Neural Association and Coding Theory

Amir Hesam Salavati

A joint project with:

Raj K. Kumar

EPFL

algolma

In this talk...

- Motivation
- Introduction to Neural Networks
- Current problem formulation
- Proposed solution(s)
- A glimpse to future works
- Questions and discussion

Motivation

- Coding theory: finding methods to overcome the issue of noise in communications.

- Project goals:
  - Employ mathematical background of coding theory to analyze neural networks.
  - Inspire from neural operations to design codes.
- So far, we have been busy with the first.

Motivation

- Coding theory: finding methods to overcome the issue of noise in communications.
- In that sense, human brain is decoder:

Missespeled wrods!

- Project goals:
  - Employ mathematical background of coding theory to analyze neural networks.
  - Inspire from neural operations to design codes.
- So far, we have been busy with the first.

Motivation

- Coding theory: finding methods to overcome the issue of noise in communications.
- In that sense, human brain is decoder:

Missespeled wrods!

- Project goals:
  - Employ mathematical background of coding theory to analyze neural networks.
  - Inspire from neural operations to design codes.
- So far, we have been busy with the first.

Motivation

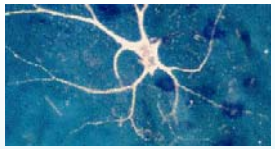
- Coding theory: finding methods to overcome the issue of noise in communications.
- In that sense, human brain is decoder:

Missespeled wrods!

- Project goals:
  - Employ mathematical background of coding theory to analyze neural networks.
  - Inspire from neural operations to design codes.
- So far, we have been busy with the first.

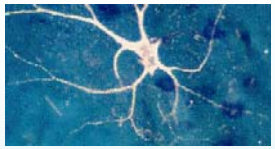
A BRIEF INTRODUCTION TO NEURAL NETWORKS

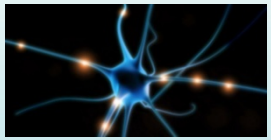
First Some Terminologies



- A Neuron is the most basic entity in the nervous system that processes information.
- Neurons also transmit information via electrical pulses, called spikes.
- The information about the outside world is encoded in the spike train:
  - usually it is encoded in the number of spikes per second (rate code),
  - But some times in their relative timing (time code) as well.

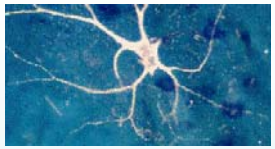
First Some Terminologies



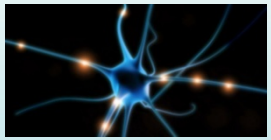


- A Neuron is the most basic entity in the nervous system that processes information.
- Neurons also transmit information via electrical pulses, called spikes.
- The information about the outside world is encoded in the spike train:
  - usually it is encoded in the number of spikes per second (rate code),
  - But some times in their relative timing (time code) as well.

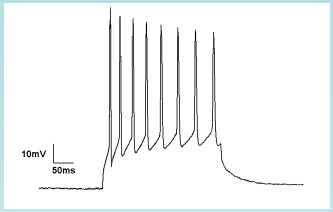
First Some Terminologies



- A Neuron is the most basic entity in the nervous system that processes information.



- Neurons also transmit information via electrical pulses, called spikes.
- The information about the outside world is encoded in the spike train:



  - usually it is encoded in the number of spikes per second (rate code),
  - But some times in their relative timing (time code) as well.

Neural Model

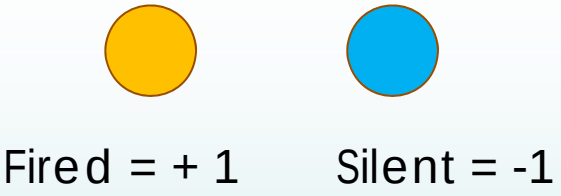
Neural Model

- Binary state neurons:

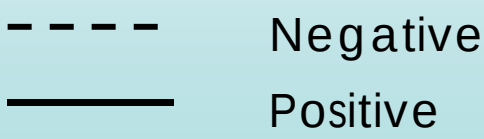
Fired = + 1 Silent = -1

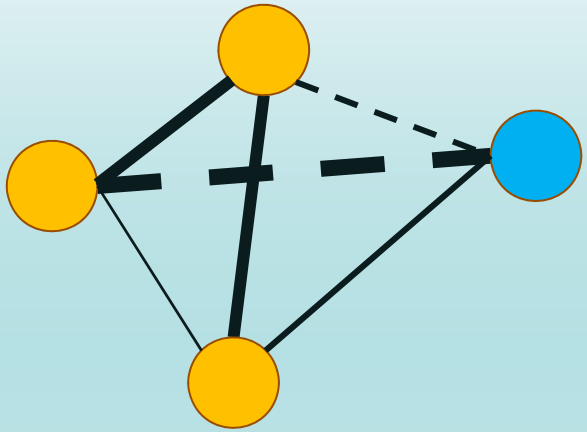
Neural Model

- Binary state neurons:



