EXPONENTIAL PATTERN RETRIEVAL CAPACITY WITH NON-BINARY ASSOCIATIVE MEMORY

Amir Hesam Salavati

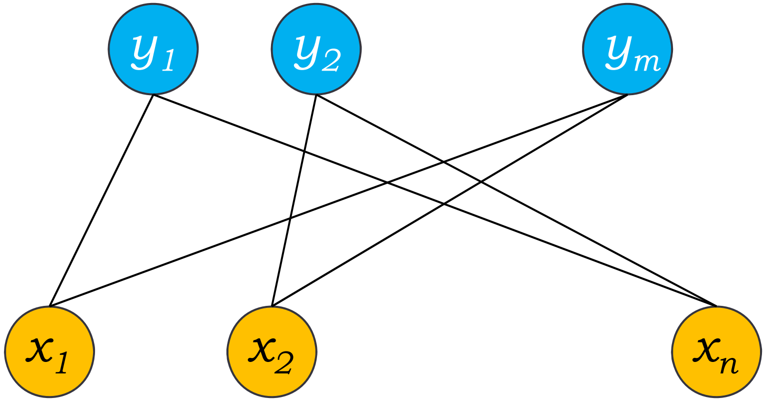
Joint work with: Raj K. Kumar Amin Shokrollahi

In This Talk...

- The problem
- Our proposed solution
  - Intuition
  - Details
  - Results
- Work in progress
- Conclusions and final remarks

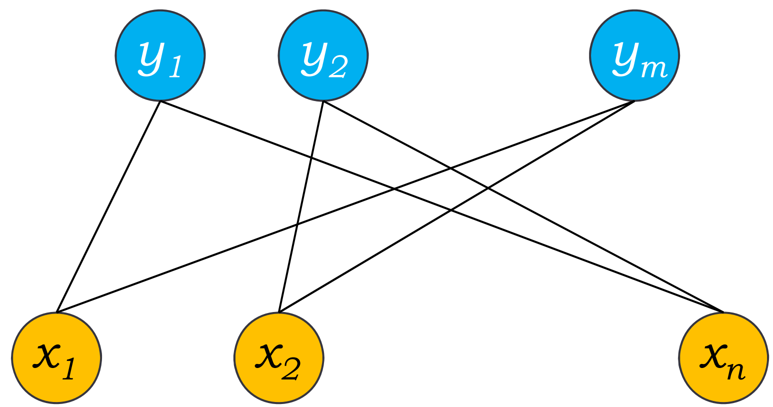
The Problem in a Nutshell

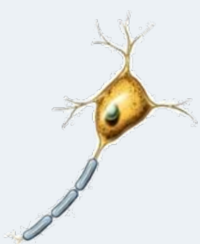
- For a coding theorist:
  - Given: A *parity check graph*.
  - Required: A "*simple*" message passing decoding algorithm with restrictions on decoding nodes.



The Problem in a Nutshell

- For a coding theorist:
  - Given: A *parity check graph*.
  - Required: A "*simple*" message passing decoding algorithm with restrictions on decoding nodes.





- For a neuroscientist:
  - Given: A *hetero-associative* neural network.
  - Required: A method to increase the storage capacity.

Why Is It Interesting?

- Associative memory problem: store a set of random binary patterns of length *n* reliably. Later, return the closest stored pattern in response to a noisy query.

WHY IS IT INTERESTING?

Introduction

- Hopfield, 1982
- Amit et al., 1985
- McEliece et al. 1987
- Komlos et al., 1993
- Muezzinoglu et al., 2003

- Associative memory problem: store a set of random binary patterns of length *n* reliably. Later, return the closest stored pattern in response to a noisy query.
- For the past three decades, best neural realizations yield pattern retrieval capacities linear in *n*.

WHY IS IT INTERESTING?

Introduction

- Hopfield, 1982
- Amit et al., 1985
- McEliece et al. 1987
- Komlos et al., 1993
- Muezzinoglu et al., 2003

- Associative memory problem: store a set of random binary patterns of length n reliably. Later, return the closest stored pattern in response to a noisy query.
- For the past three decades, best neural realizations yield pattern retrieval capacities linear in n.
- For similar structures, we have exponential "pattern retrieval" capacities in coding theory.

Solution Idea

- The reason for the gap? Might be the pure randomness requirement.
- What if we only focus on memorizing structured patterns?
  - Better distant properties.

Solution Idea

- The reason for the gap? Might be the pure randomness requirement.
- What if we only focus on memorizing structured patterns?
  - Better distant properties.

- Successful recent attempts to increasing storage capacities using *structured* patterns.

• C. Berrou, V. Gripon, 2010

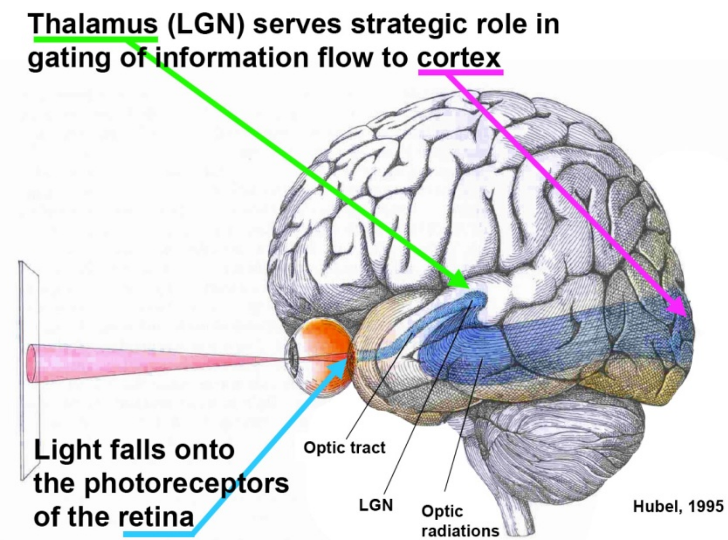
• Salavati, Kumar, Shokrollahi, Gerstner, 2011

Solution Idea

Introduction

- The reason for the gap? Might be the pure randomness requirement.
- What if we only focus on memorizing structured patterns?
  - Better distant properties.

- Successful recent attempts to increasing storage capacities using *structured* patterns.
- It seems biologically relevant as well.



- C. Berrou, V. Gripon, 2010
- Salavati, Kumar, Shokrollahi, Gerstner, 2011

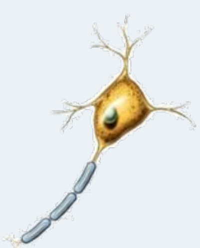
Our Suggested Solution

Introduction

Our Suggested Solution

- In coding theoretical terminology:
  - An algorithm with simple decoding nodes.
  - Message passing over non-binary codes with expander parity check graphs and majority voting broadcast nodes.

