ONLY GOD

1402/ 2023

Neural Network & Deep Learning

Single-Layer Feedforward Networks for Association

CSE & IT Department

School of ECE

Shiraz University

Biological Memory

mansoori@shirazu.ac.ir

- In neurobiological context
  - Memory refers to relatively enduring neural alterations induced by interactions of organism with its environment
  - Has capabilities of storing and retrieving data patterns
- Biological memory
  - Is physically distributed
  - Operates by association , rather than addressing
  - Content-addressable memory, no address-addressable memory

Pattern Association

mansoori@shirazu.ac.ir

- Learning is process of forming association between related patterns
- Memorization of a pattern is an association of a pattern with itself
- An input stimulus (a sensory cue) similar to associated one will invoke associated pattern
- Example:
  name of a cartoon ---> recalls TV series about it
  Picture of a place ---> recalls memories of people are met
  Recognizing a person with unlimited appearances
- There is some underlying collection of stored data which is ordered and interrelated in some way
- This data constitute a stored pattern or memory

Pattern Association

mansoori@shirazu.ac.ir

- In recalling:
  - When part of pattern of data is presented in form of sensory cue, rest of pattern (memory) is recalled or associated with it
  - When an imperfect version of stored memory is offered, true and uncorrupted pattern should be associated with it
- A very limited form of association in conventional computers:
  - Key (index) term is a sensory cue when searching a database

AM

Associative Memory (AM)

mansoori@shirazu.ac.ir

- Associative memory
  - Content-addressable memory
  - Memory that operates by association
  - Mapping of an input to an output (stimulus to output)
  - Example: face recognition

- Characteristics of associative memory
  - Distributed
  - Both key (stimulus) and output (stored pattern) are data vectors (as opposed to address and data)
  - Storage pattern different from key and is spatially distributed
  - Stimulus is address as well as stored pattern (with imperfections)
  - Tolerant to noise and damage

AM Net

mansoori@shirazu.ac.ir

Architecture of an AM net:

- Feed-forward:
  - Can store pattern associations in weights of net and can retrieve them
- Recurrent:
  - Can store patterns as stable states of a dynamic physical system with minimum energy

Feed-forward AM Net

mansoori@shirazu.ac.ir

- A highly simplified model of human memory
- Single-layer net:
  - Can store pattern associations in weights through training
  - Can recall known patterns by providing imperfect or incomplete form of them
- Distributed memory: simultaneous activities of many neurons that contain information about external stimuli
- Each association is an input-output vector pair $< \vec{s}:\vec{t} >$

  $\vec{s}$: key pattern, $\vec{t}$: stored or memorized pattern
- All associations $< S:T > = < [\vec{s}(1) \ \dots \ \vec{s}(P)]:[\vec{t}(1) \ \dots \ \vec{t}(P)] >$
- Bipolar representation is more powerful than binary in pattern association

Feed-forward AM Net

mansoori@shirazu.ac.ir

- Distributed memory mapping: each pair $< \vec{s}:\vec{t} >$ transforms activity pattern $\vec{s}$ in input space to activity pattern $\vec{t}$ in output space

- Types of AMs:
  - Auto-associative (AAM)
    - A key vector is associated with itself in memory
    - Dimension of input and output are same
    - $\vec{t} = \vec{s}$ in each association pair $< \vec{s}:\vec{t} >$
  - Hetro-associative (HAM)
    - A key vector is associated with another memorized vector
    - Dimension of input and output may or may not be same
    - $\vec{t} \neq \vec{s}$ in each association pair $< \vec{s}:\vec{t} >$

AM Training

mansoori@shirazu.ac.ir

- Training methods:
  - Hebb rule: for threshold activation functions
  - Delta rule: for differentiable activation functions

- AM capacity of a net:
  - How many patterns can be stored before net starts to forget learned patterns
  - Some factors influence it (as in human memory)
    - Complexity of patterns (number of components of $\vec{s}, \vec{t}$)
    - Similarity of input patterns that are associated with significantly different response patterns

