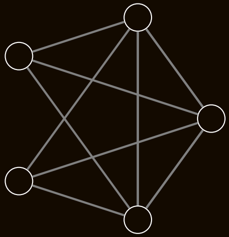
- $d_{\min} = 2(c-1) \approx 2c$, rate $r \approx \frac{c}{2}\binom{c}{2}^{-1}$
- $\Rightarrow F = rd_{\min} \approx 2$,
- Cliques are codewords of a very interesting error correcting code. . . and they are free!

Codes made of cliques of constant size

Clique

Set of nodes that are all connected one to another.

Example: codewords = 4 nodes cliques



Symbols = edges

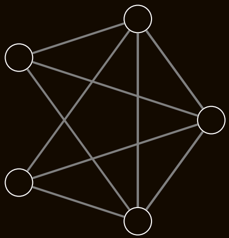
2 distinct nodes
$\Rightarrow d_{\min} = 6$ edges

Codes of cliques of size $c \ll n$

- $d_{\min} = 2(c-1) \approx 2c$, rate $r \approx \frac{c}{2}\binom{c}{2}^{-1}$
- $\Rightarrow F = rd_{\min} \approx 2$,
- Cliques are codewords of a very interesting error correcting code... and they are free!

Codes made of cliques of constant size

Example: codewords = 4 nodes cliques



Symbols = edges

2 distinct nodes

$\Rightarrow d_{\min} = 6$ edges

Clique

Set of nodes that are all connected one to another.

Codes of cliques of size $c \ll n$

- $d_{\min} = 2(c-1) \approx 2c$, rate $r \approx \frac{c}{2}\binom{c}{2}^{-1}$
- $\Rightarrow F = rd_{\min} \approx 2$,
- Cliques are codewords of a very interesting error correcting code. . . and they are free!

Plan

1. Biology and error correction codes
   - Starting point
   - From the neocortex to recurrent graphs : our three hypotheses
   - A few words about error correcting codes
2. A new architecture of associative memory
   - Storing
   - Retrieving
   - Performance
3. Developments
   - Blurred messages
   - Correlated sources
   - Learning sequences
4. Current and future work
   - Storing hierarchical messages
   - Combining associative memories and classifiers
   - Towards new computation models based on information

Associative memories and the Hopfield network

What is an associative memory?

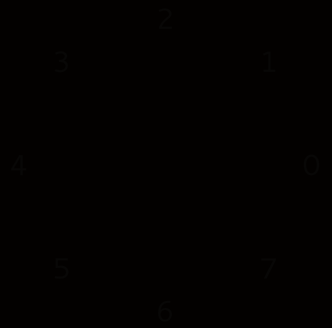
Two operations:

- Store a message,
- Retrieve a previously stored message from part of its content.

Our reference: the Hopfield network

Example:

- Store binary message -11-111-1-11
- Retrieve it from -11-111-1?1



Associative memories and the Hopfield network

What is an associative memory?

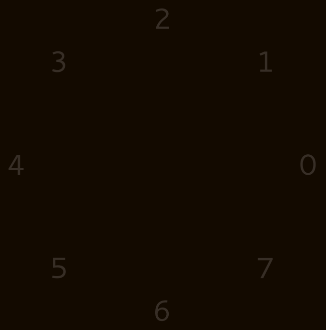
Two operations:

- Store a message,
- Retrieve a previously stored message from part of its content.

Our reference: the Hopfield network

Example:

- Store binary message -11-111-1-11
- Retrieve it from -11-111-1?1



Associative memories and the Hopfield network

What is an associative memory?

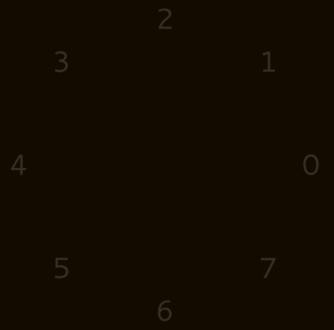
Two operations:

- Store a message,
- Retrieve a previously stored message from part of its content.

Our reference: the Hopfield network

Example:

- Store binary message -11-111-1-11
- Retrieve it from -11-111-1?1



Associative memories and the Hopfield network

What is an associative memory?

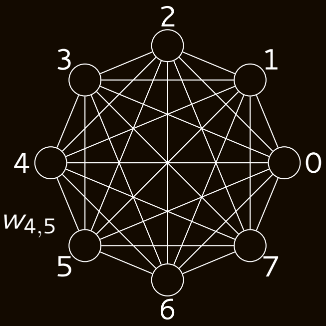
Two operations:

- Store a message,
- Retrieve a previously stored message from part of its content.

Our reference: the Hopfield network

Example:

- Store binary message -11-111-1-11
- Retrieve it from -11-111-1?1



What is an associative memory?

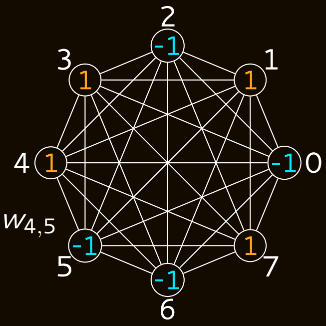
Two operations:

- Store a message,
- Retrieve a previously stored message from part of its content.

Our reference: the Hopfield network

Example:

- Store binary message -11-111-1-11
- Retrieve it from -11-111-1?1



Associative memories and the Hopfield network

What is an associative memory?

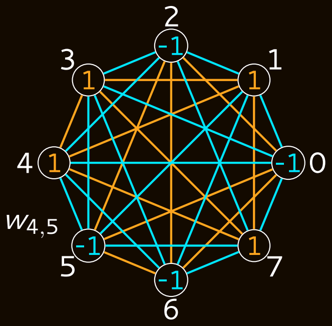
Two operations:

- Store a message,
- Retrieve a previously stored message from part of its content.

Our reference: the Hopfield network

Example:

- Store binary message -11-111-1-11
- Retrieve it from -11-111-1?1



Associative memories and the Hopfield network

What is an associative memory?

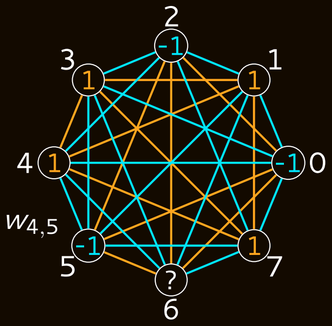
Two operations:

- Store a message,
- Retrieve a previously stored message from part of its content.

Our reference: the Hopfield network

Example:

- Store binary message -11-111-1-11
- Retrieve it from -11-111-1?1



Associative memories and the Hopfield network

What is an associative memory?

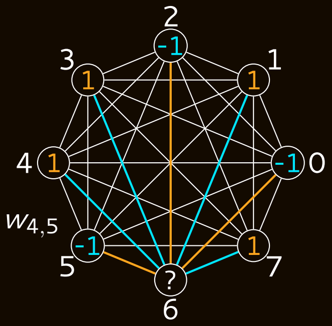
Two operations:

- Store a message,
- Retrieve a previously stored message from part of its content.

Our reference: the Hopfield network

Example:

- Store binary message -11-111-1-11
- Retrieve it from -11-111-1?1



Associative memories and the Hopfield network

What is an associative memory?

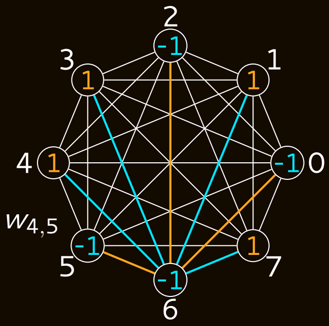
Two operations:

- Store a message,
- Retrieve a previously stored message from part of its content.