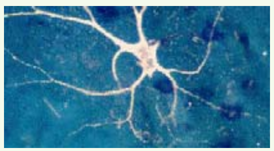
Our Suggested Solution



- In coding theoretical terminology:
  - An algorithm with simple decoding nodes.
  - Message passing over non-binary codes with expander parity check graphs and majority voting broadcast nodes.
- Rephrased in neuroscience parlance:
  - Only store patterns that satisfy some constraints.
  - Constraints from pre-processing stages in brain or outside world.
  - Constraints will help in dealing with noise.

Model and Method

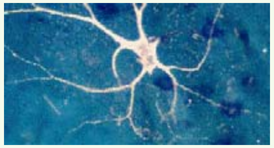
NEURAL NETWORKS

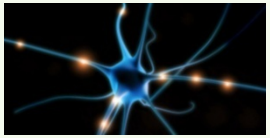


- Neuron: a basic processor in the nervous system.

Solution

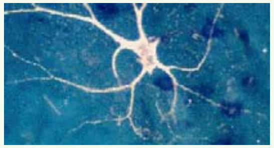
NEURAL NETWORKS

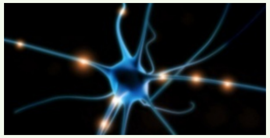




- Neuron: a basic processor in the nervous system.
- Neurons communicate via spikes.

NEURAL NETWORKS





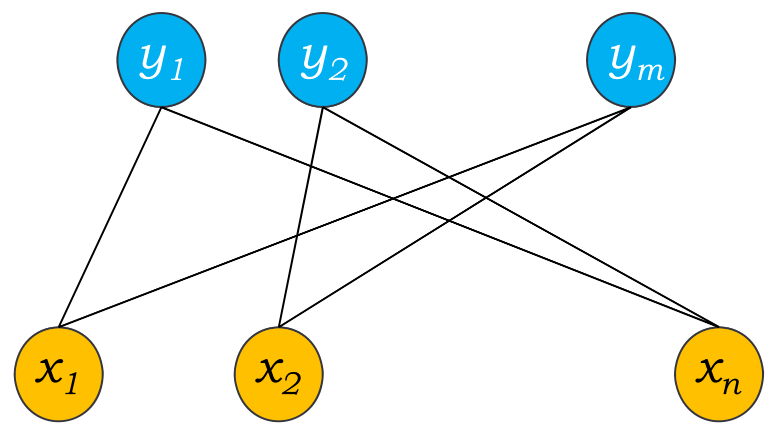
Solution

- Neuron: a basic processor in the nervous system.
- Neurons communicate via spikes.
- Neurons can:
  - Compute a linear sum (count the spikes they receive).
  - Transmit a spike train based on this sum.
  - What they transmit goes to all their neighbors (broadcast system).

MODEL

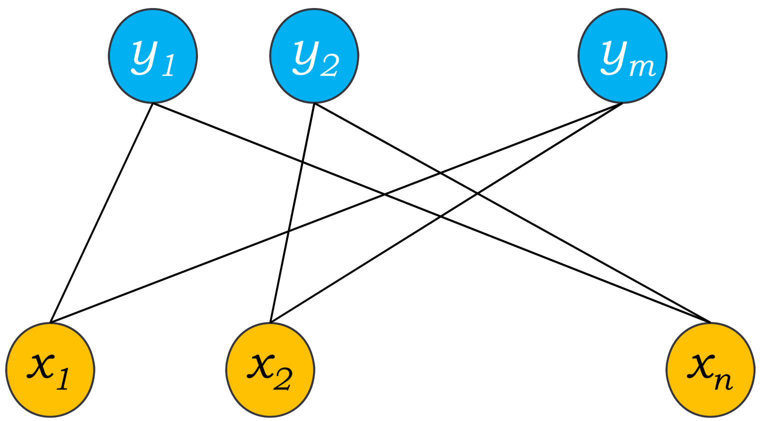
Solution

- A bipartite graph with ***n*** *pattern nodes* and ***m*** *constraint nodes*.
  - Nodes represent neurons.
  - Link weights are 0 or 1.



MODEL

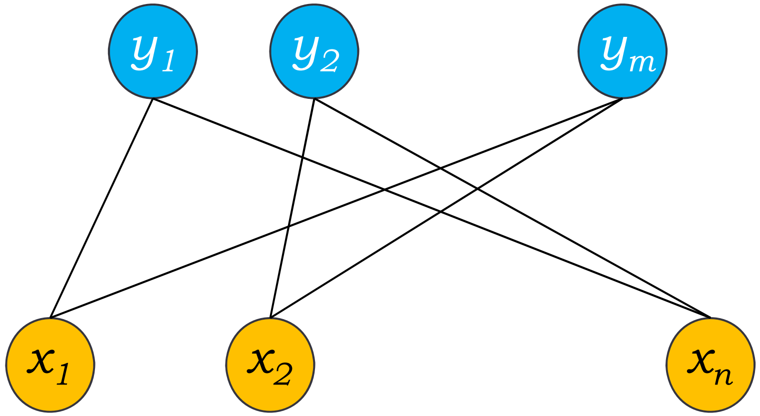
- A bipartite graph with ***n*** *pattern nodes* and ***m*** *constraint nodes*.
  - Nodes represent neurons.
  - Link weights are 0 or 1.
- Broadcast system.



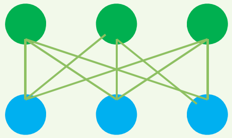
Model

Solution

- A bipartite graph with ***n*** *pattern nodes* and ***m*** *constraint nodes*.
  - Nodes represent neurons.
  - Link weights are 0 or 1.
- Broadcast system.
- *Sparse* and *expander*.

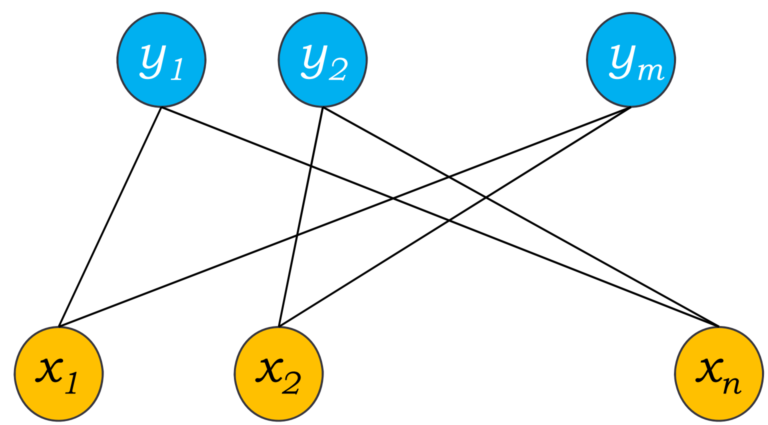


- Sparse: constant number of 1's in a row and column.

