

How to improve associative memories using neural coding

Vincent Gripon

Joint work with:

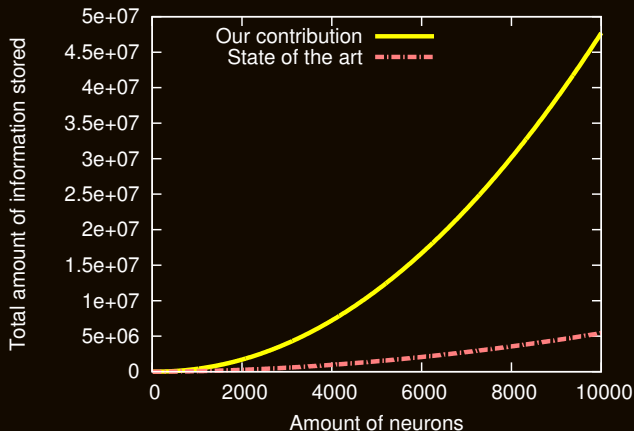
Claude Berrou, Behrooz Kamary Aliabadi and Xiaoran Jiang

Télécom Bretagne, Lab-STICC

2012, Sept. 13th

In one figure...

Storing messages in recurrent neural networks



1 Associative memories and error correcting codes

- Associative memory
- Error correcting codes

2 Sparse networks, principles and performance

- Storing
- Retrieving
- Performance

3 Developments

- Blurred messages
- Correlated sources
- Sparse messages

4 Conclusion

1 Associative memories and error correcting codes

- Associative memory
- Error correcting codes

2 Sparse networks, principles and performance

- Storing
- Retrieving
- Performance

3 Developments

- Blurred messages
- Correlated sources
- Sparse messages

4 Conclusion

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 $(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$
- Retrieve it from $(1, 1, 1, 1, 1, 1, 1, 1, 1, 0)$

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?

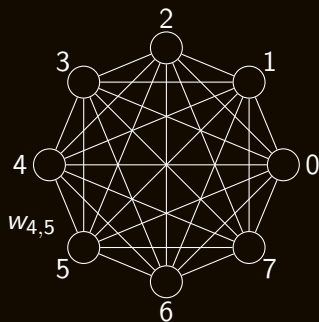
Two operations:

- **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?

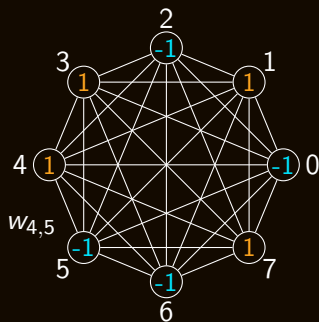
Two operations:

- **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**



Associative memories and the Hopfield network

What is an associative memory?

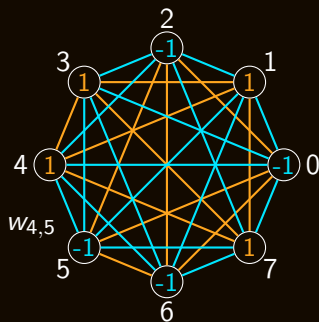
Two operations:

- **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?

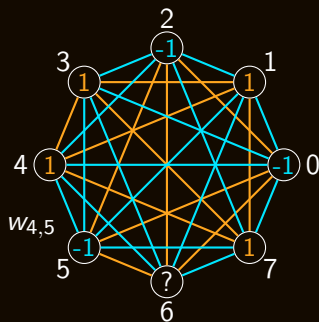
Two operations:

- **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?

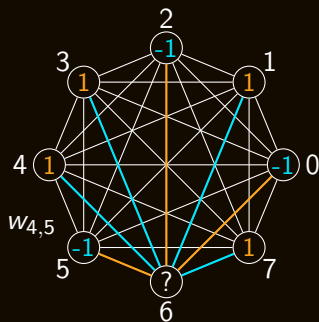
Two operations:

- **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?

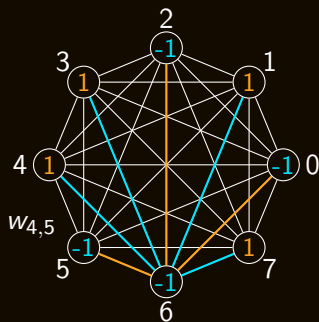
Two operations:

- **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 $-11-111-1-11$




Hopfield networks (n neurons \longleftrightarrow)

- **Diversity** : $M = \frac{n}{2\log(n)}$, \longleftrightarrow
- **Capacity** : $\frac{n^2}{2\log(n)}$, $\text{---} = \blacksquare$
- **Efficiency** $\approx \frac{1}{\log(n)\log_2(M+1)}$, $\begin{matrix} \blacksquare & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \end{matrix}$


Example with $n = 790$:

Hopfield networks (n neurons \longleftrightarrow)

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


Example with $n = 790$: 


Hopfield networks (n neurons \longleftrightarrow)

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

Example with $n = 790$: 

Hopfield networks (n neurons \longleftrightarrow)

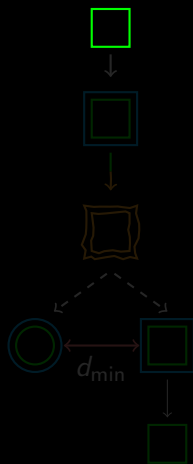
- **Diversity** : $M = \frac{n}{2\log(n)}$, \longleftrightarrow
- **Capacity** : $\frac{n^2}{2\log(n)}$, $\blacksquare = \blacksquare$
- **Efficiency** $\approx \frac{1}{\log(n)\log_2(M+1)}$. 