Our reference: the Hopfield network

Example:

- Store binary message -11-111-1-11
- Retrieve it from -11-111-1-11



Performance and bounds

Hopfield networks (n neurons $\longleftrightarrow$)

- Diversity : $M = \frac{n}{2log(n)}$, $\leftrightarrow$
- Capacity : $\frac{n^2}{2log(n)}$,
- Efficiency $\approx \frac{1}{log(n)log_2(M+1)}$.

Example with $n = 790$: 

Performance and bounds

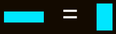
Hopfield networks (n neurons $\longleftrightarrow$)

- Diversity : $M = \frac{n}{2log(n)}$, $\leftrightarrow$
- Capacity : $\frac{n^2}{2log(n)}$, =
- Efficiency $\approx \frac{1}{log(n)log_2(M+1)}$.

Example with $n = 790$:

Performance and bounds

Hopfield networks (n neurons $\longleftrightarrow$)

- Diversity : $M = \frac{n}{2log(n)}$, $\leftrightarrow$
- Capacity : $\frac{n^2}{2log(n)}$, 
- Efficiency $\approx \frac{1}{log(n)log_2(M+1)}$. 

Example with $n = 790$: 

Performance and bounds

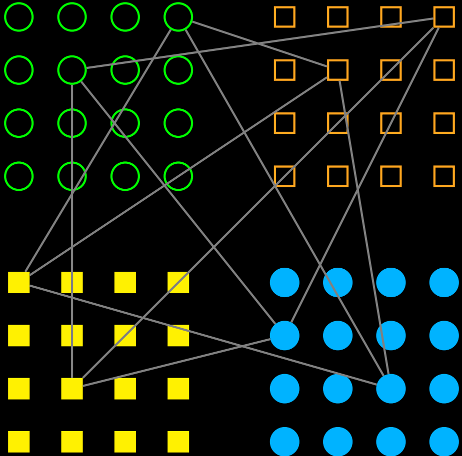
Hopfield networks (n neurons $\longleftrightarrow$)

- Diversity : $M = \frac{n}{2log(n)}$, $\leftrightarrow$
- Capacity : $\frac{n^2}{2log(n)}$, ▬ = ▮
- Efficiency $\approx \frac{1}{log(n)log_2(M+1)}$.

Example with $n = 790$: 

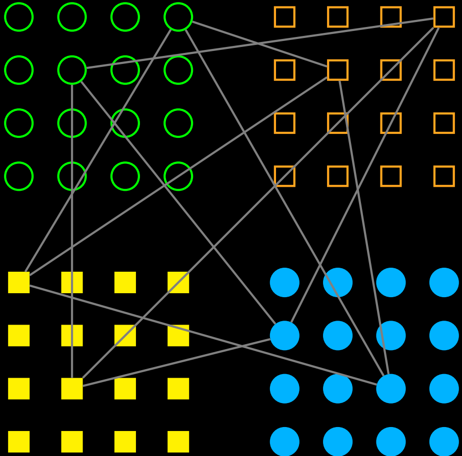
Our model: storing

- Example: $c = 4$ clusters made of $l = 16$ fanals each,
- $\underbrace{8}_{j_1 \text{ in } c_1}\underbrace{3}_{j_2 \text{ in } c_2}\underbrace{2}_{j_3 \text{ in } c_3}\underbrace{9}_{j_4 \text{ in } c_4}$,



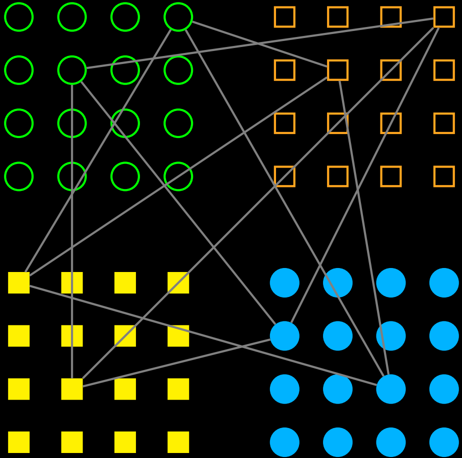
Our model: storing

- Example: $c = 4$ clusters made of $l = 16$ fanals each,
- $\underbrace{8}_{j_1 \text{ in } c_1} \underbrace{3}_{j_2 \text{ in } c_2} \underbrace{2}_{j_3 \text{ in } c_3} \underbrace{9}_{j_4 \text{ in } c_4}$,



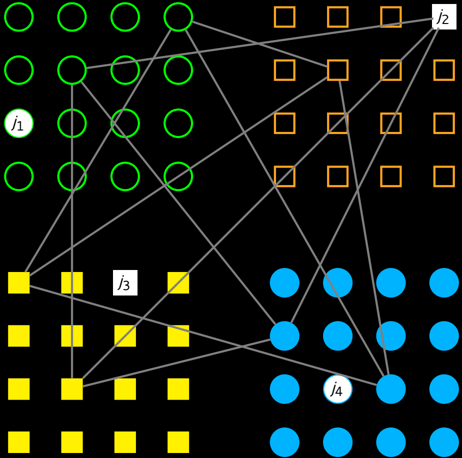
Our model: storing

- Example: $c = 4$ clusters made of $l = 16$ fanals each,
- $\underbrace{8}_{j_1 \text{ in } c_1} \underbrace{3}_{j_2 \text{ in } c_2} \underbrace{2}_{j_3 \text{ in } c_3} \underbrace{9}_{j_4 \text{ in } c_4}$,



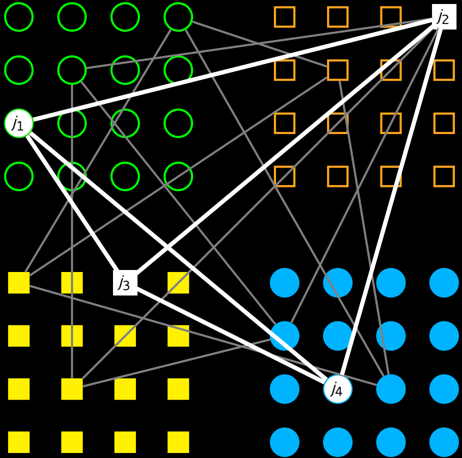
Our model: storing

- Example: $c = 4$ clusters made of $l = 16$ fanals each,
- $\underbrace{8}_{j_1 \text{ in } c_1} \underbrace{3}_{j_2 \text{ in } c_2} \underbrace{2}_{j_3 \text{ in } c_3} \underbrace{9}_{j_4 \text{ in } c_4}$,