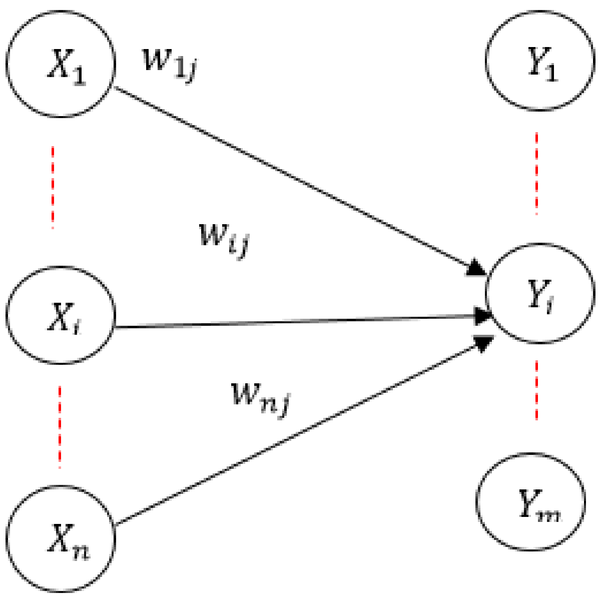
Hebb Rule in AM Net

mansoori@shirazu.ac.ir

Hebb rule for pattern association

- The simplest and most common method for determining weights of an associative memory NN
- A non-iterative learning method
- Can be used with patterns in binary or bipolar representation

$$\Delta w_{ij} = x_i \, y_j$$



Hebb Rule in AM Net

mansoori@shirazu.ac.ir

Algorithm:

1. Initialize all weights

$$w_{ij} = 0 \quad (i = 1, \dots, n; j = 1, \dots, m)$$

2. For each training input-output vector pair

$$< \vec{s} : \vec{t} > = < \vec{s}(p) : \vec{t}(p) >$$

2.1. Set activation for input and output units

$$x_i = s_i \quad (i = 1, \dots, n), \quad y_j = t_j \quad (j = 1, \dots, m)$$

2.2. Adjust the weights

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + x_i\, y_j \;,\; (i = 1, \dots, n \;;\; j = 1, \dots, m)$$

3. Stop

Hebb Rule in AM Net

mansoori@shirazu.ac.ir

- The weights found by Hebb rule (with zero initials) can be obtained by outer product of association vector pair $< \vec{s}:\vec{t} >$

$$\vec{s} = \begin{bmatrix} s_1 \\ \vdots \\ s_i \\ \vdots \\ s_n \end{bmatrix}, \vec{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_j \\ \vdots \\ t_m \end{bmatrix} \Rightarrow \vec{s}\,\vec{t}^T = \begin{bmatrix} s_1t_1 & \cdots & s_1t_j & \cdots & s_1t_m \\ \vdots & & \vdots & & \vdots \\ s_it_1 & \cdots & s_it_j & \cdots & s_it_m \\ \vdots & & \vdots & & \vdots \\ s_nt_1 & \cdots & s_nt_j & \cdots & s_nt_m \end{bmatrix} = W = \{w_{ij}\},\ w_{ij} = s_it_j$$

W: correlation memory matrix for association pair $< \vec{s}:\vec{t} >$

- Using all associations $< S:T >= \{< \vec{s}(p):\vec{t}(p) > \ ,\quad p = 1, \dots, P\}$

$$W = \{w_{ij}\},\ \ w_{ij} = \sum_{p=1}^{P} s_i(p)\, t_j(p) \quad \Longrightarrow \quad W = \sum_{p=1}^{P} \vec{s}(p)\ \vec{t}(p)^T = ST^T$$