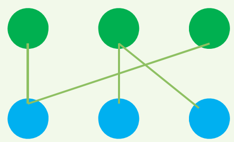


MODEL

- A bipartite graph with ***n*** *pattern nodes* and ***m*** *constraint nodes*.
  - Nodes represent neurons.
  - Link weights are 0 or 1.
- Broadcast system.
- *Sparse* and *expander*.

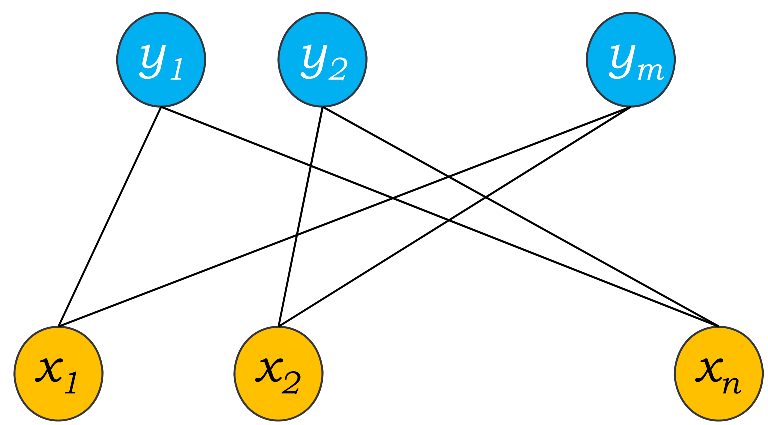


- Sparse: constant number of 1's in a row and column.



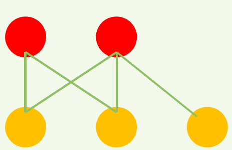
MODEL

- A bipartite graph with ***n*** *pattern nodes* and ***m*** *constraint nodes*.
  - Nodes represent neurons.
  - Link weights are 0 or 1.
- Broadcast system.
- *Sparse* and *expander*.



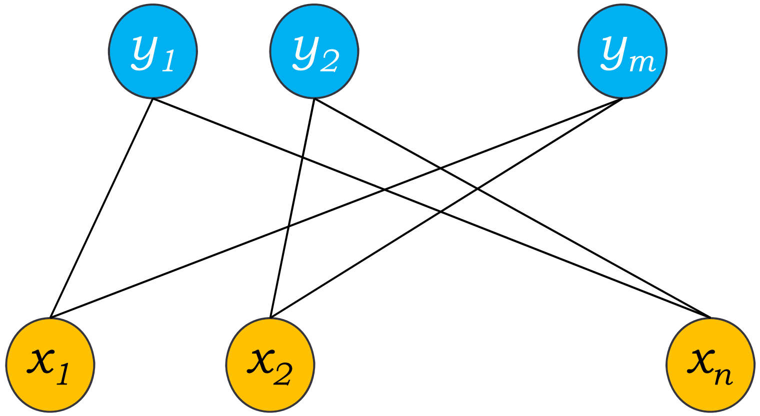
Solution

- Expander: a graph that *expands* well.



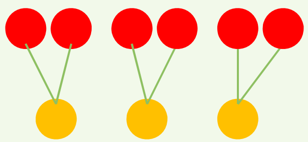
MODEL

- A bipartite graph with ***n*** *pattern nodes* and ***m*** *constraint nodes*.
  - Nodes represent neurons.
  - Link weights are 0 or 1.
- Broadcast system.
- *Sparse* and *expander*.



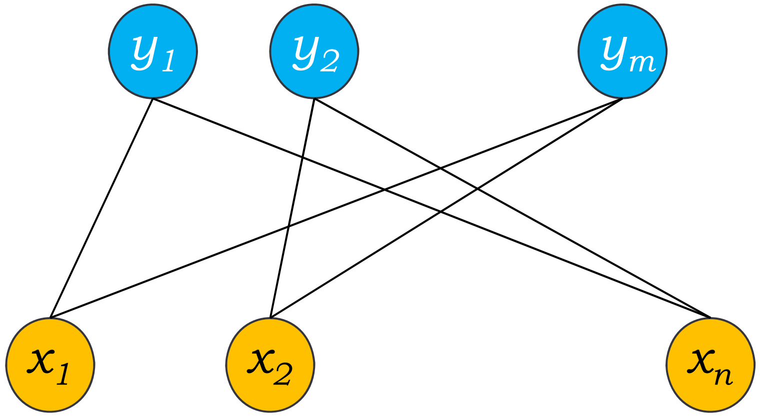
Solution

- Expander: a graph that *expands* well.



MODEL

- A bipartite graph with ***n*** *pattern nodes* and ***m*** *constraint nodes*.
  - Nodes represent neurons.
  - Link weights are 0 or 1.
- Broadcast system.
- *Sparse* and *expander*.
- Non-binary neurons
  - Output of a node is the firing rate of the neuron.



Solution

How It Works

1. Initialization
2. Iterative update
3. Convergence

Solution

- Iterative update of nodes states according to some update rule.

How It Works

1. Initialization
2. Iterative update
3. Convergence

Solution

- Iterative update of nodes states according to some update rule.
- Hopefully, we can correct errors in the input.

How It Works

1. Initialization
2. Iterative update
3. Convergence

Solution

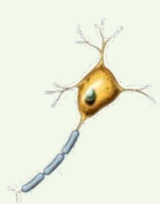
- Iterative update of nodes states according to some update rule.
- Hopefully, we can correct errors in the input.

- The structure is the same as LDPC parity check matrices except for:
  - Real field operations.
  - Simple broadcast nodes.

How It Works

1. Initialization
2. Iterative update
3. Convergence

Solution



- Iterative update of nodes states according to some update rule.
- Hopefully, we can correct errors in the input.
- The structure is the same as LDPC parity check matrices except for:
  - Real field operations.
  - Simple broadcast nodes.
- The structure is similar to a *hetero-associative* memory.
- State of each node = short term firing rate of neurons.

Design Goal

- Given: a set of integer-valued vectors of length *n*.
- Required:
  - These patterns are stable states of the network.
  - Some noise be tolerated.

Solution

Solution

DESIGN GOAL

- Given: a set of integer-valued vectors of length *n*.
- Required:
  - These patterns are stable states of the network.
  - Some noise be tolerated.
- Design parameters:
  - The connectivity matrix H.
  - The nodes/neurons update rule.