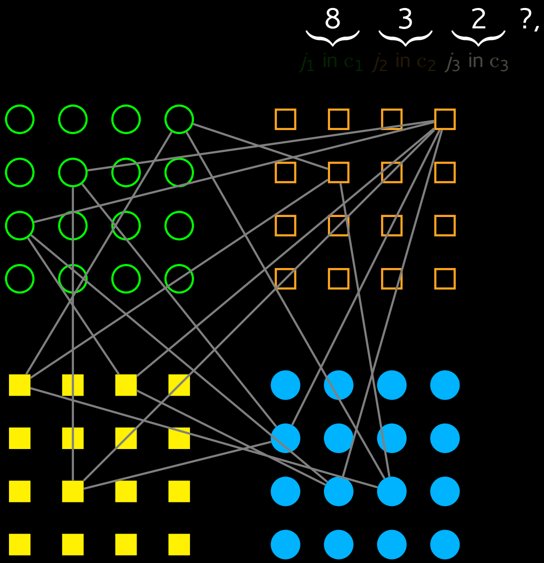


Our model: storing

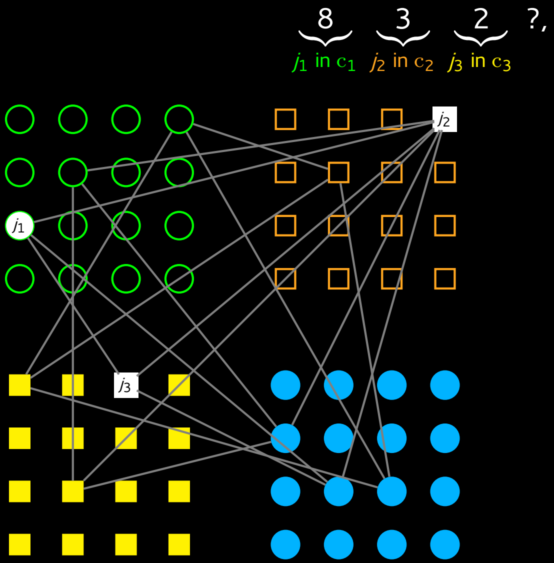
- Example: $c = 4$ clusters made of $l = 16$ fanals each,
- $\underbrace{8}_{j_1 \text{ in } c_1} \underbrace{3}_{j_2 \text{ in } c_2} \underbrace{2}_{j_3 \text{ in } c_3} \underbrace{9}_{j_4 \text{ in } c_4}$,



Our model: retrieving



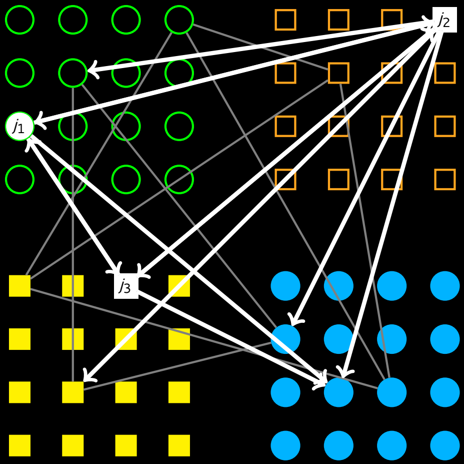
- Local connection,
- Global decoding: sum,
- Local decoding: winner-take-all,
- Possibly iterate the two decodings.



- Local connection,
- Global decoding: sum,
- Local decoding: winner-take-all,
- Possibly iterate the two decodings.

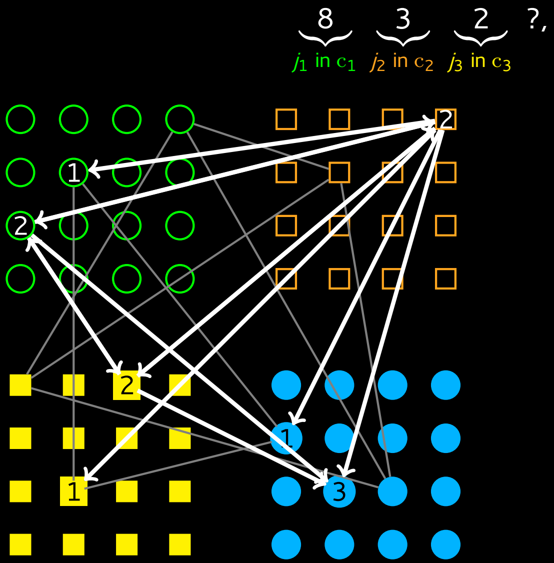
Our model: retrieving

$\underbrace{8}_{j_1 \text{ in } c_1} \; \underbrace{3}_{j_2 \text{ in } c_2} \; \underbrace{2}_{j_3 \text{ in } c_3} \; ?,$



- Local connection,
- Global decoding: sum,
- Local decoding: winner-take-all,
- Possibly iterate the two decodings.

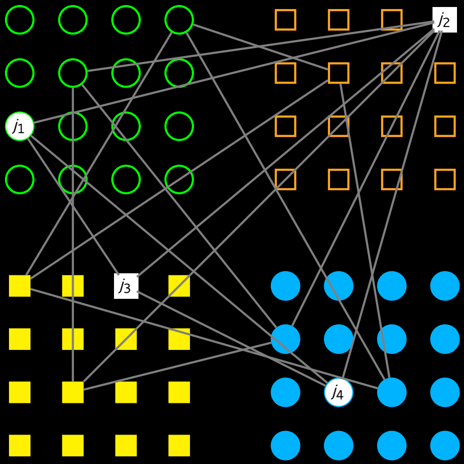
Our model: retrieving



- Local connection,
- Global decoding: sum,
- Local decoding: winner-take-all,
- Possibly iterate the two decodings.

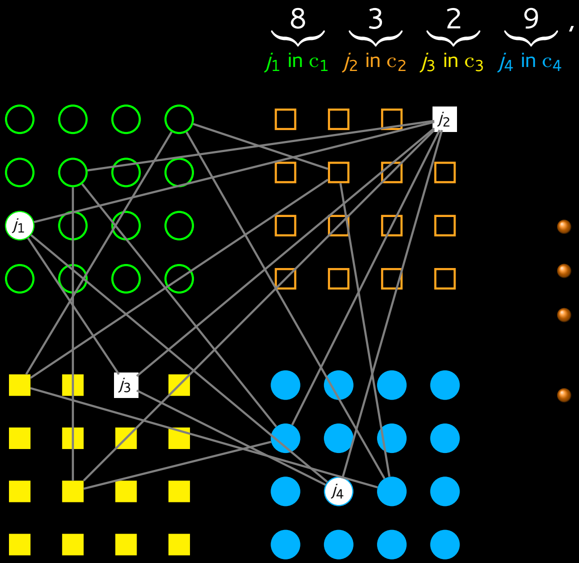
Our model: retrieving

$$\underbrace{8}_{j_1 \text{ in } c_1}\ \underbrace{3}_{j_2 \text{ in } c_2}\ \underbrace{2}_{j_3 \text{ in } c_3}\ \underbrace{9}_{j_4 \text{ in } c_4},$$



- Local connection,
- Global decoding: sum,
- Local decoding: winner-take-all,
- Possibly iterate the two decodings.

Our model: retrieving



- Local connection,
- Global decoding: sum,
- Local decoding: winner-take-all,
- Possibly iterate the two decodings.

Density

A binary model of long term memory

- Density d is the ratio of the number of used connections to the total number of possible ones,
- If messages are i.i.d.: $d \approx 1 - \left(1 - \frac{1}{l^2}\right)^M$.

Curves

Remarks

- $d = 1$: no more distinction between stored and not stored messages,
- $d = f(l, M)$, not depending on c,
- $d \approx \frac{M}{l^2}$, for $M \ll l^2$.

Density

A binary model of long term memory

- Density d is the ratio of the number of used connections to the total number of possible ones,
- If messages are i.i.d.: $d \approx 1 - \left(1 - \frac{1}{l^2}\right)^M$.