W: correlation memory matrix for all association pairs

Hebb Rule in AM Net

mansoori@shirazu.ac.ir

- Example:

$$\vec{s}(1) = \begin{bmatrix} -1 \\ 1 \\ 1 \\ 1 \end{bmatrix}, \quad \vec{s}(2) = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}, \quad \vec{s}(3) = \begin{bmatrix} 1 \\ 1 \\ -1 \\ 1 \end{bmatrix}, \quad \vec{s}(4) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ -1 \end{bmatrix}$$

$$\vec{t}(1) = \begin{bmatrix} -1 \\ -1 \end{bmatrix}, \quad \vec{t}(2) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \quad \vec{t}(3) = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \quad \vec{t}(4) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$W(1) = \begin{bmatrix} 1 & 1 \\ -1 & -1 \\ -1 & -1 \\ -1 & -1 \end{bmatrix}, W(2) = \begin{bmatrix} -1 & 1 \\ 1 & -1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}, W(3) = \begin{bmatrix} 1 & -1 \\ 1 & -1 \\ -1 & 1 \\ 1 & -1 \end{bmatrix}, W(4) = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \\ -1 & -1 \end{bmatrix},$$

$$\Rightarrow \quad W = \sum_{p=1}^{4} W(p) = \begin{bmatrix} 2 & 2 \\ 2 & -2 \\ -2 & 2 \end{bmatrix}$$

Hebb Rule in AM Net

mansoori@shirazu.ac.ir

- Suitability of Hebb rule depends on correlation among input training vectors
  - For uncorrelated (orthogonal) input vectors (linearly independent), Hebb rule will produce correct weights and can recall all stored patterns perfectly
    - Two orthogonal vectors have zero dot product
  - For non-orthogonal input vectors, response will include a portion of each of target values (cross talk)
- Correlation memory matrix (weights W) has no mechanism for feedback since it is based on Hebbian learning
- Consequently, there are errors in recall
- How continually update W to reduce error?

Delta Rule in AM Net

mansoori@shirazu.ac.ir

Delta rule for pattern association:

- Impose error-correction learning (delta rule) to update W
  - An iterative learning method
  - Can be used for input (key) vectors that are linearly independent but not orthogonal
  - Can produce least square solution when input patterns are not linearly independent

$$y\_in_j = b_j + \sum_{i=1}^{n} x_i w_{ij} = b_j + \vec{w}_{.j}{}^T \vec{x}$$

$$y_j = f(y\_in_j) \; , \qquad (j = 1, \ldots, m)$$

$$w_{ij}(\text{new}) = w_{ij}(\text{old}) + \alpha(t_j - y_j)\, f'(y\_in_j)\, x_i$$

Recalling in HAM Net

mansoori@shirazu.ac.ir

How memory is addressed and how stored information is recalled?

- To recall any pattern $\vec{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_i \\ \vdots \\ x_n \end{bmatrix}$, compute $\vec{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_j \\ \vdots \\ y_m \end{bmatrix}$ as:

$$y\_in_j = \sum_{i=1}^{n} x_i w_{ij} \;, \quad y_j = f(y\_in_j) = sgn(y\_in_j) = \begin{cases} 1 \;, & \text{if } \; y\_in_j > 0 \\ 0 \;, & \text{if } \; y\_in_j = 0 \\ -1 \;, & \text{if } \; y\_in_j < 0 \end{cases}$$

- Other association functions:
  - In nets using Hebb rule: $f(x) = \begin{cases} 1 \;, & \text{if } \; x > \theta \\ \theta \;, & \text{if } \; x = \theta \\ -1 \;, & \text{if } \; x < \theta \end{cases}$
  - In nets using delta rule: $f(x) = \frac{1-e^{-x}}{1+e^{-x}}$
- Using product operation to recall a vector

$$y\_in_j = \sum_{i=1}^{n} x_i w_{ij} \; = \; \vec{w}_{.j}{}^{T} \vec{x} \Rightarrow \overrightarrow{y\_in} \; = W^T \vec{x} \Rightarrow \vec{y} = \; f(W^T \vec{x})$$