- The neural network:





Neural Model

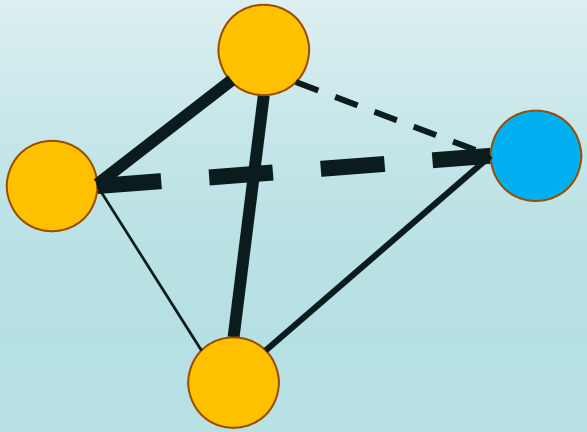
- Binary state neurons:

Fired = + 1 Silent = -1

- The neural network:

Negative

Positive



- Update rule:

$$s_i = \begin{cases} +1 & if \ \sum_{j=1}^{N} w_{ij} s_j \geq \Theta \\ -1 & o.w. \end{cases}$$

Neural Associative Memory

- Hetero-association

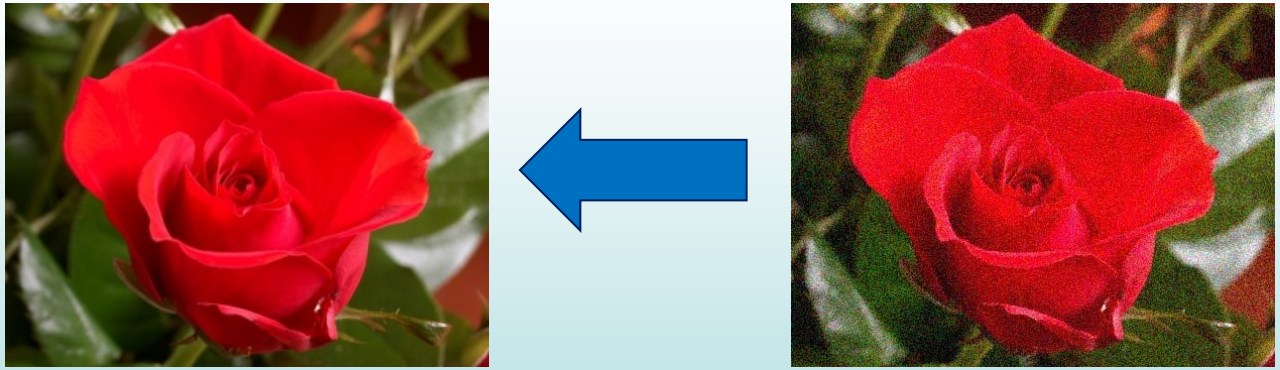
Neural Associative Memory

- Auto-association

- Hetero-association

Neural Associative Memory

- Auto-association



- Hetero-association

Neural Associative Memory