DESIGN GOAL

Solution

- Given: a set of integer-valued vectors of length *n*.
- Required:
  - These patterns are stable states of the network.
  - Some noise be tolerated.
- Design parameters:
  - The connectivity matrix H.
  - The nodes/neurons update rule.
- For the moment, we assume H is given and only address the neural update rule.

The Algorithm: Intuition

- We are interested in patterns that satisfy certain number of constraints.
  - Different from the widely used assumption of memorizing any set of purely random patterns.

The Algorithm: Intuition

Solution

- We are interested in patterns that satisfy certain number of constraints.
  - Different from the widely used assumption of memorizing any set of purely random patterns.
- Constraint nodes check for such constraints.
- Given a correct pattern x^{μ}, all constraints are satisfied.
  - Constraint nodes do not fire anything.

The Algorithm: Intuition

Solution

- We are interested in patterns that satisfy certain number of constraints.
  - Different from the widely used assumption of memorizing any set of purely random patterns.

- Constraint nodes check for such constraints.

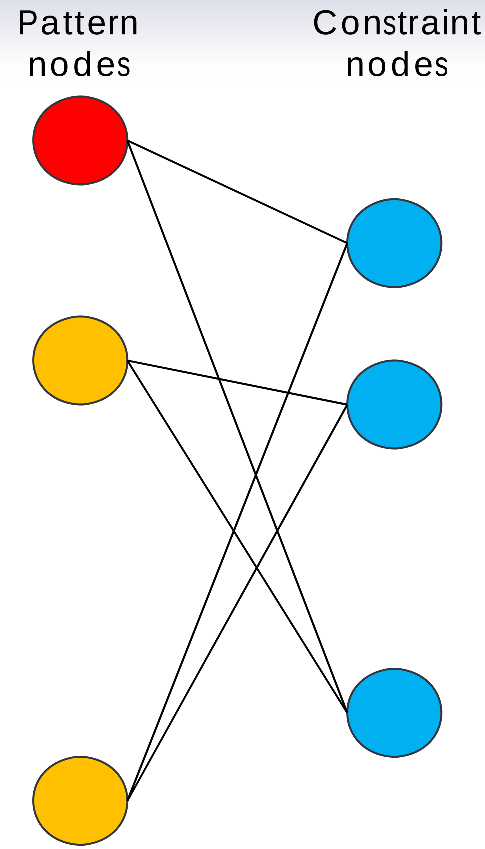
- Given a correct pattern x^{μ}, all constraints are satisfied.
  - Constraint nodes do not fire anything.

Stability

THE ALGORITHM: INTUITION (CONTD.)

Solution

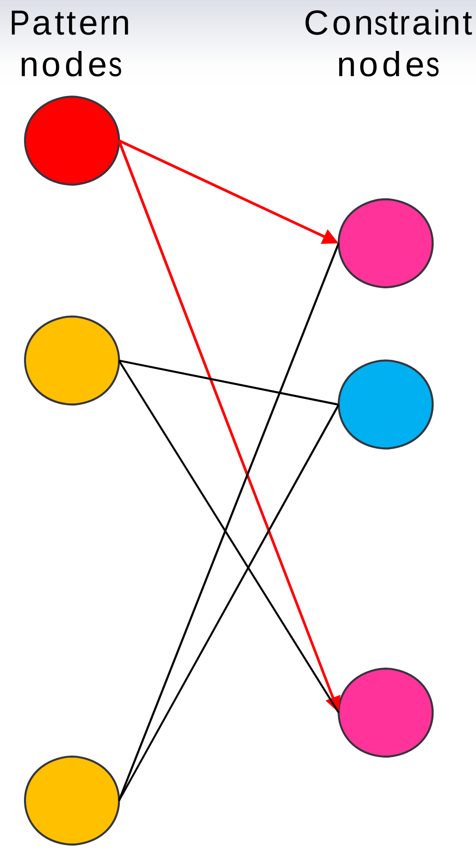
- If the network is given x^{μ} + noise:



THE ALGORITHM: INTUITION (CONTD.)

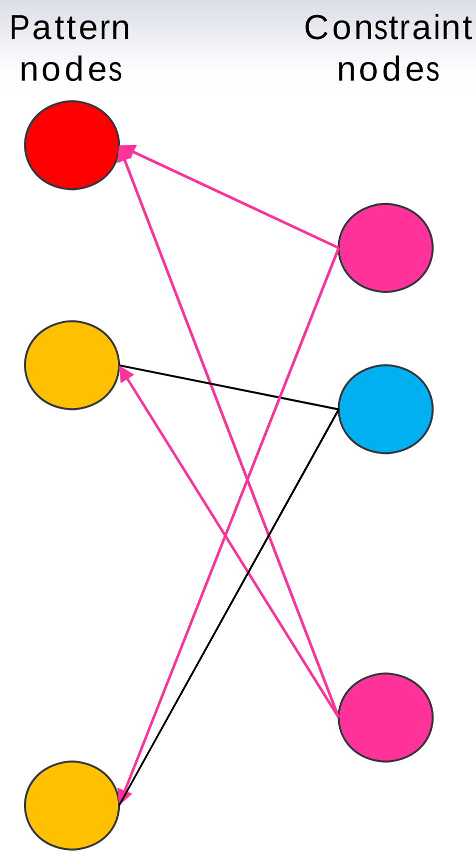
Solution

- If the network is given x^μ + noise:
  1. Some constraints are violated.



The Algorithm: Intuition (contd.)

- If the network is given x^{μ} + noise:
  1. Some constraints are violated.
  2. They send some feedback to their neighbors.

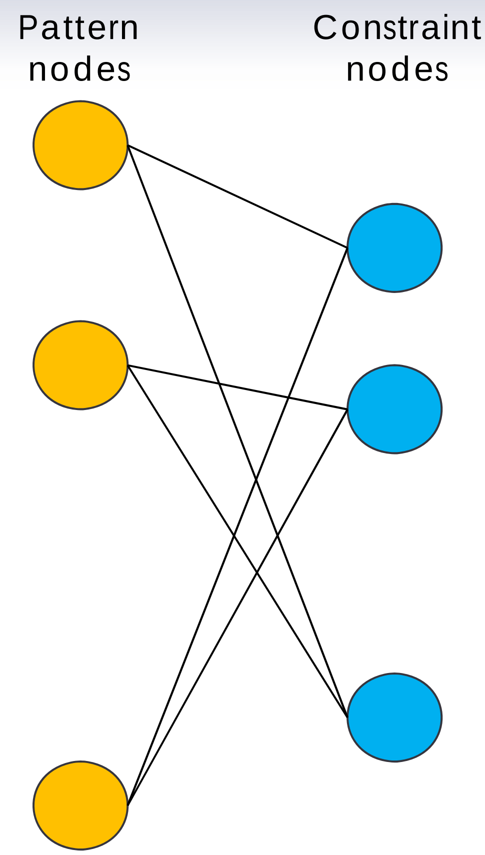


Solution

The Algorithm: Intuition (contd.)