Recalling in HAM Net

mansoori@shirazu.ac.ir

Example:

$$< \vec{s}:\vec{t} >= \{< -1,1,1,1: -1, -1 >, < 1, -1,1,1: -1,1 >,$$
$$< 1,1,-1,1: 1,-1 >, < 1,1,1,-1: 1,1 >\}$$

- Since stored patterns are orthogonal, all patterns can be recalled perfectly

$$W = \begin{bmatrix} 2 & 2 \\ 2 & -2 \\ -2 & 2 \\ -2 & -2 \end{bmatrix}, \quad \vec{x} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \Longrightarrow \overrightarrow{y\_in} = W^T\vec{x} = \begin{bmatrix} 2 & 2 & -2 & -2 \\ 2 & -2 & 2 & -2 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} -4 \\ 4 \end{bmatrix} \Longrightarrow \vec{y} = f(\overrightarrow{y\_in}) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

Recalling in HAM Net

mansoori@shirazu.ac.ir

Example:

$$W = \begin{bmatrix} 2 & 2 \\ 2 & -2 \\ -2 & 2 \\ -2 & -2 \end{bmatrix}$$

- A pattern with one missed component can be recalled

$$\vec{x} = \begin{bmatrix} 1 \\ -1 \\ 0 \\ 1 \end{bmatrix} \quad \Longrightarrow \overrightarrow{y\_in} = W^T\vec{x} = \begin{bmatrix} -2 \\ 2 \end{bmatrix} \quad \Longrightarrow \vec{y} = f(\overrightarrow{y\_in}) = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$$

- A pattern with one mistaken component cannot be recalled

$$\vec{x} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ -1 \end{bmatrix} \quad \Longrightarrow \overrightarrow{y\_in} = W^T\vec{x} = \begin{bmatrix} 0 \\ 8 \end{bmatrix} \quad \Longrightarrow \vec{y} = f(\overrightarrow{y\_in}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \text{ (cross talk)}$$

AAM

AAM Net

mansoori@shirazu.ac.ir

- Is a special case of HAM NN where $\vec{t} = \vec{s}$ for all training pairs $< \vec{s}:\vec{t} >$
- Can be used to determine whether an input vector is known or unknown to net
- Net recognizes a known vector by producing that pattern on output units if is given as input
- Training vectors stores associations in weights
- A stored vector can be retrieved from distorted (noisy) or partial input (to show its generalization)

Weights in AAM Net

mansoori@shirazu.ac.ir

- To store a single bipolar vector $\vec{x} = \vec{s}(1)$, choose weight matrix $W(1)$ as (using Hebb rule):

$$W(1) = \vec{x}\,\vec{x}^T - I \implies W(1) = W(1)^T$$

- To prove $\vec{x} = \vec{s}(1)$ is known vector for net:

$$\overrightarrow{y\_in} = W(1)^T\vec{x} = (\vec{x}\,\vec{x}^T - I)\,\vec{x} = \vec{x}\,\vec{x}^T\vec{x} - I\vec{x} = n\vec{x} - \vec{x} = (n-1)\,\vec{x}$$

$$\implies \vec{y} = sgn\left(\overrightarrow{y\_in}\right) = sgn\left((n-1)\,\vec{x}\right) = sgn(\vec{x}) = \vec{x}$$