Example with $n = 790$: 

Error correcting codes

Example: the thrifty code

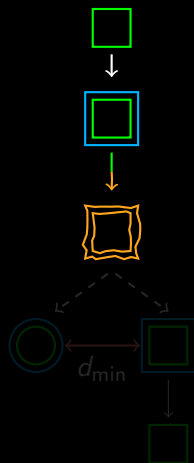
- Code containing only binary words with a single "1"
- Encoder: $0 \rightarrow 0000$, $1 \rightarrow 0001$
- Decoder: $0000 \rightarrow 0$, $0001 \rightarrow 1$
- Drawback: $d_{\min} = 2$
- Example: $0001 \rightarrow 0010$ (bit flip)
- But: $0010 \rightarrow 0000$ and $0010 \rightarrow 0001$
- Solution: $0 \rightarrow 00000$, $1 \rightarrow 00001$
- Encoder: $0 \rightarrow 00000$, $1 \rightarrow 00001$
- Decoder: $00000 \rightarrow 0$, $00001 \rightarrow 1$
- Example: $00001 \rightarrow 00010$ (bit flip)
- But: $00010 \rightarrow 00000$ and $00010 \rightarrow 00001$



Error correcting codes

Example: the thrifty code

- Code containing only binary words with a single "1"
- Example: $\{00000000, 00000001, 00000010, 00000011, 00000100, 00000101, 00000110, 00000111, 00001000, 00001001, 00001010, 00001011, 00001100, 00001101, 00001110, 00001111, 00010000, 00010001, 00010010, 00010011, 00010100, 00010101, 00010110, 00010111, 00011000, 00011001, 00011010, 00011011, 00011100, 00011101, 00011110, 00011111, 00100000, 00100001, 00100010, 00100011, 00100100, 00100101, 00100110, 00100111, 00101000, 00101001, 00101010, 00101011, 00101100, 00101101, 00101110, 00101111, 00110000, 00110001, 00110010, 00110011, 00110100, 00110101, 00110110, 00110111, 00111000, 00111001, 00111010, 00111011, 00111100, 00111101, 00111110, 00111111, 01000000, 01000001, 01000010, 01000011, 01000100, 01000101, 01000110, 01000111, 01001000, 01001001, 01001010, 01001011, 01001100, 01001101, 01001110, 01001111, 01010000, 01010001, 01010010, 01010011, 01010100, 01010101, 01010110, 01010111, 01011000, 01011001, 01011010, 01011011, 01011100, 01011101, 01011110, 01011111, 01100000, 01100001, 01100010, 01100011, 01100100, 01100101, 01100110, 01100111, 01101000, 01101001, 01101010, 01101011, 01101100, 01101101, 01101110, 01101111, 01110000, 01110001, 01110010, 01110011, 01110100, 01110101, 01110110, 01110111, 01111000, 01111001, 01111010, 01111011, 01111100, 01111101, 01111110, 01111111, 10000000, 10000001, 10000010, 10000011, 10000100, 10000101, 10000110, 10000111, 10001000, 10001001, 10001010, 10001011, 10001100, 10001101, 10001110, 10001111, 10010000, 10010001, 10010010, 10010011, 10010100, 10010101, 10010110, 10010111, 10011000, 10011001, 10011010, 10011011, 10011100, 10011101, 10011110, 10011111, 10100000, 10100001, 10100010, 10100011, 10100100, 10100101, 10100110, 10100111, 10101000, 10101001, 10101010, 10101011, 10101100, 10101101, 10101110, 10101111, 10110000, 10110001, 10110010, 10110011, 10110100, 10110101, 10110110, 10110111, 10111000, 10111001, 10111010, 10111011, 10111100, 10111101, 10111110, 10111111, 11000000, 11000001, 11000010, 11000011, 11000100, 11000101, 11000110, 11000111, 11001000, 11001001, 11001010, 11001011, 11001100, 11001101, 11001110, 11001111, 11010000, 11010001, 11010010, 11010011, 11010100, 11010101, 11010110, 11010111, 11011000, 11011001, 11011010, 11011011, 11011100, 11011101, 11011110, 11011111, 11100000, 11100001, 11100010, 11100011, 11100100, 11100101, 11100110, 11100111, 11101000, 11101001, 11101010, 11101011, 11101100, 11101101, 11101110, 11101111, 11110000, 11110001, 11110010, 11110011, 11110100, 11110101, 11110110, 11110111, 11111000, 11111001, 11111010, 11111011, 11111100, 11111101, 11111110, 11111111\}$
- Drawback: $d_{\min} = 2$
- Example: 00000001 and 00000010
- But: $d_{\min} = 2$ and $n = 16$ is not good
- Example: 00000001 and 00000100
- Example: 00000001 and 00001000
- Example: 00000001 and 00010000
- Example: 00000001 and 00100000
- Example: 00000001 and 01000000
- Example: 00000001 and 10000000



Error correcting codes

Example: the thrifty code

- Code containing only binary words with a single '1'

→ $\{0000, 0001, 0010, 0100, 1000\}$

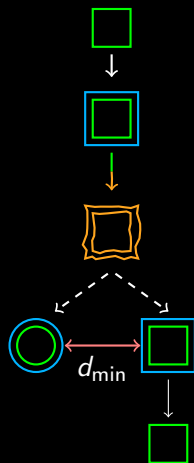
- Drawback: $d_{\min} = 2$

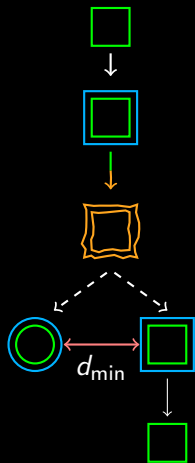
→ $\{0000, 0001, 0010, 0100, 1000, 1001, 1010, 1100\}$

- But: $d_{\min} = 2$ and $d_{\min} = 3$

→ $\{0000, 0001, 0010, 0100, 1000, 1001, 1010, 1100, 1101, 1110\}$

→ $\{0000, 0001, 0010, 0100, 1000, 1001, 1010, 1100, 1101, 1110, 1111\}$





Example: the thrifty code

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



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



- But **easy to decode** and **minimise the energy**:



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

Example: the thrifty code

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



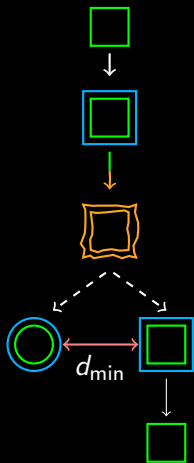
- Drawback: $d_{\min} = 2$:



- But **easy to decode** and **minimise the energy**:



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



Example: the thrifty code

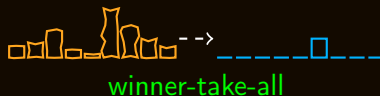
- Code containing only binary words with a single "1":



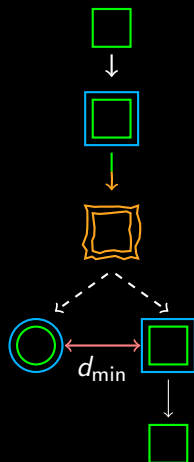
- Drawback: $d_{\min} = 2$:



- But **easy to decode** and **minimise the energy**:



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



Example: the thrifty code

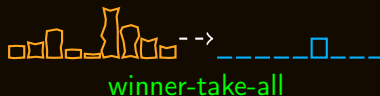
- Code containing only binary words with a single "1":



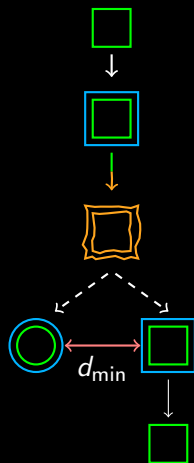
- Drawback: $d_{\min} = 2$:



- But **easy to decode** and **minimise the energy**:



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



1 Associative memories and error correcting codes

- Associative memory
- Error correcting codes

2 Sparse networks, principles and performance

- Storing
- Retrieving
- Performance

3 Developments

- Blurred messages
- Correlated sources
- Sparse messages

4 Conclusion

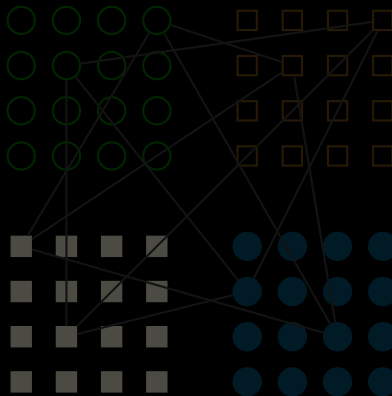
Our model: Storing

Example:

Message to store: 1000001100101001

- For example: a network of $c = 4$ clusters made of $l = 16$ neurons each,

- $\underbrace{1000}_{j_3 \text{ in } c_3}$ $\underbrace{0011}$ $\underbrace{0010}$ $\underbrace{1001}$,



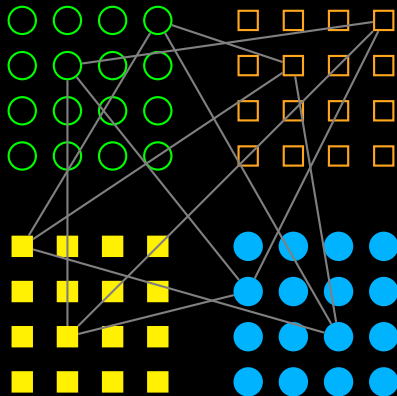
Our model: Storing

Example:

Message to store: 1000001100101001

- For example: a network of $c = 4$ clusters made of $l = 16$ neurons each,

- $\underbrace{1000}_{j_3 \text{ in } c_3}$ $\underbrace{0011}$ $\underbrace{0010}$ $\underbrace{1001}$,



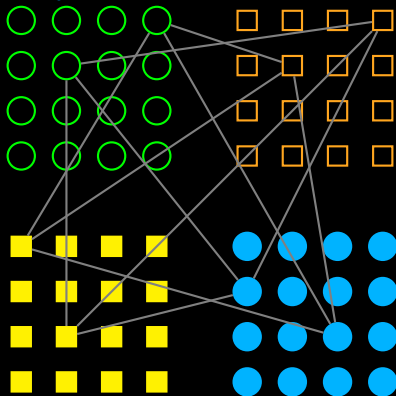
Our model: Storing

Example:

Message to store: 1000001100101001

- For example: a network of $c = 4$ clusters made of $l = 16$ neurons each,

$\underbrace{1000}_{j_1 \text{ in } c_1}$ $\underbrace{0011}_{j_2 \text{ in } c_2}$ $\underbrace{0010}_{j_3 \text{ in } c_3}$ $\underbrace{1001}_{j_4 \text{ in } c_4}$,



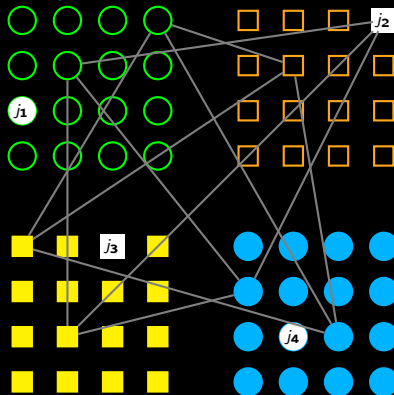
Our model: Storing

Example:

Message to store: 1000001100101001

- For example: a network of $c = 4$ clusters made of $l = 16$ neurons each,

$\underbrace{1000}_{j_1 \text{ in } c_1}$ $\underbrace{0011}_{j_2 \text{ in } c_2}$ $\underbrace{0010}_{j_3 \text{ in } c_3}$ $\underbrace{1001}_{j_4 \text{ in } c_4}$,



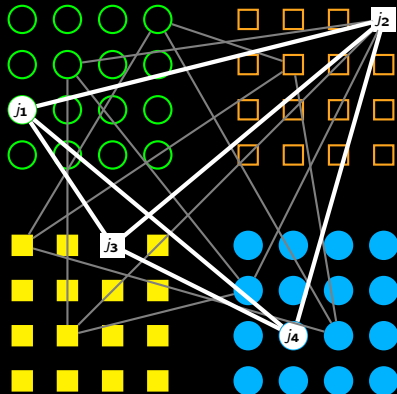
Our model: Storing

Example:

Message to store: 1000001100101001

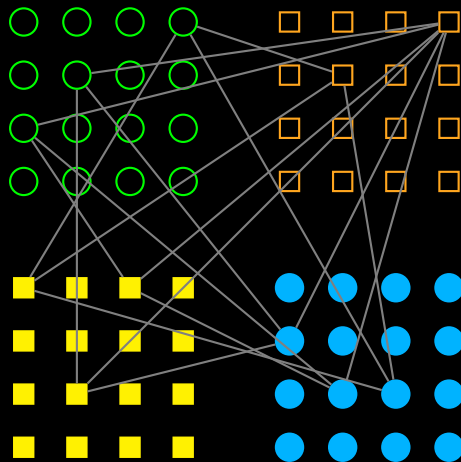
- For example: a network of $c = 4$ clusters made of $l = 16$ neurons each,

$\underbrace{1000}_{j_1 \text{ in } c_1}$ $\underbrace{0011}_{j_2 \text{ in } c_2}$ $\underbrace{0010}_{j_3 \text{ in } c_3}$ $\underbrace{1001}_{j_4 \text{ in } c_4}$,



Our model: retrieving

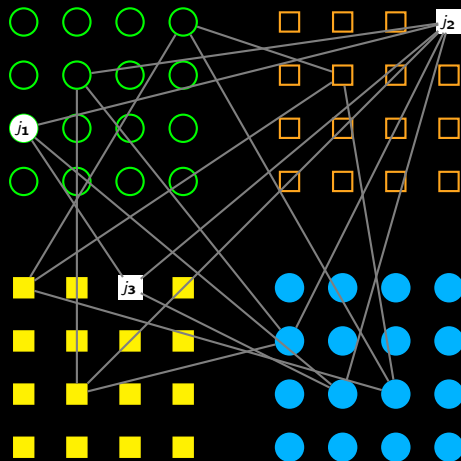
$\underbrace{1000}_{j_1 \text{ in } c_1}$ $\underbrace{0011}_{j_2 \text{ in } c_2}$ $\underbrace{0010}_{j_3 \text{ in } c_3}$????,



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

Our model: retrieving

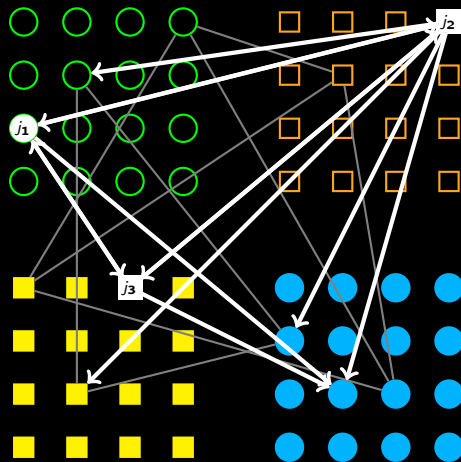
$\underbrace{1000}_{j_1 \text{ in } c_1}$ $\underbrace{0011}_{j_2 \text{ in } c_2}$ $\underbrace{0010}_{j_3 \text{ in } c_3}$????,



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

Our model: retrieving

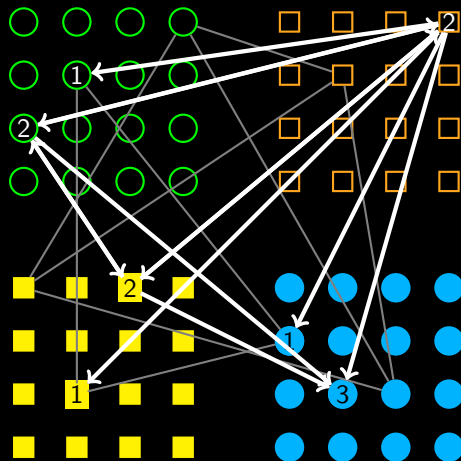
$\underbrace{1000}_{j_1 \text{ in } c_1}$ $\underbrace{0011}_{j_2 \text{ in } c_2}$ $\underbrace{0010}_{j_3 \text{ in } c_3}$????,



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

Our model: retrieving

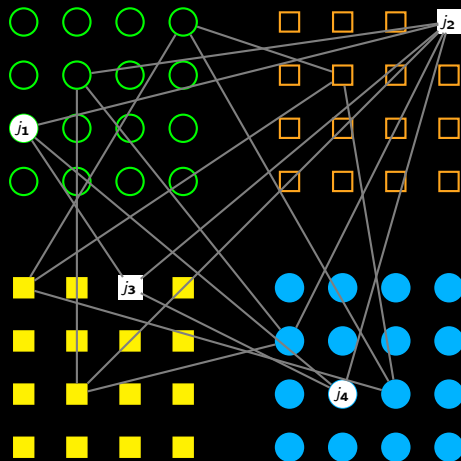
$\underbrace{1000}_{j_1 \text{ in } c_1}$ $\underbrace{0011}_{j_2 \text{ in } c_2}$ $\underbrace{0010}_{j_3 \text{ in } c_3}$????,



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

Our model: retrieving

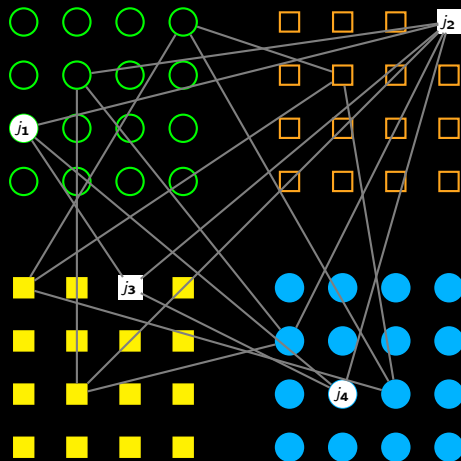
$\underbrace{1000}_{j_1 \text{ in } c_1}$ $\underbrace{0011}_{j_2 \text{ in } c_2}$ $\underbrace{0010}_{j_3 \text{ in } c_3}$ $\underbrace{1001}_{j_4 \text{ in } c_4}$,