Curves

Remarks

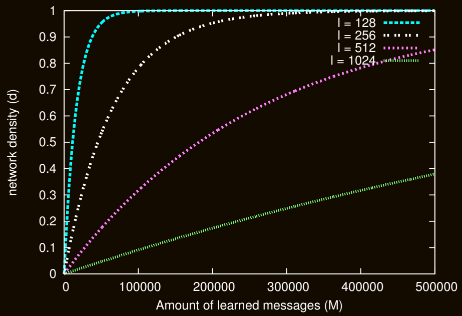
- $d = 1$: no more distinction between stored and not stored messages,
- $d = f(l, M)$, not depending on c,
- $d \approx \frac{M}{l^2}$, for $M \ll l^2$.

Density

A binary model of long term memory

- Density d is the ratio of the number of used connections to the total number of possible ones,
- If messages are i.i.d.: $d \approx 1 - \left(1 - \frac{1}{l^2}\right)^M$.

Curves



Remarks

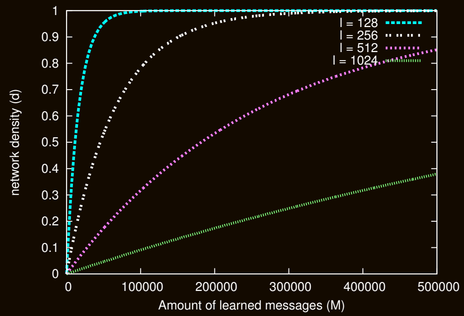
- $d = 1$: no more distinction between stored and not stored messages,
- $d = f(l, M)$, not depending on c,
- $d \approx \frac{M}{l^2}$, for $M \ll l^2$.

Density

A binary model of long term memory

- Density d is the ratio of the number of used connections to the total number of possible ones,
- If messages are i.i.d.: $d \approx 1 - \left(1 - \frac{1}{l^2}\right)^M$.

Curves



Remarks

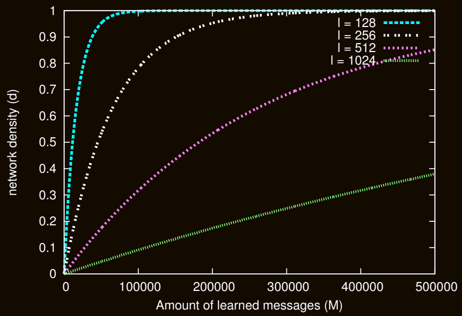
- $d = 1$: no more distinction between stored and not stored messages,
- $d = f(l, M)$, not depending on c,
- $d \approx \frac{M}{l^2}$, for $M \ll l^2$.

Density

A binary model of long term memory

- Density d is the ratio of the number of used connections to the total number of possible ones,
- If messages are i.i.d.: $d \approx 1 - \left(1 - \frac{1}{l^2}\right)^M$.

Curves



Remarks

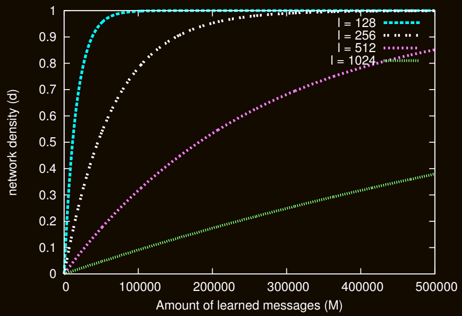
- $d = 1$: no more distinction between stored and not stored messages,
- $d = f(l, M)$, not depending on c,
- $d \approx \frac{M}{l^2}$, for $M \ll l^2$.

Density

A binary model of long term memory

- Density d is the ratio of the number of used connections to the total number of possible ones,
- If messages are i.i.d.: $d \approx 1 - \left(1 - \frac{1}{l^2}\right)^M$.

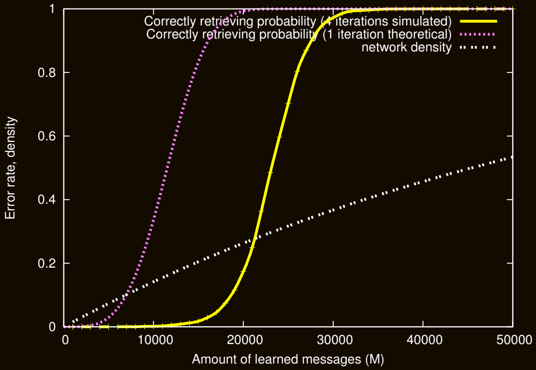
Curves



Remarks

- $d = 1$: no more distinction between stored and not stored messages,
- $d = f(l, M)$, not depending on c,
- $d \approx \frac{M}{l^2}$, for $M \ll l^2$.

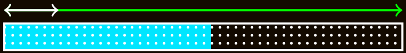
As an associative memory



$c = 8$ clusters of $l = 256$ fanals each ($\sim$ messages of 64 bits),
Error probability when retrieving messages half erased.

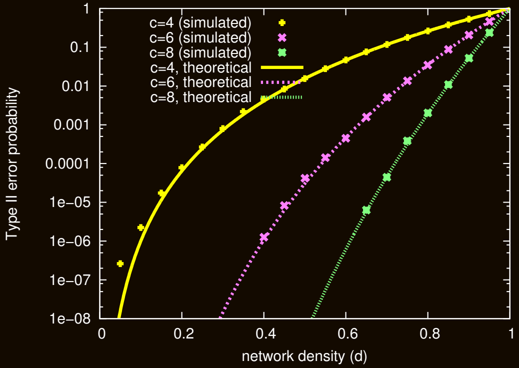
Hopfield network ($n = 790$)

Our network



Performance

Set implementation



Second kind error rate for various sizes of clusters c and for $l = 512$ fanals per cluster.

Hopfield network ($n = 740$)

Our network

Plan

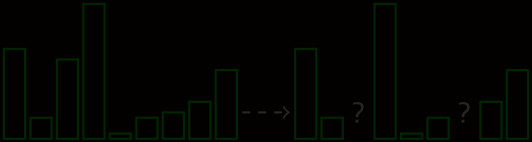
1. Biology and error correction codes
   - Starting point
   - From the neocortex to recurrent graphs : our three hypotheses
   - A few words about error correcting codes
2. A new architecture of associative memory
   - Storing
   - Retrieving
   - Performance
3. Developments
   - Blurred messages
   - Correlated sources
   - Learning sequences
4. Current and future work
   - Storing hierarchical messages
   - Combining associative memories and classifiers
   - Towards new computation models based on information

Blurred messages

Limitation

Partial messages must contain perfect information.

Noise model



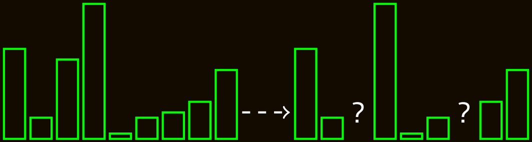
Soft decoding

Blurred messages

Limitation

Partial messages must contain perfect information.

Noise model



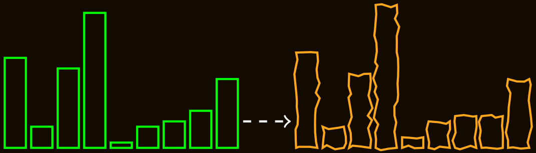
Soft decoding

Blurred messages

Limitation

Partial messages must contain perfect information.

Noise model



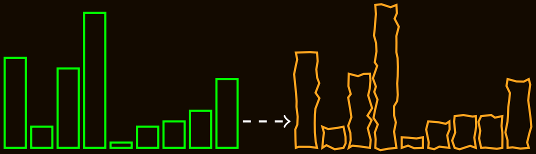
Soft decoding

Blurred messages

Limitation

Partial messages must contain perfect information.