Solution

- If the network is given x^{μ} + noise:
  1. Some constraints are violated.
  2. They send some feedback to their neighbors.
  3. Based on the number of constraint nodes they receive feedback from, pattern nodes update their value.

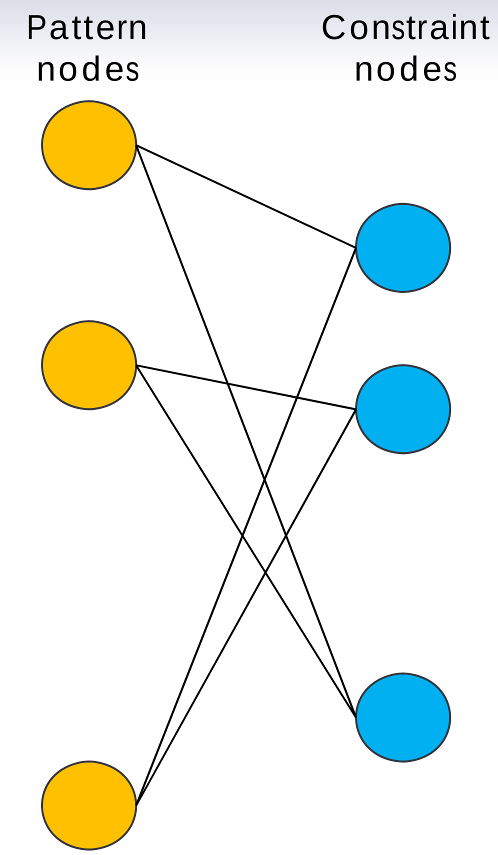


THE ALGORITHM: INTUITION (CONTD.)

Solution

Noise tolerance

- If the network is given x^{μ} + noise:
  1. Some constraints are violated.
  2. They send some feedback to their neighbors.
  3. Based on the number of constraint nodes they receive feedback from, pattern nodes update their value.



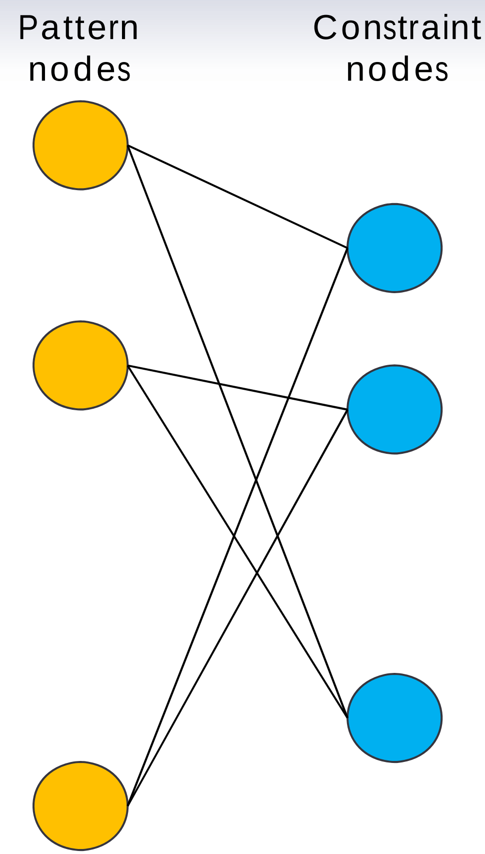
The Algorithm: Intuition (contd.)

Solution

Similar in nature to Sipser & Spielsman's expander codes.

Noise tolerance ✓

- If the network is given x^{μ} + noise:
  1. Some constraints are violated.
  2. They send some feedback to their neighbors.
  3. Based on the number of constraint nodes they receive feedback from, pattern nodes update their value.



THE ALGORITHM: DETAILS

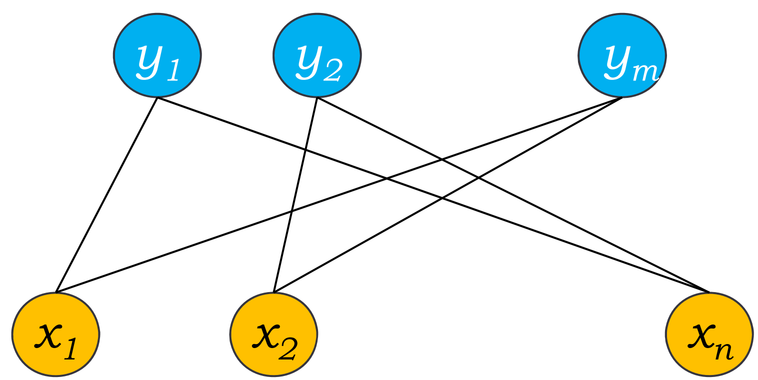
Solution

H: m×n 0/1 matrix.

x: n×1 integer vector.

b: m×1 integer vector.

- The constraint satisfaction problem: $Hx=b$



The Algorithm: Details

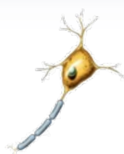
H: m×n 0/1 matrix.

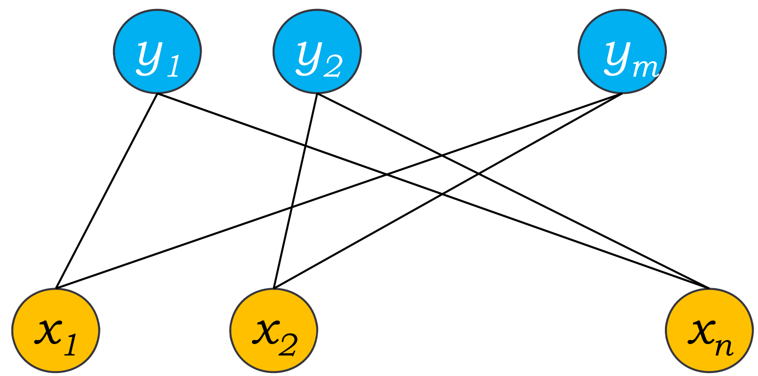
x: n×1 integer vector.

b: m×1 integer vector.

Solution

- The constraint satisfaction problem: Hx=b
  - Firing rate of a number of pattern nodes should always add up to some value.