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

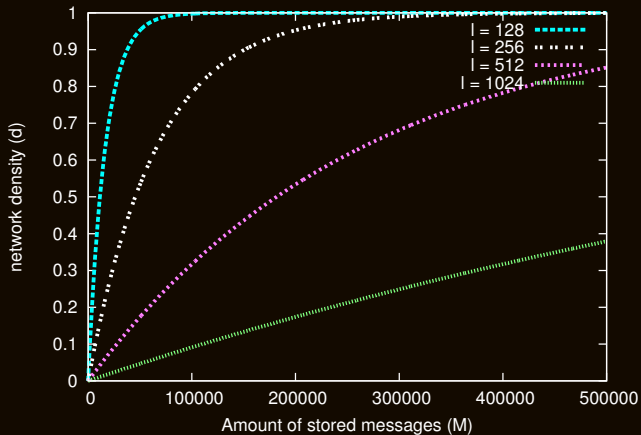
Our model: retrieving

$\underbrace{1000}_{j_1 \text{ in } c_1}$ $\underbrace{0011}_{j_2 \text{ in } c_2}$ $\underbrace{0010}_{j_3 \text{ in } c_3}$ $\underbrace{1001}_{j_4 \text{ in } c_4}$,



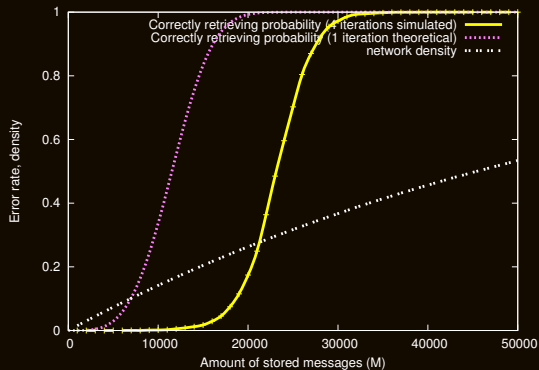
- Local connection,
- Global decoding: sum,
- Local decoding: winner-take-all,
- Possibly iterate the two decodings.

A parameter to assess performance



Performance (1/3)

As an associative memory



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

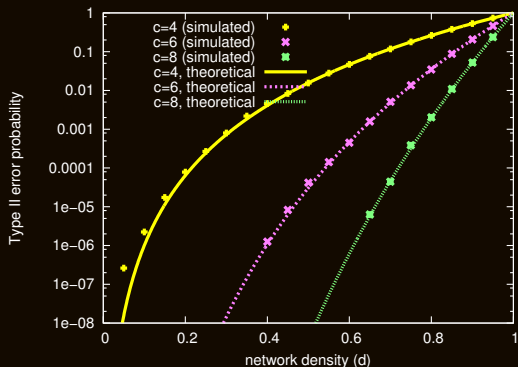
Hopfield network ($n = 790$)



Our network



Classification



Type II error rate for various sizes of clusters c and for $l = 512$ neurons per cluster.

Hopfield network ($n = 740$)

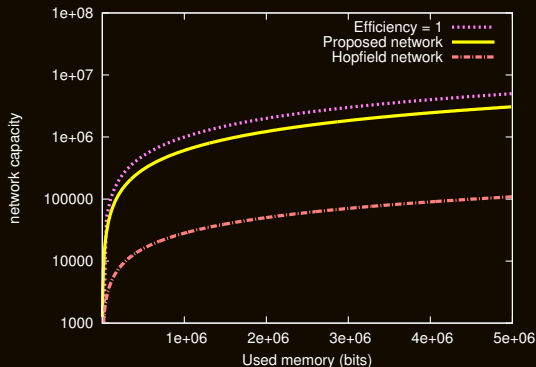


Our network



Comparison of capacities of our network and of the Hopfield one

Performance (3/3)

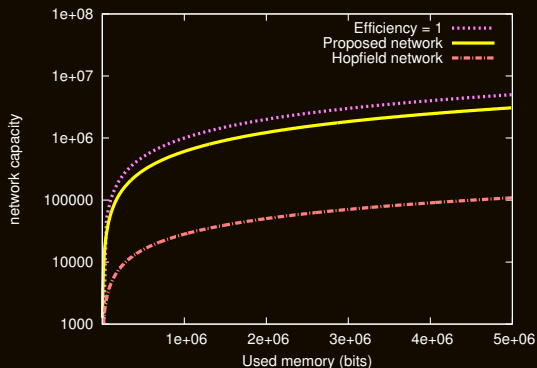


Comparison of the capacities of the Hopfield network with ours (as associative memories) and for the same amount of memory used.

But with some limitations...

Comparison of capacities of our network and of the Hopfield one

Performance (3/3)



Comparison of the capacities of the Hopfield network with ours (as associative memories) and for the same amount of memory used.

But with some limitations...

1 Associative memories and error correcting codes

- Associative memory
- Error correcting codes

2 Sparse networks, principles and performance

- Storing
- Retrieving
- Performance

3 Developments

- Blurred messages
- Correlated sources
- Sparse messages

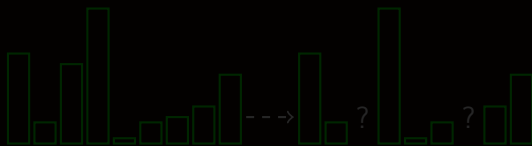
4 Conclusion

Blurred messages

Limitation

Partial messages must contain perfect information.

Noise model



Soft decoding

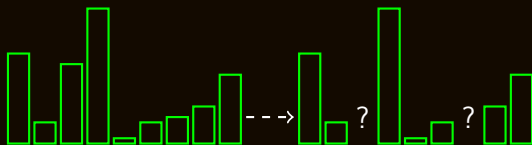


Blurred messages

Limitation

Partial messages must contain perfect information.

Noise model



Soft decoding

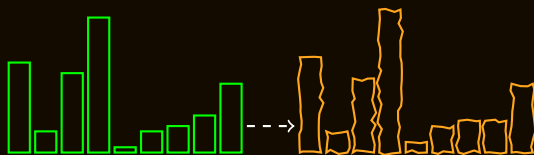


Blurred messages

Limitation

Partial messages must contain perfect information.