- For P mutually orthogonal vectors in S, weights can also be set:

$$W = \sum_{p=1}^{P} W(p)$$

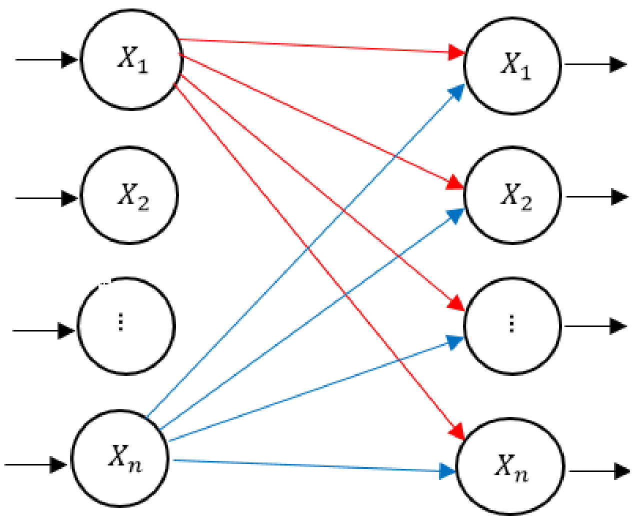
Or

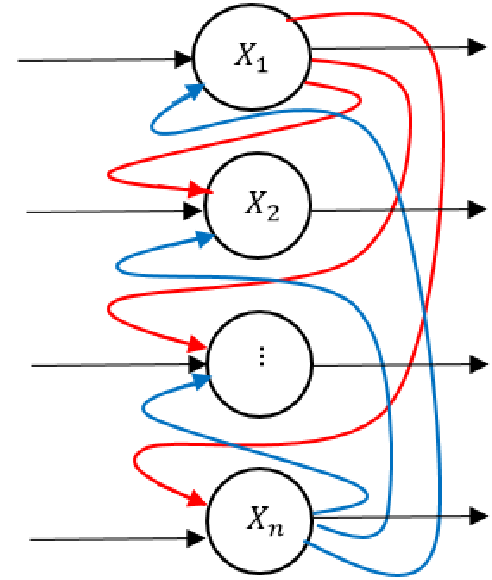
$$W = \sum_{p=1}^{P} \vec{s}(p)\,\vec{s}(p)^T = S\,S^T$$, then set diagonal weights to zero

W: correlation memory matrix

AAM Net

mansoori @shirazu.ac.ir





- Weights on diagonal of W are set to zero:
  - To improve generalization ability of net
  - To increase biological plausibility of net
  - Necessary for iterative AM NNs
  - Necessary if delta rule is used for training

AAM Net

mansoori@shirazu.ac.ir

Example: $\vec{s} = \{< -1,1,1,1 >, < 1,-1,1,1 >, < 1,1,-1,1 >\}$

$$W = \begin{bmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \\ -1 & 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 1 & 1 \\ -1 & 1 & -1 & -1 \\ 1 & -1 & 1 & 1 \\ 1 & -1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 1 & -1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & 1 & -1 \\ 1 & 1 & -1 & 1 \end{bmatrix} = \begin{bmatrix} 3 & -1 & -1 & 1 \\ -1 & 3 & -1 & 1 \\ -1 & -1 & 3 & 1 \\ 1 & 1 & 1 & 3 \end{bmatrix}$$

$$\Rightarrow \quad W' = \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & -1 & 1 \\ -1 & -1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

- Recalling a stored pattern:

$$\vec{x} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \Rightarrow \begin{cases} \overrightarrow{y\_in} = W^T \vec{x} = \begin{bmatrix} 4 \\ -4 \\ 4 \\ 4 \end{bmatrix} \Rightarrow \vec{y} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \\ \overrightarrow{y\_in} = W'^T \vec{x} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \Rightarrow \vec{y} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \end{cases}$$

AAM Net

mansoori@shirazu.ac.ir

Example: $\vec{s} = \{< -1, 1, 1, 1 >, < 1, -1, 1, 1 >, < 1, 1, -1, 1 >\}$

$$W = \begin{bmatrix} 3 & -1 & -1 & 1 \\ -1 & 3 & -1 & 1 \\ -1 & -1 & 3 & 1 \\ 1 & 1 & 1 & 3 \end{bmatrix} \Longrightarrow W' = \begin{bmatrix} 0 & -1 & -1 & 1 \\ -1 & 0 & -1 & 1 \\ -1 & -1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$$