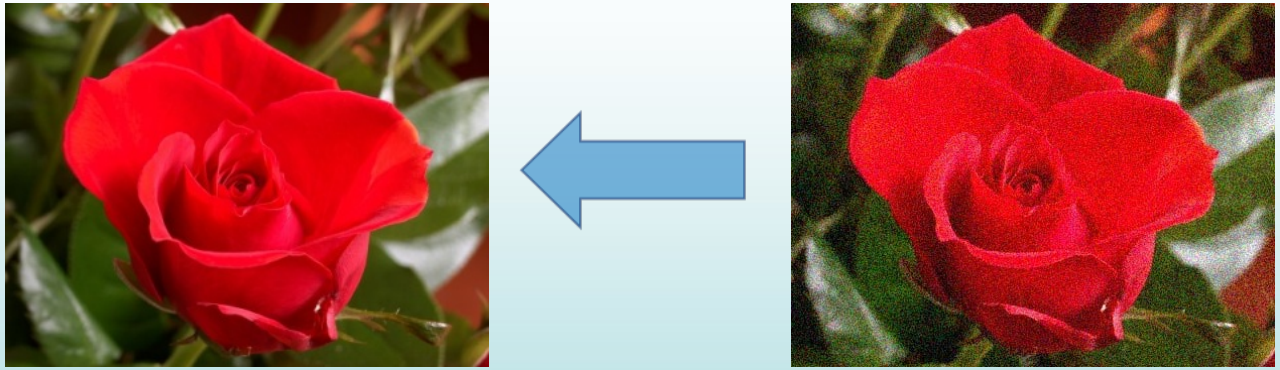
- Auto-association

- Hetero-association

Rose

Neural Associative Memory

- Auto-association



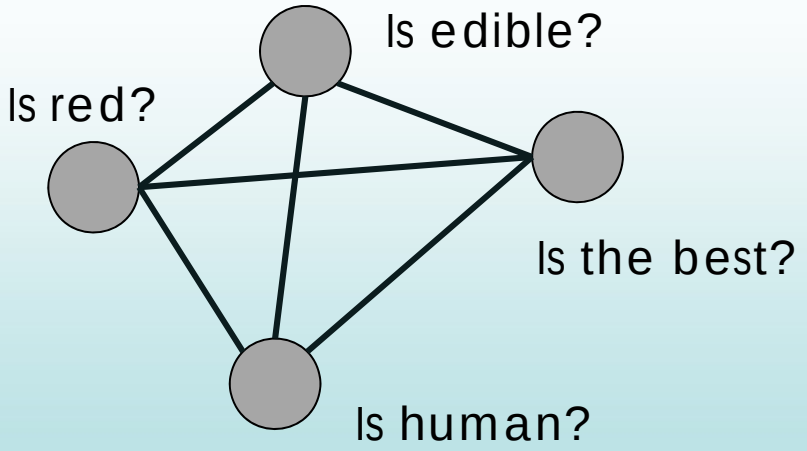
- Hetero-association

Roze → Rose ↔

Neural Association (contd.)

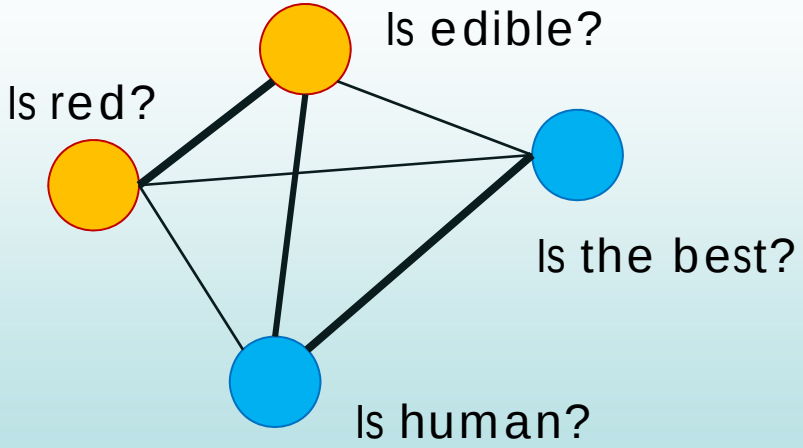
Neural Association (contd.)

- Learning phase: adjust the weights



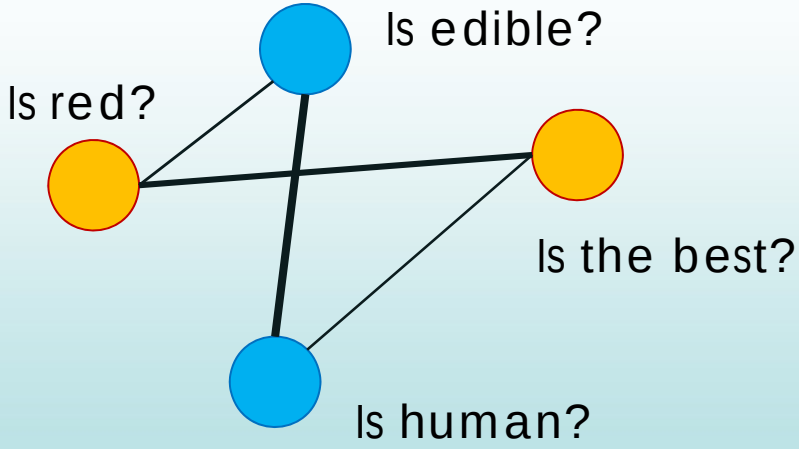
Neural Association (contd.)

- Learning phase: adjust the weights



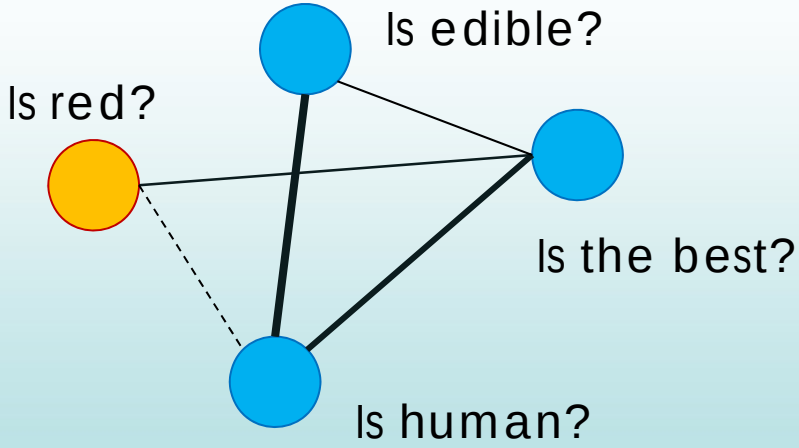
Neural Association (contd.)