Noise model



Soft decoding

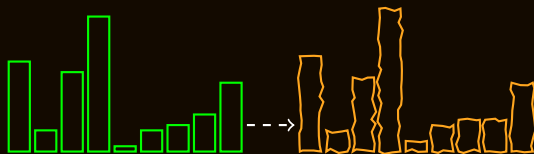


Blurred messages

Limitation

Partial messages must contain perfect information.

Noise model



Soft decoding

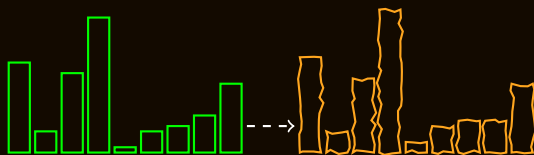


Blurred messages

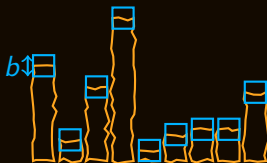
Limitation

Partial messages must contain perfect information.

Noise model



Soft decoding

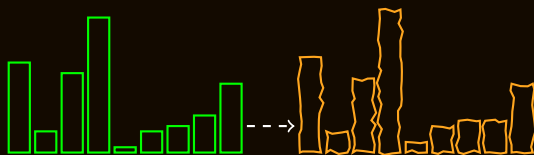


Blurred messages

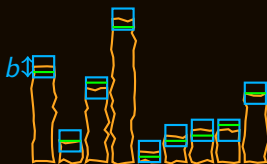
Limitation

Partial messages must contain perfect information.

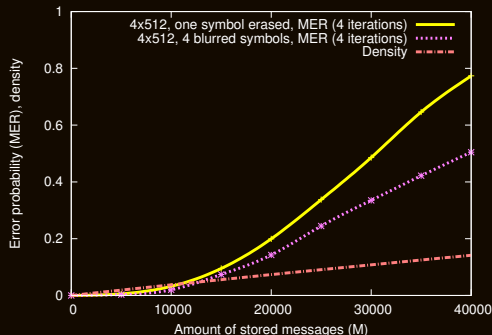
Noise model



Soft decoding



Simulations



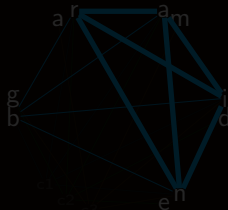
Comparison of performance when messages are partially erased and when they are blurred ($b = 5$).

Correlated messages

Limitation

With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy



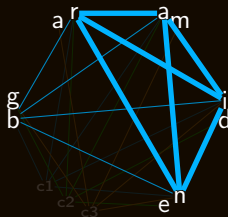
Correlated messages

Limitation

With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy

brain



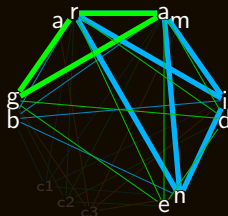
Correlated messages

Limitation

With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy

brain
grade



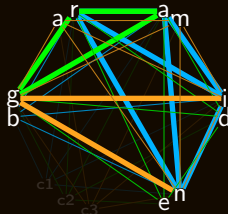
Correlated messages

Limitation

With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy

brain
grade
gamin



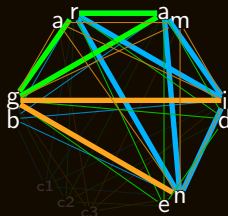
Correlated messages

Limitation

With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy

brain
grade
gamin
grain



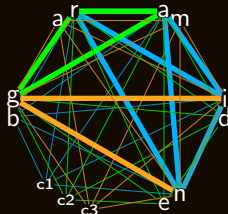
Correlated messages

Limitation

With correlations grows the number of Type II errors.

Fighting correlation by adding random redundancy

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



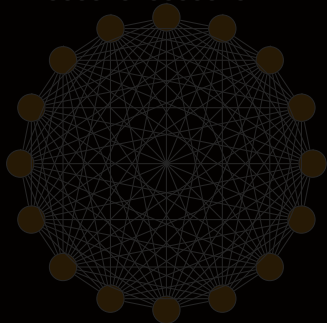
Towards a fourth level of sparsity

Limitations

- Clusters must be large and few,
- Stored messages are all of the same length.

Illustration

0000101000001011



Idea

- Shorter messages,
- Clusters and thrifty codes,
- Sparse network,
- Sparse messages.

Solution

- Global winner-take-all

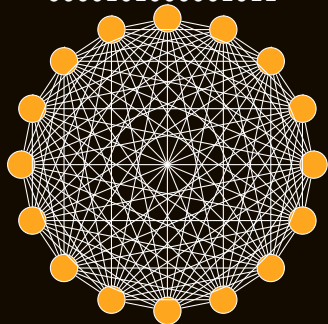
Towards a fourth level of sparsity

Limitations

- Clusters must be large and few,
- Stored messages are all of the same length.

Illustration

0000101000001011



Idea

- Shorter messages,
- Clusters and thorny codes
- Sparse network

Solution

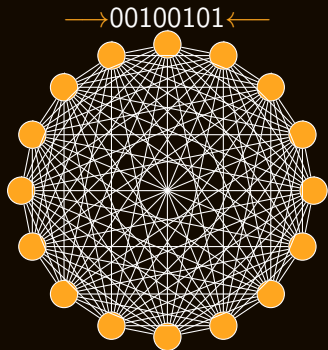
Global winner-take-all

Towards a fourth level of sparsity

Limitations

- Clusters must be large and few,
- Stored messages are all of the same length.

Illustration



Idea

- 1 Shorter messages,
- 2 Clusters and thrifty codes,
- 3 Sparse network,
- 4 Sparse messages.