The Algorithm: Details

Solution

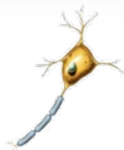
H: $m \times n$ 0/1 matrix.

x: $n \times 1$ integer vector.

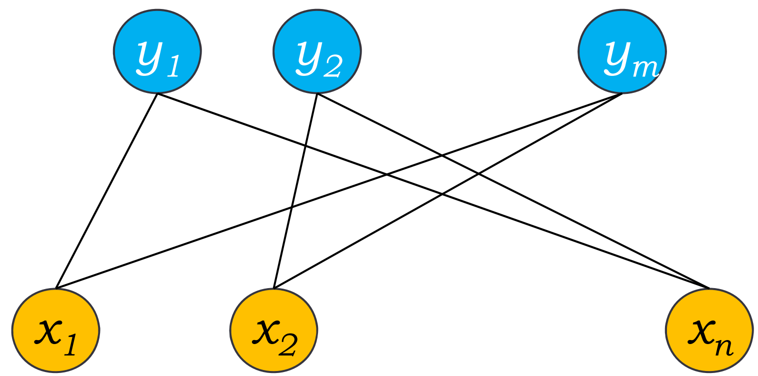
b: $m \times 1$ integer vector.

$h_i = \Sigma\, H_{ij}\, x_j$

- The constraint satisfaction problem: $Hx=b$



  - Firing rate of a number of pattern nodes should always add up to some value.



- Constraint nodes:

$$y_i = \begin{cases} 1, & h_i < b_i \\ 0, & h_i = b_i \\ -1, & \text{otherwise} \end{cases}$$

The Algorithm: Details

Solution

H: m×n 0/1 matrix.

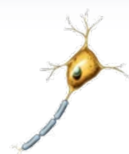
x: n×1 integer vector.

b: m×1 integer vector.

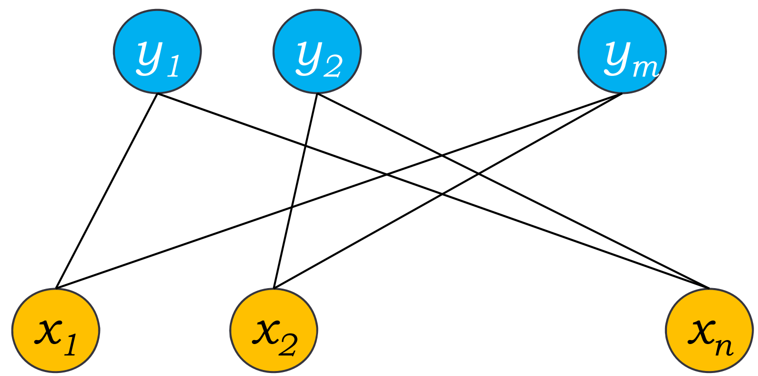
$h_i = \Sigma\, H_{ij}\, x_j$

$d_p = \deg(x_j)$

- The constraint satisfaction problem: $Hx{=}b$



  - Firing rate of a number of pattern nodes should always add up to some value.



- Constraint nodes:

$$y_i = \begin{cases} 1, & h_i < b_i \\ 0, & h_i = b_i \\ -1, & \text{otherwise} \end{cases}$$

- Pattern nodes:

$$g_j = \frac{\sum_{i=1}^{m} H_{ij} y_i}{d_p}$$

The Algorithm: Details

H: m×n 0/1 matrix.

x: n×1 integer vector.

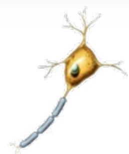
b: m×1 integer vector.

Solution

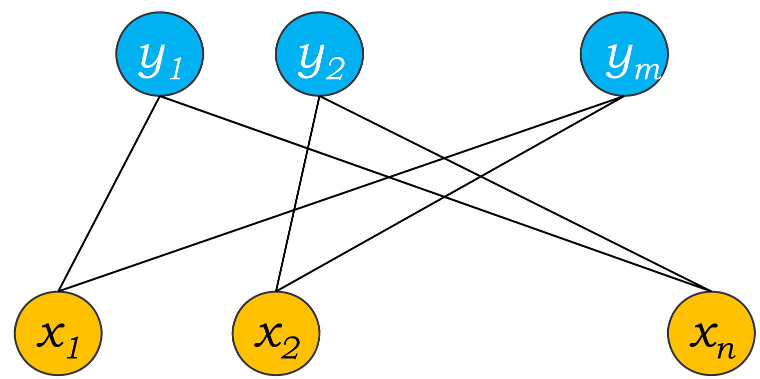
$h_i = \Sigma\, H_{ij}\, x_j$

$d_p = \deg(x_j)$

- The constraint satisfaction problem: $Hx=b$



  - Firing rate of a number of pattern nodes should always add up to some value.



- Constraint nodes:

$$y_i = \begin{cases} 1, & h_i < b_i \\ 0, & h_i = b_i \\ -1, & \text{otherwise} \end{cases}$$

- Pattern nodes:

$$g_j = \frac{\sum_{i=1}^{m} H_{ij} y_i}{d_p}$$

  - Winner-take-all
  - Bit-flip

THE ALGORITHM: DETAILS (CONTD.)

Solution

The Algorithm: Details (contd.)

$\mathrm{H}x^{\mu} = b$

$h_i = \Sigma\ \mathrm{H}_{ij}\ x_j$

Solution

- Noise free case: If x^{μ} is given, no constraint node fires.

$$y_i = \begin{cases} 1, & h_i < b_i \\ 0, & h_i = b_i \\ -1, & \text{otherwise} \end{cases}$$

The Algorithm: Details (contd.)

$Hx^{\mu} = b$

$h_i = \Sigma H_{ij} x_j$

Solution

- Noise free case: If x^{μ} is given, no constraint node fires.

$$y_i = \begin{cases} 1, & h_i < b_i \\ 0, & h_i = b_i \\ -1, & \text{otherwise} \end{cases}$$

- If x^{μ} + noise is given:

$$g_j = \frac{\sum_{i=1}^{m} H_{ij} y_i}{d_p}$$

  - g_j is larger for corrupted nodes.
  - If the graph is expander, the update will reduce noise.
  - Firing rates are saturated to make this happen.