Noise model



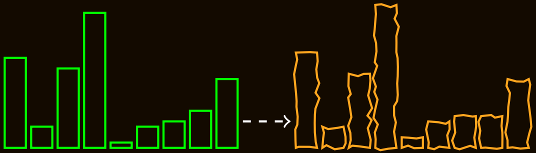
Soft decoding

Blurred messages

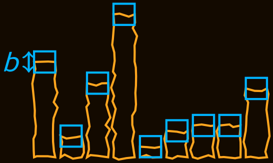
Limitation

Partial messages must contain perfect information.

Noise model



Soft decoding

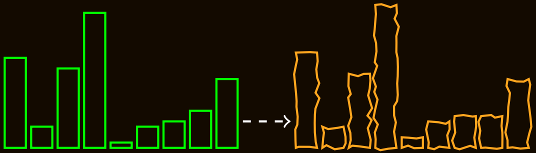


Blurred messages

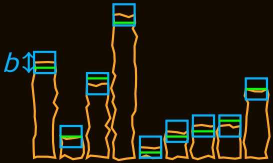
Limitation

Partial messages must contain perfect information.

Noise model

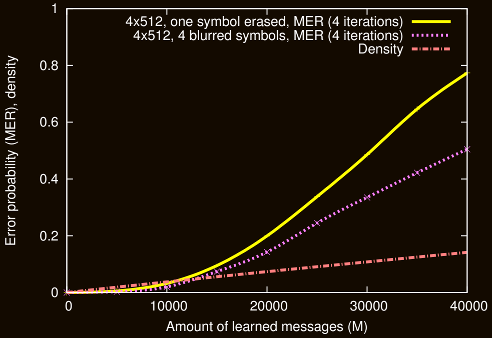


Soft decoding



Performance

Simulations



Comparison of performance when messages are partially erased and when they are blurred ($b = 5$).

Correlated messages

Limitation

With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy

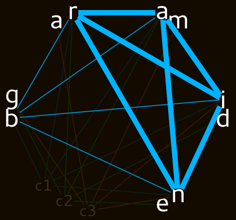
Correlated messages

Limitation

With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy

brain

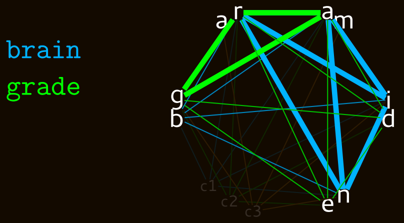


Correlated messages

Limitation

With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy



Correlated messages

Limitation

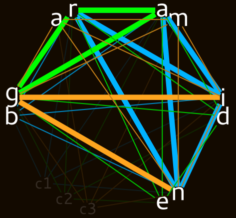
With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy

brain

grade

gamin



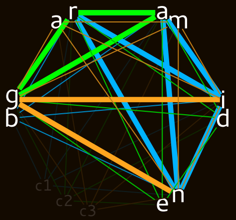
Correlated messages

Limitation

With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy

brain
grade
gamin
grain



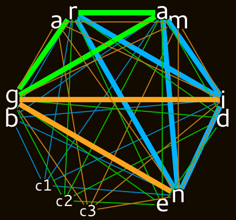
Correlated messages

Limitation

With correlations grows the number of Type II errors.

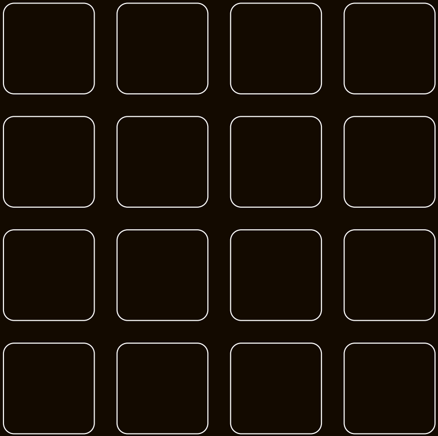
Fighting correlation by adding random redundancy

brain +c1
grade +c2
gamin +c3
grain +c?



Global winner-take-all

Illustration



Idea

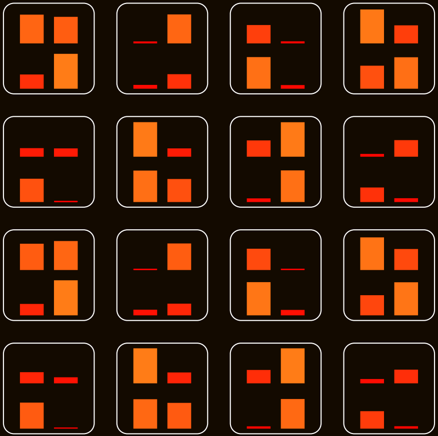
- After global message passing...
- After local maximum selections...
- Global maximum selection.

Interests

- Diversity $\propto c^2$,
- Stored messages length may vary.

Global winner-take-all

Illustration



Idea

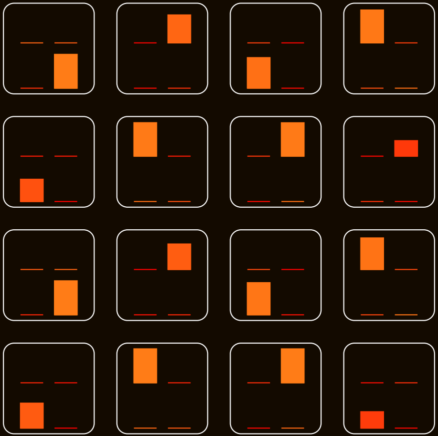
- After global message passing...
- After local maximum selections...
- Global maximum selection.

Interests

- Diversity $\propto c^2$,
- Stored messages length may vary.

Global winner-take-all

Illustration



Idea

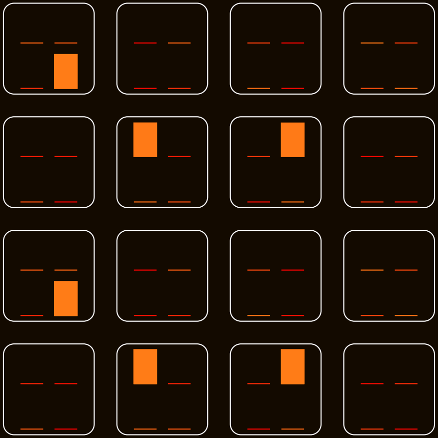
- After global message passing...
- After local maximum selections...
- Global maximum selection.

Interests

- Diversity $\propto c^2$,
- Stored messages length may vary.

Global winner-take-all

Illustration



Idea

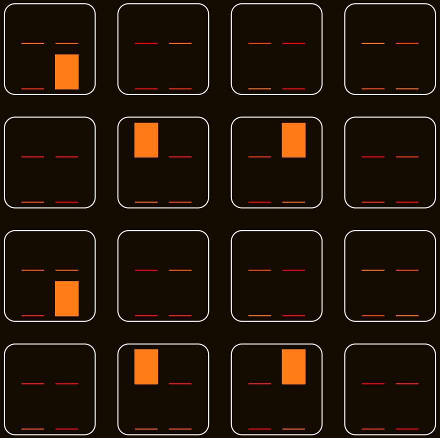
- After global message passing...
- After local maximum selections...
- Global maximum selection.

Interests

- Diversity $\propto c^2$,
- Stored messages length may vary.