Solution

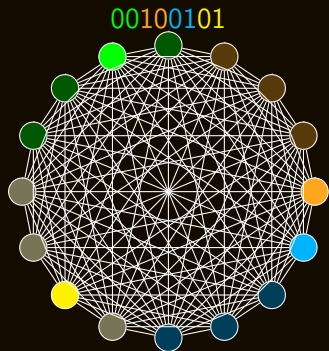
Global winner-take-all

Towards a fourth level of sparsity

Limitations

- Clusters must be large and few,
- Stored messages are all of the same length.

Illustration



Idea

- 1 Shorter messages,
- 2 Clusters and thrifty codes,
- 3 Sparse network,
- 4 Sparse messages.

Solution

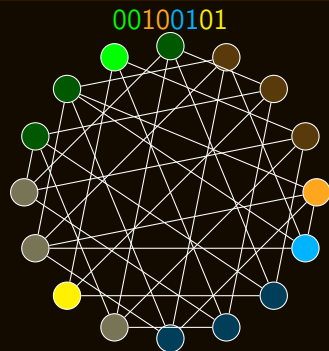
Global winner take-all

Towards a fourth level of sparsity

Limitations

- Clusters must be large and few,
- Stored messages are all of the same length.

Illustration



Idea

- 1 Shorter messages,
- 2 Clusters and thrifty codes,
- 3 Sparse network,
- 4 Sparse messages.

Solution

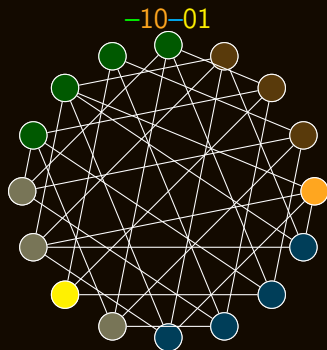
Global winner-take-all

Towards a fourth level of sparsity

Limitations

- Clusters must be large and few,
- Stored messages are all of the same length.

Illustration



Idea

- 1 Shorter messages,
- 2 Clusters and thrifty codes,
- 3 Sparse network,
- 4 Sparse messages.

Solution

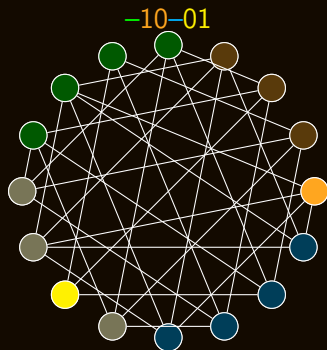
Global winner take all

Towards a fourth level of sparsity

Limitations

- Clusters must be large and few,
- Stored messages are all of the same length.

Illustration



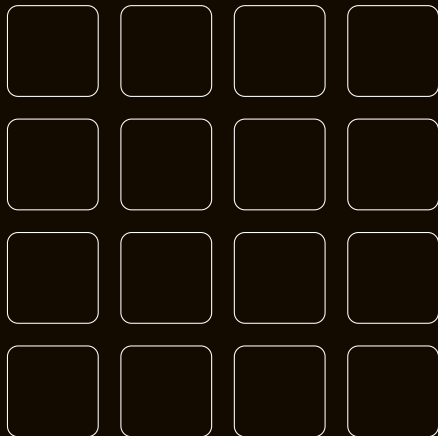
Idea

- 1 Shorter messages,
- 2 Clusters and thrifty codes,
- 3 Sparse network,
- 4 Sparse messages.

Solution

- Global winner-take-all.

Illustration



Idea

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

Interests

- Diversity
- Stored messages length may vary

Global winner-take-all

Illustration



Idea

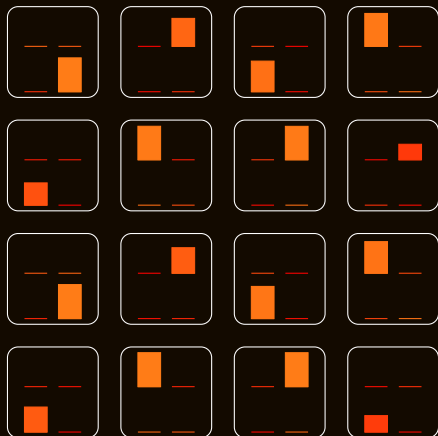
- After global message passing. . .
- After local maximum selections. . .
- Global maximum selection.