RESULTS

- The pattern retrieval capacity:

$$S^n / (d_c S)^m$$

  - S: the maximum firing rate.
  - d_c: deg(constraint nodes)

Solution

RESULTS

Solution

- The pattern retrieval capacity:

$$S^n / (d_c S)^m$$

  - S: the maximum firing rate.
  - d_c: deg(constraint nodes)
  - Is exponential if: $\log(S) > m \log(d_c)/(n-m)$

RESULTS

Solution

- The pattern retrieval capacity:

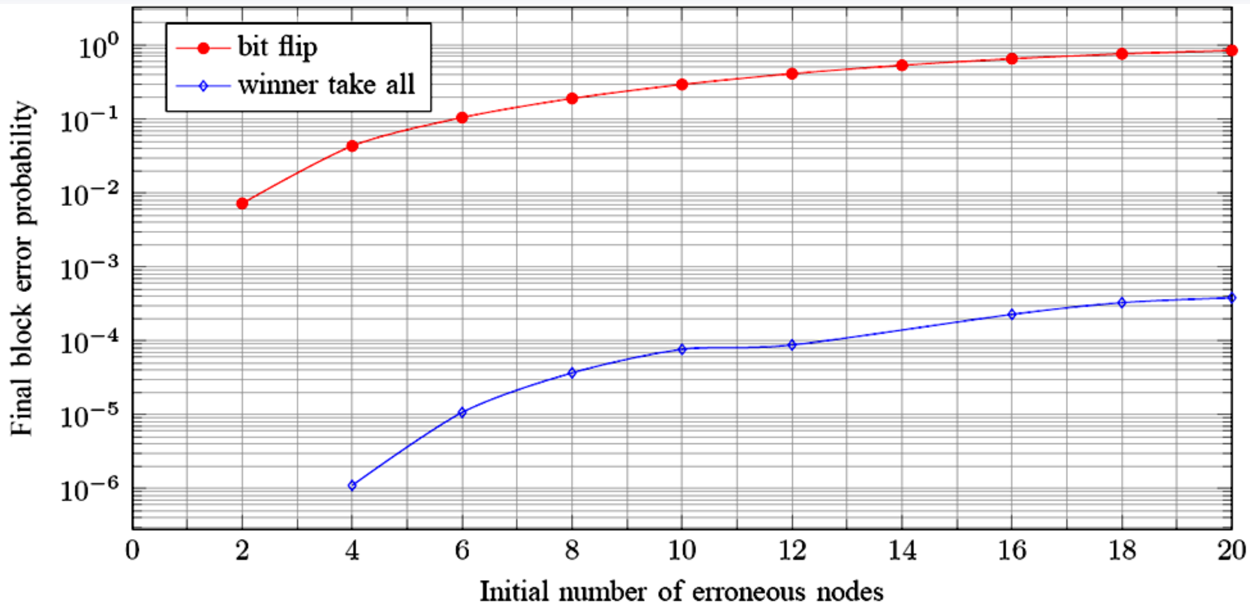
$$S^n/(d_c S)^m$$

  - S: the maximum firing rate.
  - d_c: deg(constraint nodes)
  - Is exponential if: $\log(S) > m\log(d_c)/(n-m)$
- **Theorem**: If H is an expander graph, the *winner-take-all* and *bit-flipping** algorithms are guaranteed to correct two erroneous nodes.

RESULTS (CONTD.)

Solution

- For more than two errors:



- $n = 600$, $m = 300$, $d_c = 10$, $d_p = 5$
- Noise model: integer values in [-5,5]

Ongoing Work

What Happens to H?

- So far, H was assumed to be given.
- In order to find H, there are two possible approaches:

Discussion

WHAT HAPPENS TO H?

- So far, H was assumed to be given.
- In order to find H, there are two possible approaches:

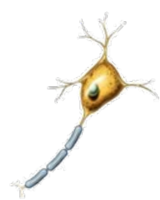
1. Infer H from data

Discussion

WHAT HAPPENS TO H?

- So far, H was assumed to be given.
- In order to find H, there are two possible approaches:

1. Infer H from data

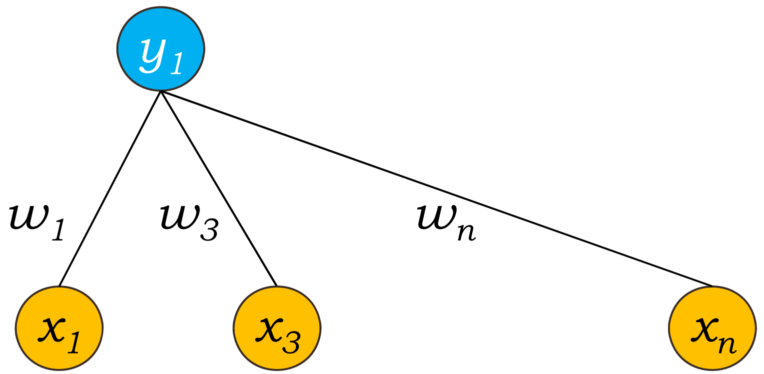


   - Back propagation does not work: H is not guaranteed to be an expander.
   - Low activity assumption might help us here.
   - If data is sparse itself, we might be able to use a similar structure to learn from data

Discussion

Inferring H from Data

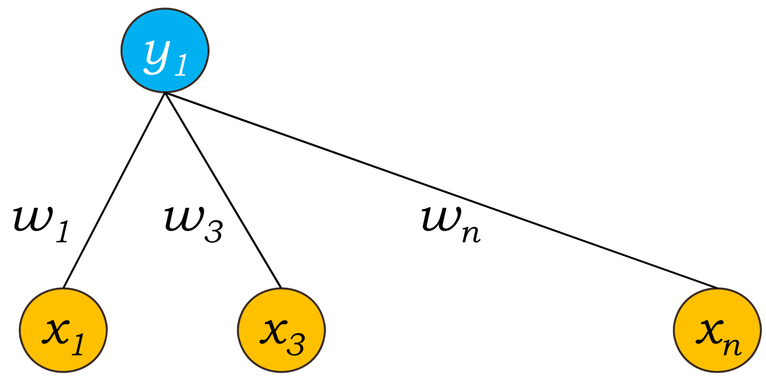
- Consider the neighborhood of constraint node one.



Discussion

INFERRING H FROM DATA

- Consider the neighborhood of constraint node one.

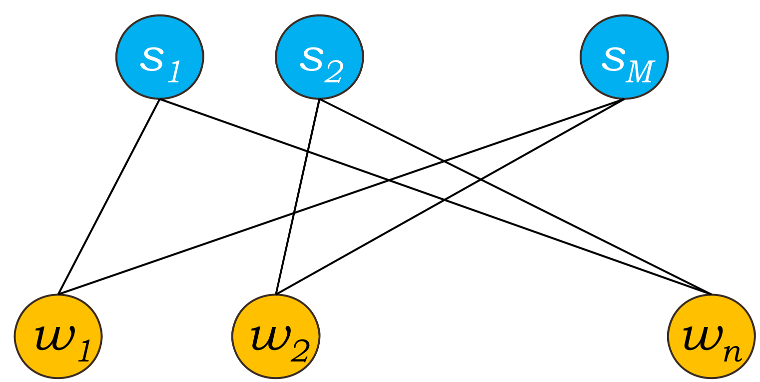


- If the weights are correct, y_1 will be zero <u>for all patterns</u>.
- If not, let of s_μ be the output of y_1 for pattern μ.