- Learning phase: adjust the weights



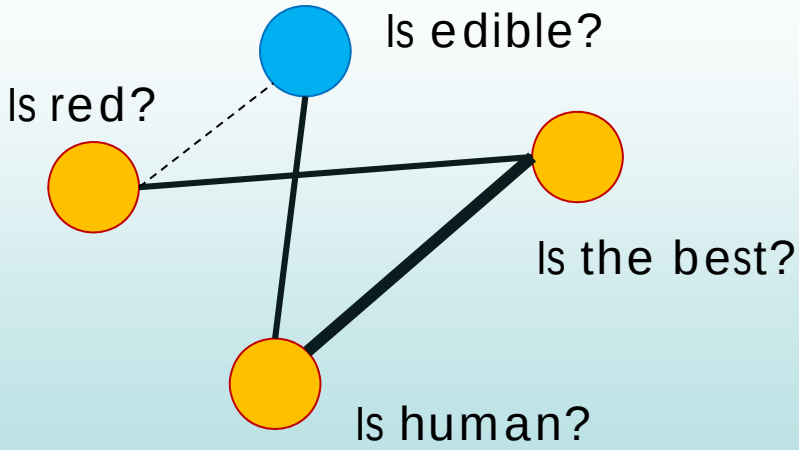
Neural Association (contd.)

- Learning phase: adjust the weights



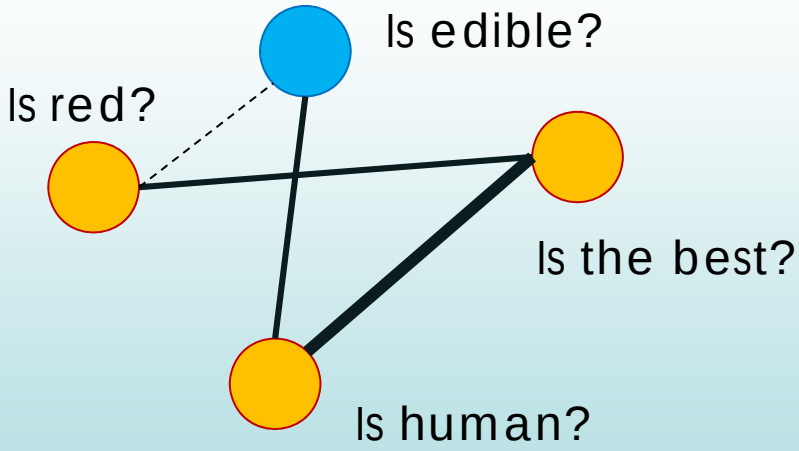
Neural Association (contd.)

- Learning phase: adjust the weights



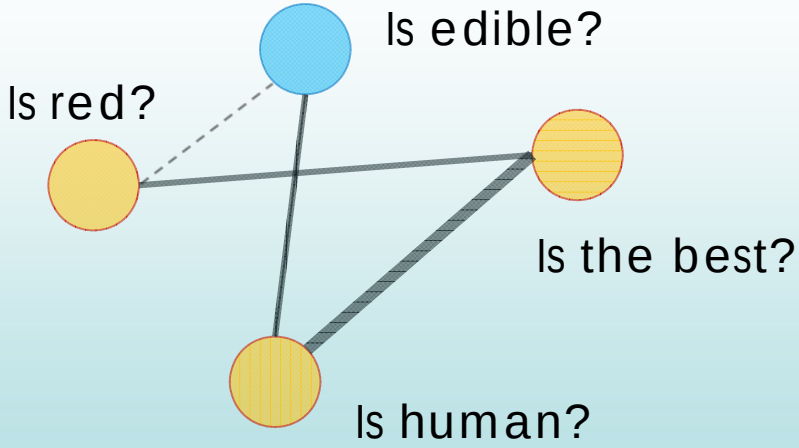
Neural Association (contd.)

- Learning phase: adjust the weights



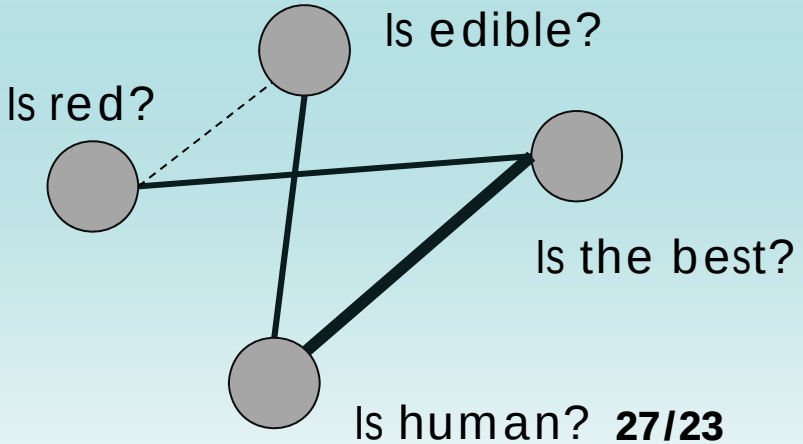
Neural Association (contd.)