Interests

- Diversity
- Stored messages length may vary

Global winner-take-all

Illustration



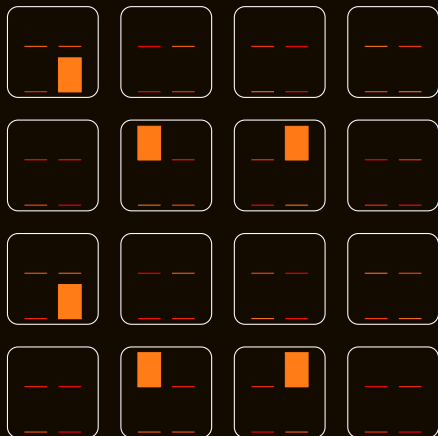
Idea

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

Interests

- Diversity
- Stored messages length may vary

Illustration



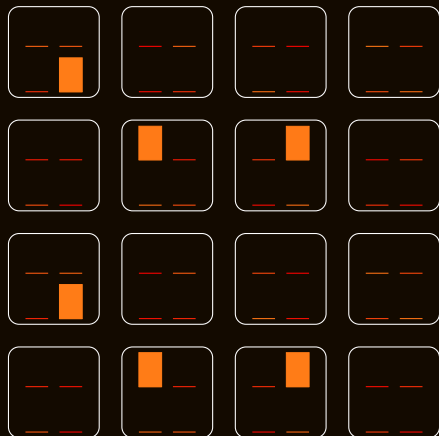
Idea

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

Interests

- Diversity
- Stored messages length may vary

Illustration



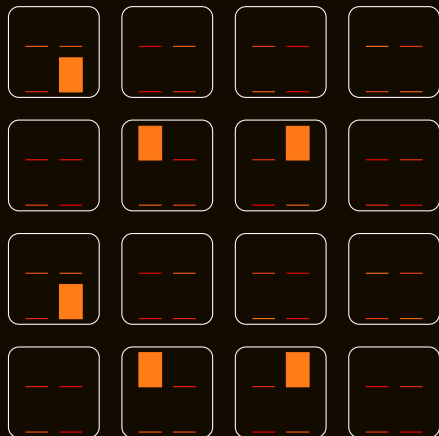
Idea

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

Interests

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

Illustration



Idea

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

Interests

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

1 Associative memories and error correcting codes

- Associative memory
- Error correcting codes

2 Sparse networks, principles and performance

- Storing
- Retrieving
- Performance

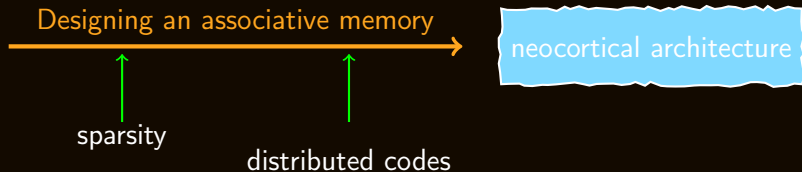
3 Developments

- Blurred messages
- Correlated sources
- Sparse messages

4 Conclusion

Conclusion

Approach

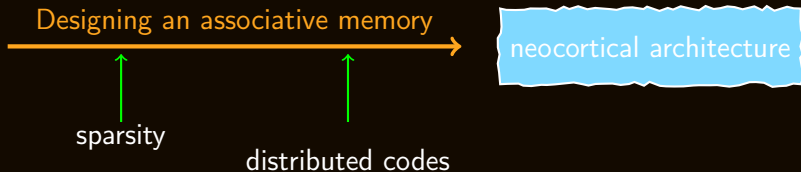


Results

- Nearly optimal capacities, substantial diversities,

Conclusion

Approach

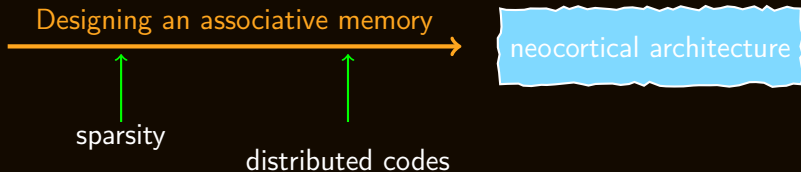


Results

- Nearly optimal capacities, substantial diversities,
- Massively parallel architecture,
- Analogies with neurobiological architecture and functioning,
- Robustness, resiliency. . . ,
- Degrees of freedom: inhibitions, time, weights,
- No trade off required between performance and plausibility.

Conclusion

Approach

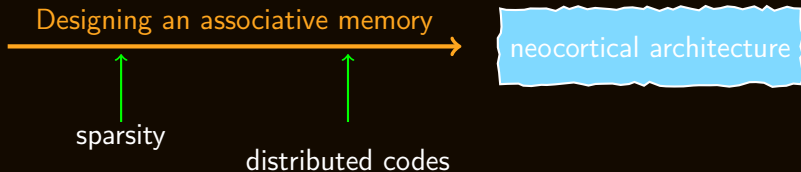


Results

- Nearly optimal capacities, substantial diversities,
- Massively parallel architecture,
- Analogies with neurobiological architecture and functioning,
- Robustness, resiliency. . . ,
- Degrees of freedom: inhibitions, time, weights,
- No trade off required between performance and plausibility.

Conclusion

Approach

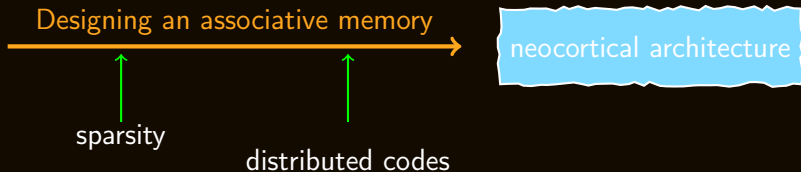


Results

- Nearly optimal capacities, substantial diversities,
- Massively parallel architecture,
- Analogies with neurobiological architecture and functioning,
- Robustness, resiliency. . . ,
- Degrees of freedom: inhibitions, time, weights,
- No trade off required between performance and plausibility.

Conclusion

Approach

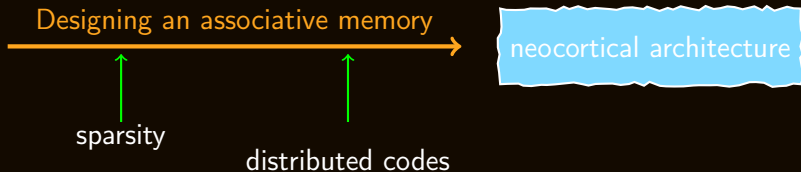


Results

- Nearly optimal capacities, substantial diversities,
- Massively parallel architecture,
- Analogies with neurobiological architecture and functioning,
- Robustness, resiliency. . . ,
- Degrees of freedom: inhibitions, time, weights,
- No trade off required between performance and plausibility.

Conclusion

Approach

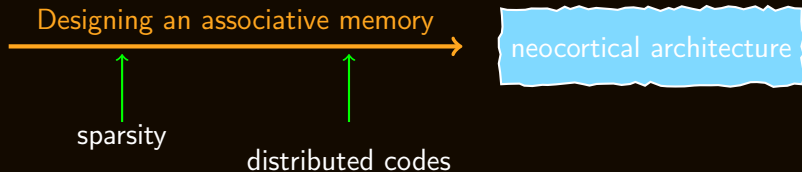


Results

- Nearly optimal capacities, substantial diversities,
- Massively parallel architecture,
- Analogies with neurobiological architecture and functioning,
- Robustness, resiliency. . . ,
- Degrees of freedom: inhibitions, time, weights,
- No trade off required between performance and plausibility.

Conclusion

Approach



Results

- Nearly optimal capacities, substantial diversities,
- Massively parallel architecture,
- Analogies with neurobiological architecture and functioning,
- Robustness, resiliency. . . ,
- Degrees of freedom: inhibitions, time, weights,
- No trade off required between performance and plausibility.