Discussion

INFERRING H FROM DATA (CONTD.)

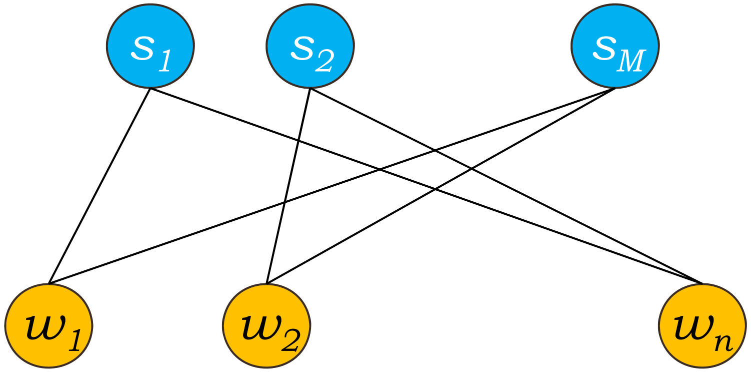
- Consider the following structure:
  - Link weights are patterns.



Discussion

INFERRING H FROM DATA (CONTD.)

- Consider the following structure:
  - Link weights are patterns.
- If $w_1,\ldots, w_n$ are correct, $s_1,\ldots,s_M$ will all be zero.
- If not, we have the same problem as before.

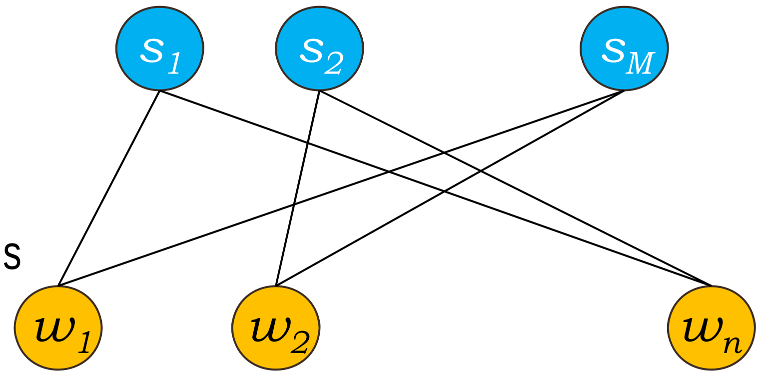


Discussion

Inferring H from Data (contd.)

- Consider the following structure:
  - Link weights are patterns.
- If $w_1, \ldots, w_n$ are correct, $s_1, \ldots, s_M$ will all be zero.
- If not, we have the same problem as before.

- If we have low activity patterns, this graph is also sparse!
- We might apply the previous algorithm for the learning phase as well!



Discussion

WHAT HAPPENS TO H? (CONTD.)

2. Map a set of random inputs to a higher dimension:

What Happens to H? (Contd.)

2. Map a set of random inputs to a higher dimension:

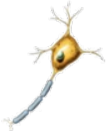
- Similar to encoding problem: Find a generator matrix G that HG = 0.
- But G must be sparse as well!

Discussion

WHAT HAPPENS TO H? (CONTD.)

2. Map a set of random inputs to a higher dimension:

- Similar to encoding problem: Find a generator matrix G that HG = 0.
- But G must be sparse as well!

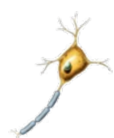


- This corresponds to pre-processing stages in the brain.

WHAT HAPPENS TO H? (CONTD.)

2. Map a set of random inputs to a higher dimension:

   - Similar to encoding problem: Find a generator matrix G that HG = 0.
   - But G must be sparse as well!

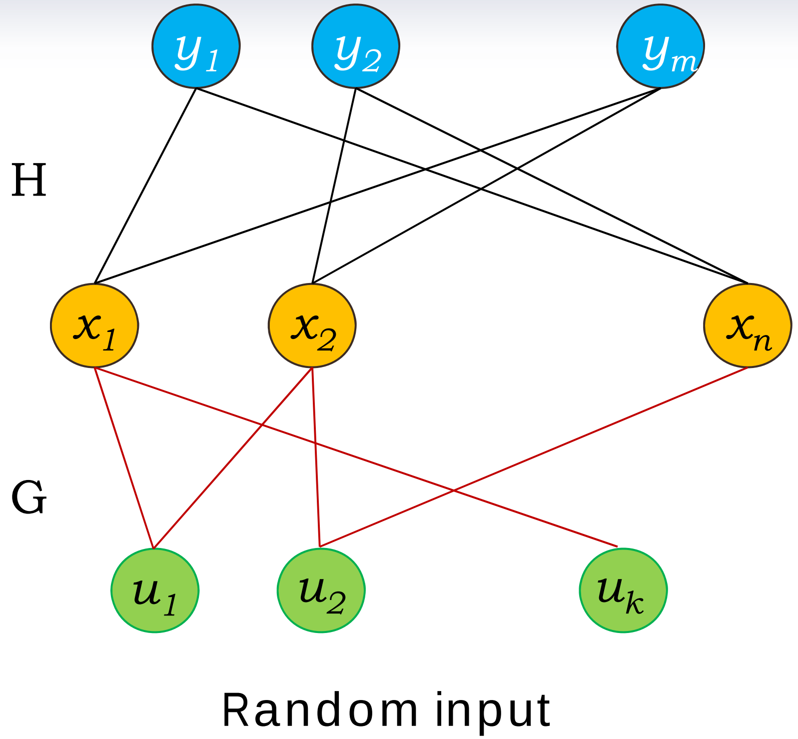


   - This corresponds to pre-processing stages in the brain.

- For the second approach, noise model must be modified.

What Happens to H? (Contd.)

- $u_1,\ldots,u_k$ can be purely random.
- $Gu^\mu = x^\mu$
- $HG = 0$
- G Should be sparse as well.



Discussion

Conclusions and Final Remarks

- An associative memory with exponential capacity is proposed.
  - All that is needed is a two-layer neural network which is also an expander.
  - Simple update rules

Conclusions and Final Remarks

- An associative memory with exponential capacity is proposed.
  - All that is needed is a two-layer neural network which is also an expander.
  - Simple update rules
- We are now working on:
  - Inferring H from data or
  - Mapping input to a higher dimension.

Thanks For Your Attention

Any Questions?