- Learning phase: adjust the weights



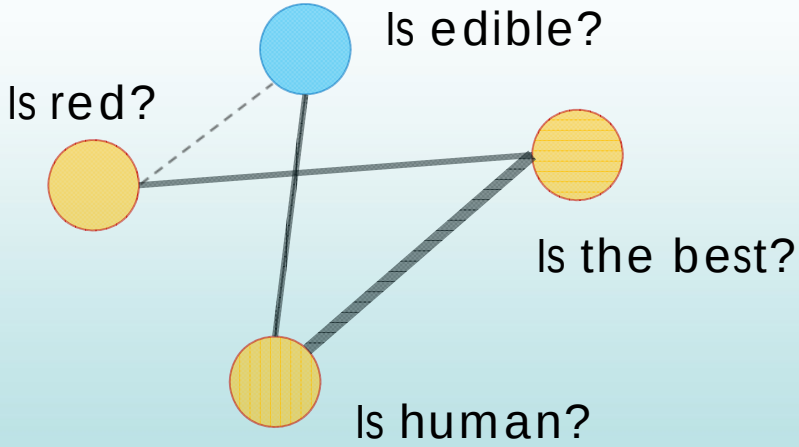
- Recall phase:

  get rid of noise



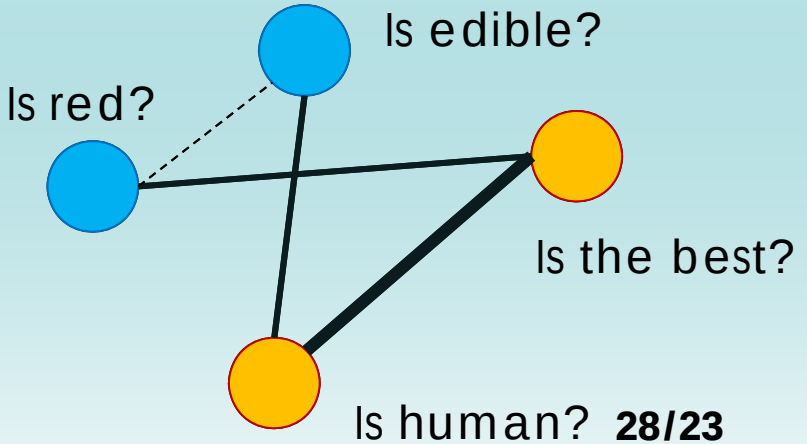
Neural Association (contd.)

- Learning phase: adjust the weights



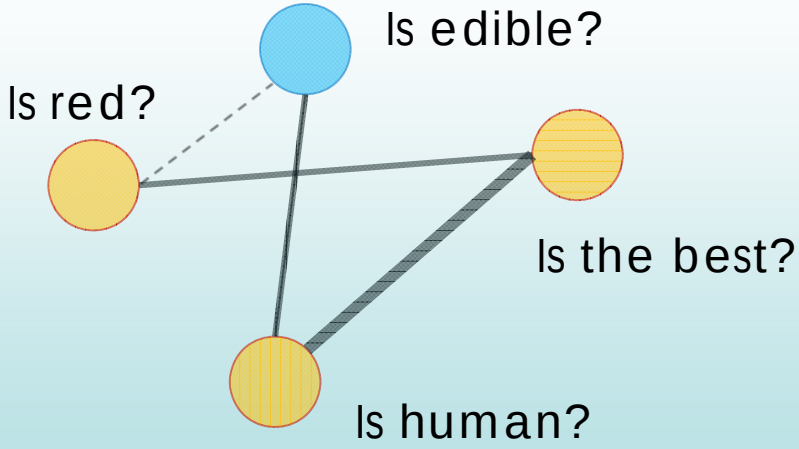
- Recall phase:

  get rid of noise



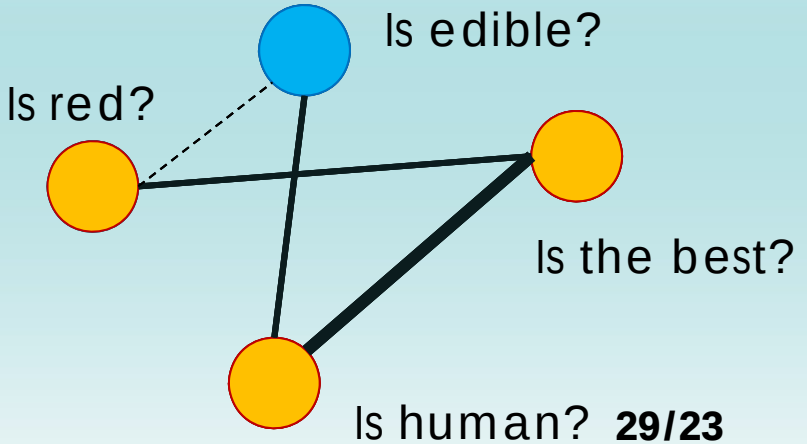
Neural Association (contd.)