Global winner-take-all

Illustration



Idea

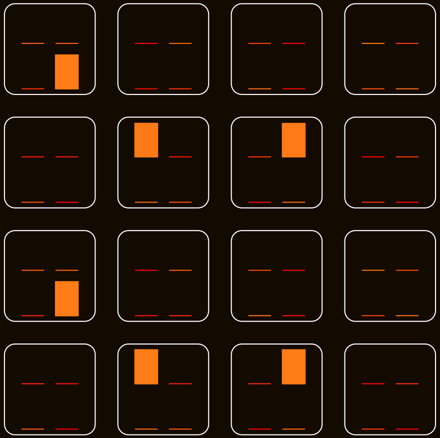
- After global message passing...
- After local maximum selections...
- Global maximum selection.

Interests

- Diversity $\propto c^2$,
- Stored messages length may vary.

Global winner-take-all

Illustration



Idea

- After global message passing...
- After local maximum selections...
- Global maximum selection.

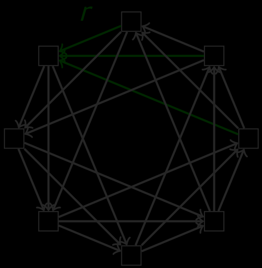
Interests

- Diversity $\propto c^2$,
- Stored messages length may vary.

Tournament chains and unidirectional connections

Problem

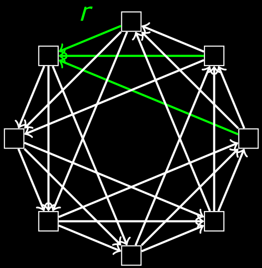
Bidirectional connections and full inter-connectivity.



Tournament chains and unidirectional connections

Problem

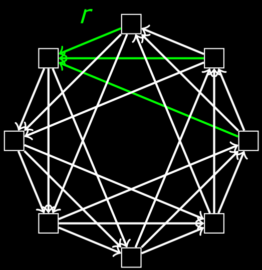
Bidirectional connections and full inter-connectivity.

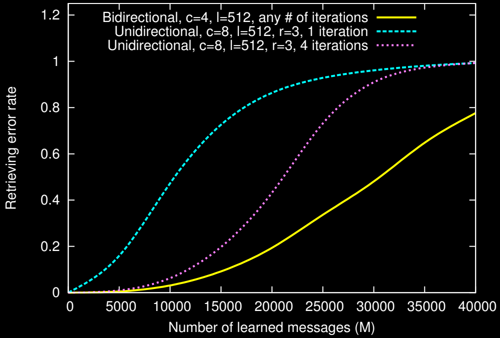


Tournament chains and unidirectional connections

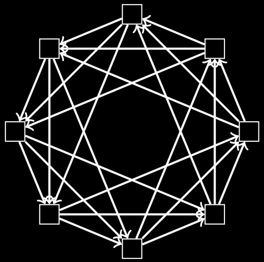
Problem

Bidirectional connections and full inter-connectivity.





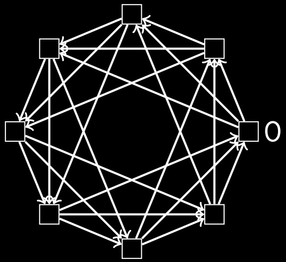
Learning arbitrarily long sequences



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

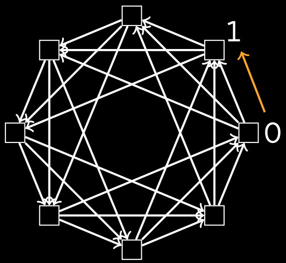
Learning arbitrarily long sequences



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

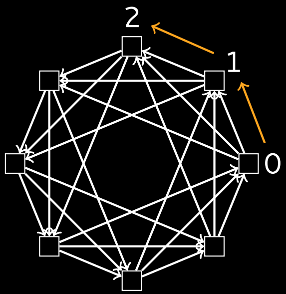
Learning arbitrarily long sequences



Performance

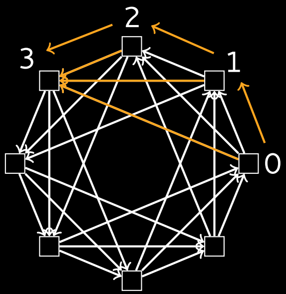
- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

Learning arbitrarily long sequences



Performance

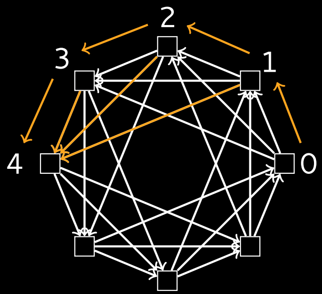
- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

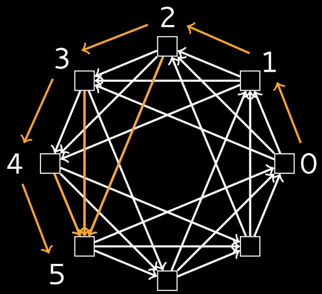
Learning arbitrarily long sequences



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

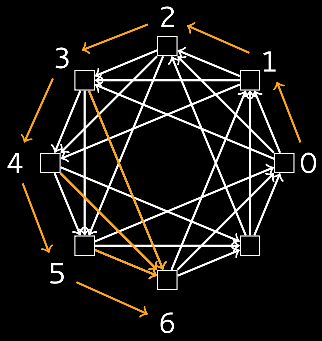
Learning arbitrarily long sequences



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

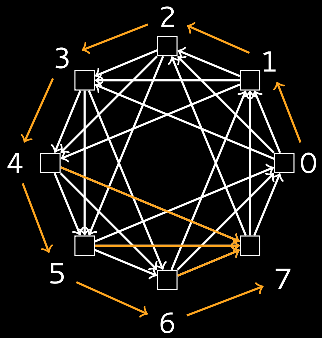
Learning arbitrarily long sequences



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

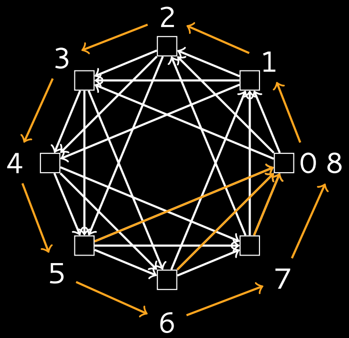
Learning arbitrarily long sequences



Performance

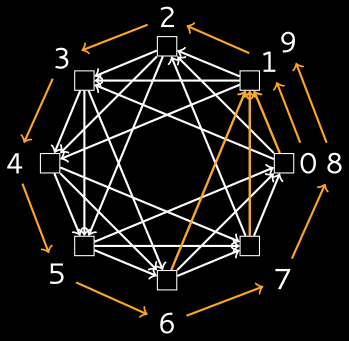
- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

Learning arbitrarily long sequences



Performance

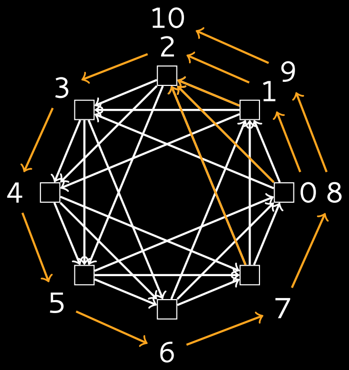
- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