- Recalling a pattern with one missed component

$$\vec{x} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 0 \end{bmatrix} \Longrightarrow \begin{cases} \overrightarrow{y\_in} = W^T \vec{x} = \begin{bmatrix} 3 \\ -5 \\ 3 \\ 1 \end{bmatrix} \Longrightarrow \vec{y} = \begin{bmatrix} 1 \\ -1 \\ 1 \\ 1 \end{bmatrix} \\ \overrightarrow{y\_in} = W'^T \vec{x} = \begin{bmatrix} 0 \\ -2 \\ 0 \\ 1 \end{bmatrix} \Longrightarrow \vec{y} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ 1 \end{bmatrix} \end{cases}$$

AAM Net

mansoori@shirazu.ac.ir

Storage capacity of AAM nets

- Input (key) vectors $S = [\vec{s}(1) \ \dots \ \vec{s}(P)]$ are orthonormal if

$$\vec{s}(i)^T \vec{s}(j) = \begin{cases} 1\,, & \text{if } i = j \\ 0\,, & \text{if } i \neq j \end{cases}$$

- Number of patterns stored (storage capacity) by AAM is
    - n if S is orthonormal
    - rank of correlation memory matrix if S isnot orthonormal
        - $\mathrm{rank}(W) < n$ where n is input dimension
- So, correlation memory matrix can reliably store a maximum of n patterns

AAM Net

mansoori@shirazu.ac.ir

Storage capacity of AAM nets:

- Real-world patterns are not orthonormal
- If key vectors in S are linearly independent, they can be preprocessed to form an orthonormal set (Gram-Schmidt procedure)
- Preprocessing to enhance separability of key vectors in S (i.e. feature enhancement) helps improve storage capacity
- Correlation memory matrix may recall patterns that it has never seen before
  - It can make errors if new pattern is not orthonormal against key vectors in S

AAM Net

mansoori@shirazu.ac.ir

Storage capacity of AAM nets:

- **Szu theorem**: $n-1$ mutually orthogonal bipolar vectors with n components can be stored using sum of outer product weight matrices (with diagonal terms set to zero)

- Storing n mutually orthogonal vectors will result in a weight matrix that cannot recall any of stored patterns

Example:

$\vec{s} = \{<1,1,-1,-1>, <-1,1,1,-1>, <-1,1,-1,1>\}$

$$W = W(1) + W(2) + W(3) = \begin{bmatrix} 0 & -1 & -1 & -1 \\ -1 & 0 & -1 & -1 \\ -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 \end{bmatrix}$$

$$\vec{s'} = \vec{s} \cup \{<1,1,1,1>\} \implies W' = W + W(4) = W + \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Iterative AAM Net

mansoori@shirazu.ac.ir

- In some cases, when net does not respond immediately to an input signal, but response is like a stored pattern, it can be applied to net again

$$\vec{s} = \{< 1,1,-1,-1 >\} \Rightarrow W = \begin{bmatrix} 0 & 1 & -1 & -1 \\ 1 & 0 & -1 & -1 \\ -1 & -1 & 0 & 1 \\ -1 & -1 & 1 & 0 \end{bmatrix}$$

$$\vec{x} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \overrightarrow{y\_in} = \begin{bmatrix} 0 \\ 1 \\ -1 \\ -1 \end{bmatrix} \Rightarrow \vec{y} = \begin{bmatrix} 0 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$

$$\vec{x} = \begin{bmatrix} 0 \\ 1 \\ -1 \\ -1 \end{bmatrix} \Rightarrow \overrightarrow{y\_in} = \begin{bmatrix} 3 \\ 2 \\ -2 \\ -2 \end{bmatrix} \Rightarrow \vec{y} = \begin{bmatrix} 1 \\ 1 \\ -1 \\ -1 \end{bmatrix}$$