- Learning phase: adjust the weights



- Recall phase:

  get rid of noise



PROBLEM FORMULATION

Formal Problem Formulation

Formal Problem Formulation

- **Given**:

Formal Problem Formulation

- **Given**:
  - An *undirected complete graph* of *binary neurons*

Formal Problem Formulation

- **Given**:
  - An *undirected complete graph* of *binary neurons*
  - Neural update rule:

$$s_i = \begin{cases} +1 & if \ \sum_{j=1}^{N} w_{ij} s_j \geq \Theta \\ -1 & o.w. \end{cases}$$

Formal Problem Formulation

- **Given**:
  - An *undirected complete graph* of *binary neurons*
  - Neural update rule:

$$s_i = \begin{cases} +1 & if \ \sum_{j=1}^{N} w_{ij} s_j \geq \Theta \\ -1 & o.w. \end{cases}$$

- **Required**: the weight matrix

$$w_{ij} = f(X^1, X^2, \ldots, X^M)$$

Formal Problem Formulation

- **Given**:
  - An *undirected complete graph* of *binary neurons*
  - Neural update rule:

$$s_i = \begin{cases} +1 & if \ \sum_{j=1}^{N} w_{ij} s_j \geq \Theta \\ -1 & o.w. \end{cases}$$

- **Required**: the weight matrix

$$w_{ij} = f(X^1, X^2, \ldots, X^M)$$

  - Stability

Formal Problem Formulation

- **Given**:
  - An *undirected complete graph* of *binary neurons*
  - Neural update rule:

$$s_i = \begin{cases} +1 & if \ \sum_{j=1}^{N} w_{ij} s_j \geq \Theta \\ -1 & o.w. \end{cases}$$

- **Required**: the weight matrix

$$w_{ij} = f(X^1, X^2, \ldots, X^M)$$

  - Stability
  - Noise tolerance

Hopfield Associative Memory

- Hopfield weighting rule:

$$w_{ij} = \frac{1}{n} \sum_{\mu=1}^{M} x_i^\mu x_j^\mu$$

  - x_i^μ is the *ith* bit of pattern μ.

  - Storage capacity: maximum number of patterns that can be memorized

- For Hopfield (with random patterns):

Hopfield Associative Memory

- Hopfield weighting rule:

$$w_{ij} = \frac{1}{n} \sum_{\mu=1}^{M} x_i^\mu x_j^\mu$$

  - x_i^μ is the *ith* bit of pattern μ.

- Storage capacity: maximum number of patterns that can be memorized
- For Hopfield (with random patterns):

- For *any** other method: at best *2n*

Hopfield Associative Memory

- Hopfield weighting rule:

$$w_{ij} = \frac{1}{n} \sum_{\mu=1}^{M} x_i^\mu x_j^\mu$$

  - x_i^μ is the *ith* bit of pattern μ.

- Storage capacity: maximum number of patterns that can be memorized
- For Hopfield (with random patterns):

Vanishing *bit* error probability:	Vanishing *pattern* error probability:
$M_{max} = 0.138\ n$	$M_{max} = O(n/log(n))$

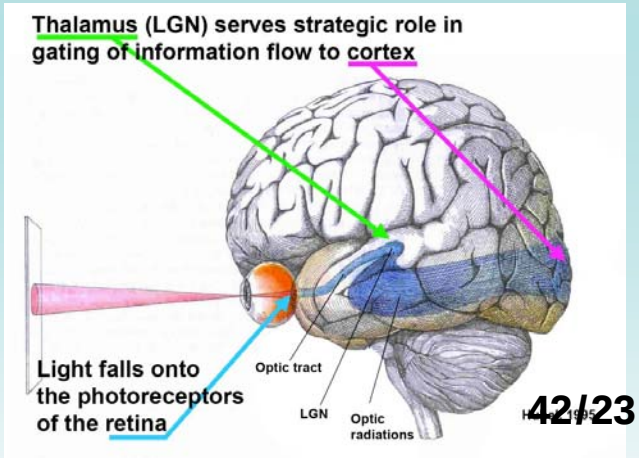
- For *any** other method: at best *2n*

There Is a Gap, But Why?