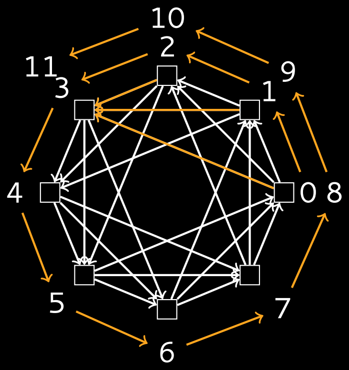
Learning arbitrarily long sequences



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

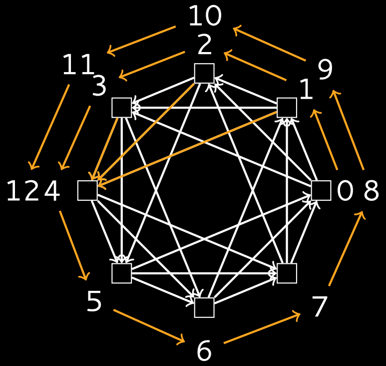
Learning arbitrarily long sequences



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

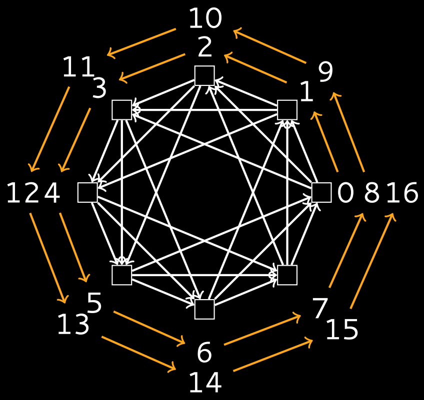
Learning arbitrarily long sequences



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

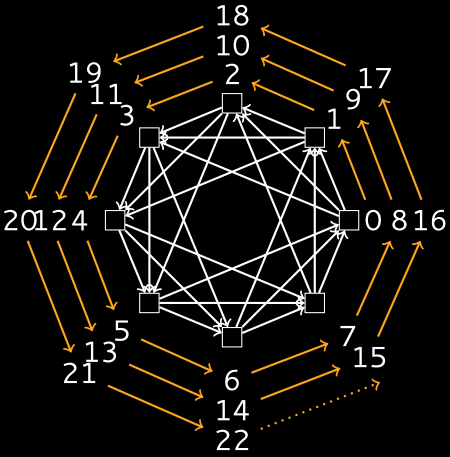
Learning arbitrarily long sequences



Performance

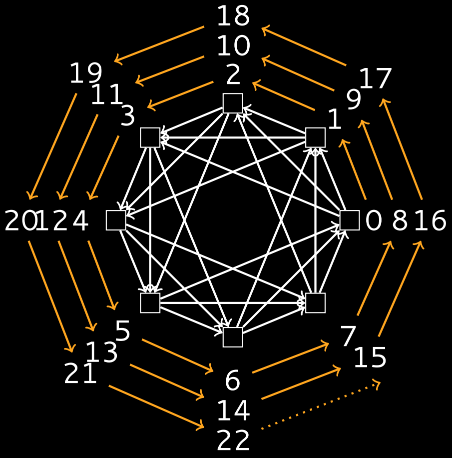
- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

Learning arbitrarily long sequences



Performance

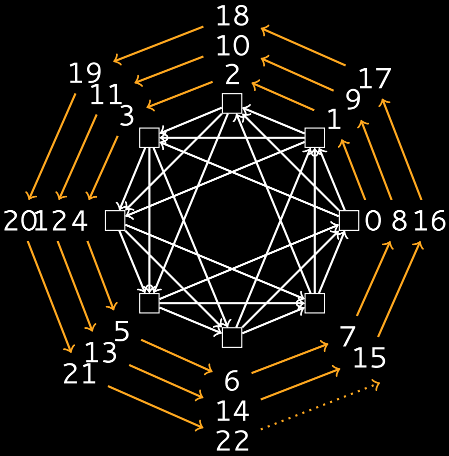
- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

Learning arbitrarily long sequences



Performance

- $c = 50$ clusters,
- $l = 256$ fanals/cluster,
- $L = 1000$ symbols in messages,
- $m = 1823$ learned messages,
- $P_e \leq 0.01$.

Plan

1. Biology and error correction codes
   - Starting point
   - From the neocortex to recurrent graphs : our three hypotheses
   - A few words about error correcting codes
2. A new architecture of associative memory
   - Storing
   - Retrieving
   - Performance
3. Developments
   - Blurred messages
   - Correlated sources
   - Learning sequences
4. Current and future work
   - Storing hierarchical messages
   - Combining associative memories and classifiers
   - Towards new computation models based on information

Storing hierarchical messages

Question

How to store pieces of hierarchical information (e.g. sentences of words of letters) into associative memories?

Idea

- Provide networks with a third dimension,
- Connect layers using subsampling,
- Use time in decoding.

Possibility to store hierarchical messages with unchanged efficiency.

Storing hierarchical messages

Question

How to store pieces of hierarchical information (e.g. sentences of words of letters) into associative memories?

Idea

- Provide networks with a third dimension,
- Connect layers using subsampling,
- Use time in decoding.

Possibility to store hierarchical messages with unchanged efficiency.

Storing hierarchical messages

Question

How to store pieces of hierarchical information (e.g. sentences of words of letters) into associative memories?

Idea

- Provide networks with a third dimension,
- Connect layers using subsampling,
- Use time in decoding.

Possibility to store hierarchical messages with unchanged efficiency.

Storing hierarchical messages

Question

How to store pieces of hierarchical information (e.g. sentences of words of letters) into associative memories?

Idea

- Provide networks with a third dimension,
- Connect layers using subsampling,
- Use time in decoding.

Possibility to store hierarchical messages with unchanged efficiency.

Storing hierarchical messages

Question

How to store pieces of hierarchical information (e.g. sentences of words of letters) into associative memories?

Idea

- Provide networks with a third dimension,
- Connect layers using subsampling,
- Use time in decoding.

Possibility to store hierarchical messages with unchanged efficiency.

Combining associative memories and classifiers

Idea

- Combine the discriminative abilities of associative memories with that, associative, of classifiers,
- Use classifiers to project input space into one where Hamming distance is meaningful.

Combining associative memories and classifiers

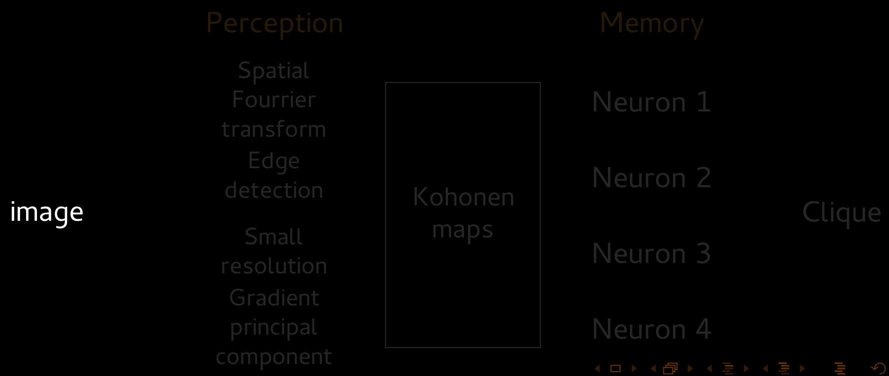
Idea

- Combine the discriminative abilities of associative memories with that, associative, of classifiers,
- Use classifiers to project input space into one where Hamming distance is meaningful.