- Random sequences → bad distance properties
- We suggest considering non-random patterns.
  - Not all 2^n binary patterns are allowed.
- Is it biologically meaningful?

There Is a Gap, But Why?

- Random sequences → bad distance properties
- We suggest considering non-random patterns.
  - Not all 2^n binary patterns are allowed.
- Is it biologically meaningful?
- Could be: visual pathway is a good example.



Memorizing Patterns with Low Correlation

- First candidate: codewords of a linear code.
  - Hopfield network can not learn XOR!
- Second attempt: **Gold sequences**.
  - Gold family is a set of binary vectors with optimal correlation properties.
- Why low correlation?

Memorizing Patterns with Low Correlation

- First candidate: codewords of a linear code.
  - 
  - Hopfield network can not learn XOR!
- Second attempt: **Gold sequences**.
  - Gold family is a set of binary vectors with optimal correlation properties.
- Why low correlation?

Memorizing Patterns with Low Correlation

- First candidate: codewords of a linear code.
  - Hopfield network can not learn XOR!
- Second attempt: **Gold sequences**.
  - Gold family is a set of binary vectors with optimal correlation properties.
- Why low correlation?

$$h_i = \sum_{j=1}^{n} w_{ij} x_j^p$$

Memorizing Patterns with Low Correlation

- First candidate: codewords of a linear code.
  - Hopfield network can not learn XOR!
- Second attempt: **Gold sequences**.
  - Gold family is a set of binary vectors with optimal correlation properties.
- Why low correlation?

$$h_i = \sum_{j=1}^{n} w_{ij} x_j^p = \frac{1}{n} \sum_{\mu=1}^{M} x_i^\mu \sum_{j=1}^{n} x_j^p x_j^\mu$$

Memorizing Patterns with Low Correlation

- First candidate: codewords of a linear code.
  - Hopfield network can not learn XOR!
- Second attempt: **Gold sequences**.
  - Gold family is a set of binary vectors with optimal correlation properties.
- Why low correlation?

$$h_i = \sum_{j=1}^{n} w_{ij} x_j^p = \frac{1}{n} \sum_{\mu=1}^{M} x_i^\mu \sum_{j=1}^{n} x_j^p x_j^\mu$$

$$= x_i^p + \frac{1}{n} \sum_{\mu=1 \neq p}^{M} x_i^\mu \sum_{j=1}^{n} x_j^p x_j^\mu$$

Undesired correlation term

Memorizing Patterns with Low Correlation (contd.)

- With original Hopfield rule and Gold sequences, we will have:
  - ✓ Stability
  - ✗ But not error correction
- Solution: modify the weighting rule

  - The scaling factors alternate in sign such that the correlation terms cancel out.

Memorizing Patterns with Low Correlation (contd.)

- With original Hopfield rule and Gold sequences, we will have:
  - ✓ Stability
  - ✗ But not error correction
- Solution: modify the weighting rule

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{M} \lambda_\mu x_i^\mu x_j^\mu$$

  - The scaling factors alternate in sign such that the correlation terms cancel out.

Some Results

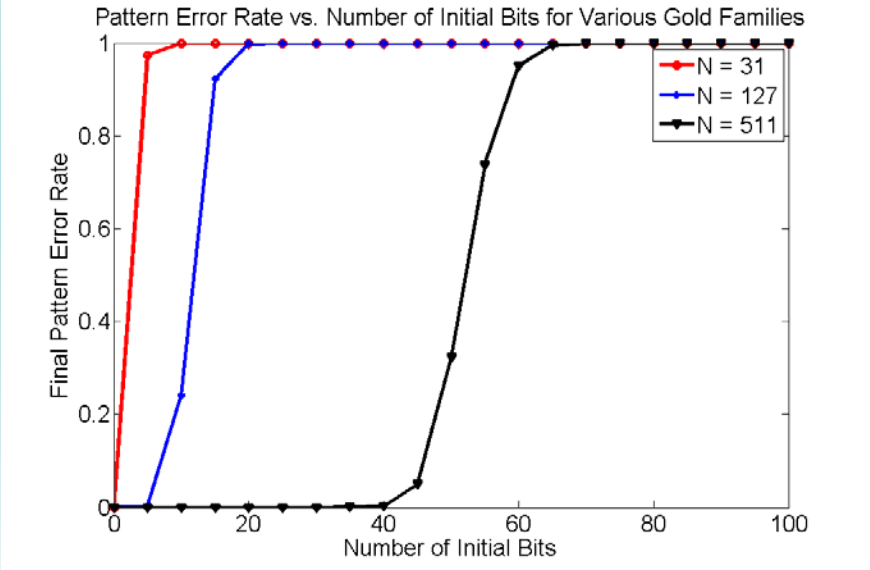
- Increasing capacity to M = N
  - As compared to M = O(N/log(N))
- And maintain error correction

(without error correction, W = identity is the trivial choice for M = N)

Some Results

- Increasing capacity to M = N
  - As compared to M = O(N/log(N))
- And maintain error correction

  (without error correction, W = identity is the trivial choice for M = N)
- Simulation Results:



A GLIMPSE TOWARD FUTURE

Lattice Codes & Neural Associative Memory

- Two main differences between a neural graph and that of an LDPC code:
  - Neural update rule
  - GF(2) operation
- The latter can be resolved by considering codes on real fields, like lattice codes.
- The trick is to consider spike train as the neurons state.

Lattice Codes & Neural Associative Memory

- Two main differences between a neural graph and that of an LDPC code:
  - Neural update rule
  - GF(2) operation
- The latter can be resolved by considering codes on real fields, like lattice codes.
- The trick is to consider spike train as the neurons state.

The Model

- Everything the same as before.
- Except the update rule: $s_i = f(\sum_{j=1}^{N} w_{ij} s_j)$
- **Constraints**:

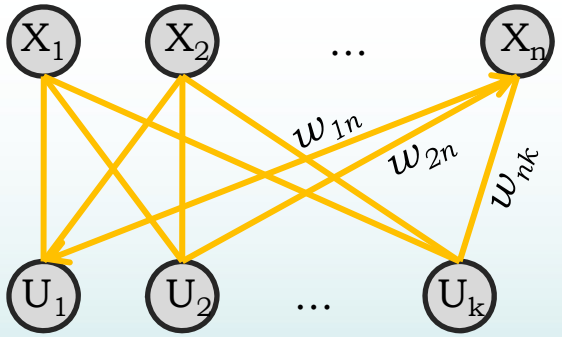
1) Neural states must be positive integers
   - Although, we might have a trick for this in our sleeves as well!
2) Reasonable weights

Some Preliminary Works

- Asymmetric

  Hetero-association:

  - Lattice code: x = u.G



Hetero-association

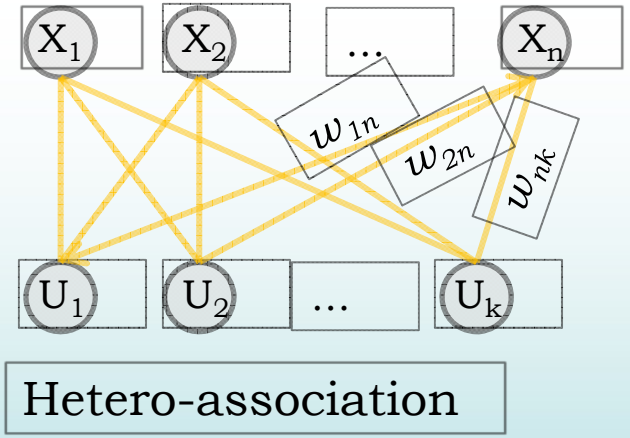
- **Learning phase**
  - Memorize the association "x ↔ u"
  - Simply done by $W_f = G$

Some Preliminary Works

- Asymmetric

  Hetero-association:

  - Lattice code: x = u.G



- **Learning phase**
  - Memorize the association "x ↔ u"
  - Simply done by $W_f = G$

Some Preliminary Works (contd.)

- **Recall phase**:

  Retrieve ***u*** from a noisy version of ***x***

- If we pick W_b such that $G.W_b = I$ then the input sum to z_i is:

$$h_i = (x + e)W_b = u + eW_b = u + e'$$

- So we need **G** such that e' is small.

Some Results

- Message symbols: vectors of length ***k*** with elements either being ***up*** and ***down***.
- **G**: a full rank k×n random matrix with large norm.
- Neural output
  - At code neuron:
  - At message neurons:
- Such a simple scheme can result in exponential capacities.

Some Results

- Message symbols: vectors of length ***k*** with elements either being ***up*** and ***down***.
- **G**: a full rank k×n random matrix with large norm.
- Neural output
  - At code neuron: $x = \min(zW_f, x_{max})$
  - At message neurons: $z = Tr(xW_b)$
- Such a simple scheme can result in exponential capacities.

Some Results

- Message symbols: vectors of length ***k*** with elements either being ***up*** and ***down***.
- **G**: a full rank k×n random matrix with large norm.
- Neural output
  - At code neuron: $x = \min(zW_f, x_{max})$
  - At message neurons: $z = Tr(xW_b)$
- Such a simple scheme can result in exponential capacities.

Future Works

Future Works

- Finding proper weight matrices for neural networks based on lattice codes
- Or working on weight matrices based on parity check matrix of lattice codes

Future Works

- Finding proper weight matrices for neural networks based on lattice codes
- Or working on weight matrices based on parity check matrix of lattice codes

- Multi-layered neural networks:
  - Hopfield networks can not learn XOR
  - Boltzmann machines can!

Thank You!

Any

Questions?