Combining associative memories and classifiers

Idea

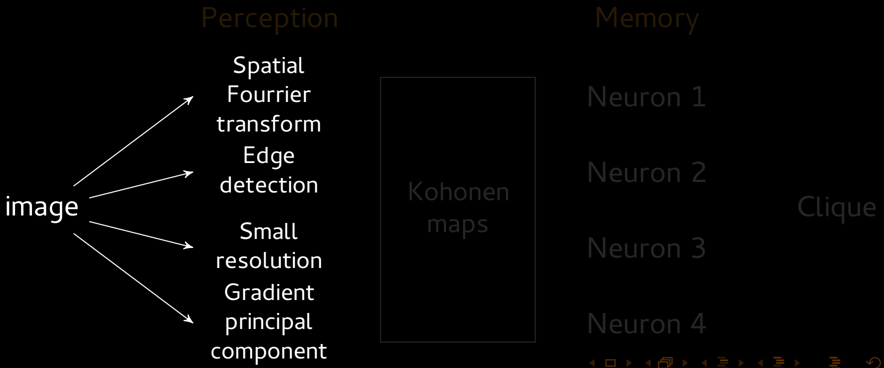
- Combine the discriminative abilities of associative memories with that, associative, of classifiers,
- Use classifiers to project input space into one where Hamming distance is meaningful.



Combining associative memories and classifiers

Idea

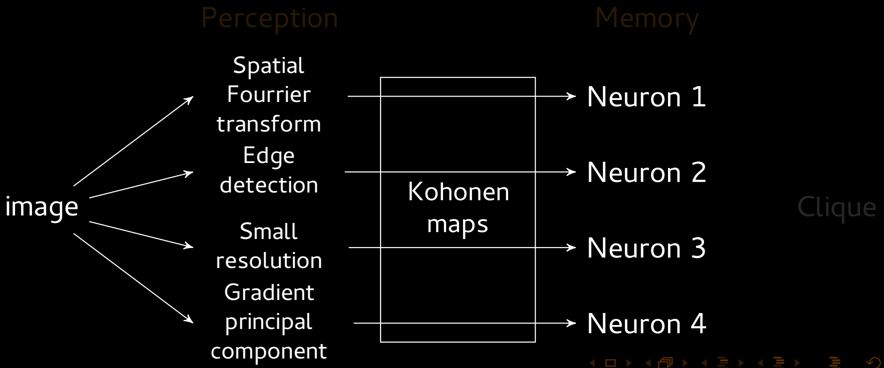
- Combine the discriminative abilities of associative memories with that, associative, of classifiers,
- Use classifiers to project input space into one where Hamming distance is meaningful.



Combining associative memories and classifiers

Idea

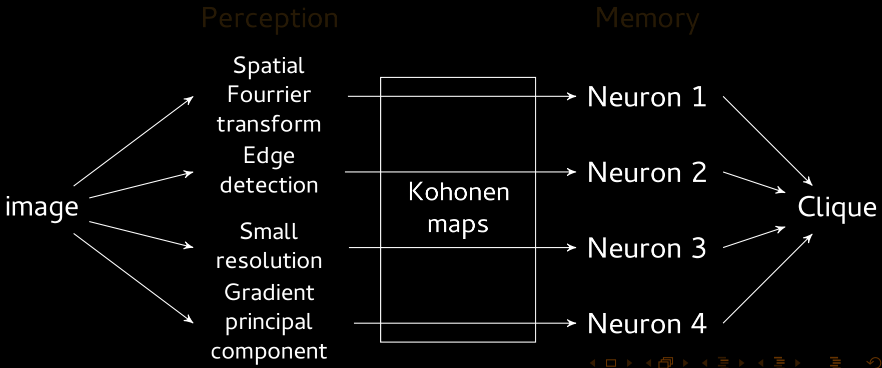
- Combine the discriminative abilities of associative memories with that, associative, of classifiers,
- Use classifiers to project input space into one where Hamming distance is meaningful.



Combining associative memories and classifiers

Idea

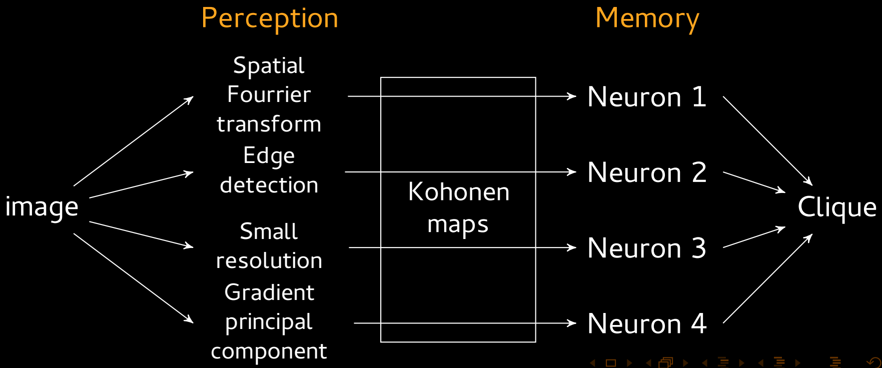
- Combine the discriminative abilities of associative memories with that, associative, of classifiers,
- Use classifiers to project input space into one where Hamming distance is meaningful.



Combining associative memories and classifiers

Idea

- Combine the discriminative abilities of associative memories with that, associative, of classifiers,
- Use classifiers to project input space into one where Hamming distance is meaningful.



Towards new computation models based on information

Idea

- Finite input/output problem $\equiv$ finite set of couples $(input, output)$,
- Store them into an associative memory to obtain a solution.

Perspectives

- Extend to nonfinite input/output problems,
- Organize multiple associative memories jointly,
- Construct meta associative memories to build up solutions to more complex problems.

Towards new computation models based on information

Idea

- Finite input/output problem $\equiv$ finite set of couples $(input, output)$,
- Store them into an associative memory to obtain a solution.

Perspectives

- Extend to nonfinite input/output problems,
- Organize multiple associative memories jointly,
- Construct meta associative memories to build up solutions to more complex problems.

Towards new computation models based on information

Idea

- Finite input/output problem $\equiv$ finite set of couples $(\mathit{input}, \mathit{output})$,
- Store them into an associative memory to obtain a solution.

Perspectives

- Extend to nonfinite input/output problems,
- Organize multiple associative memories jointly,
- Construct meta associative memories to build up solutions to more complex problems.

Towards new computation models based on information

Idea

- Finite input/output problem $\equiv$ finite set of couples $(input, output)$,
- Store them into an associative memory to obtain a solution.

Perspectives

- Extend to nonfinite input/output problems,
- Organize multiple associative memories jointly,
- Construct meta associative memories to build up solutions to more complex problems.

Towards new computation models based on information

Idea

- Finite input/output problem $\equiv$ finite set of couples $(input, output)$,
- Store them into an associative memory to obtain a solution.

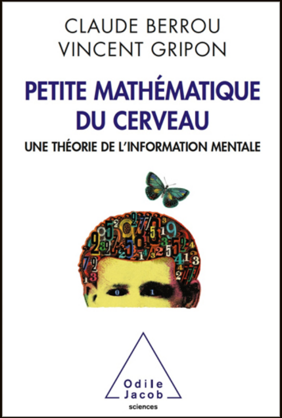
Perspectives

- Extend to nonfinite input/output problems,
- Organize multiple associative memories jointly,
- Construct meta associative memories to build up solutions to more complex problems.

Questions

I am at your disposal if you have any question.

A bit of reading



To learn more

Visit:
http://www.vincent-gripon.com/?p1=100

Acknowledgements

Collaborators:

- C. Berrou,
- A. Abudib, X. Jiang,
- G. Coppin, D. Pastor, E. Sedgh Gooya.