ASSOCIATIVE MEMORIES

INT301 Bio-computation, Week 13, 2021

Memory in Computer System

- Standard computer memory is accessed through **assigned addresses**.
- When a user searches for a file, the CPU must convert the request to a numerical instruction and then search through the memory for the corresponding address
- A computer's memory is most commonly referred to as RAM (random access memory).

Associative Memory & Pattern Association

- An associative memory is a **content-addressable structure** that maps a set of input patterns to a set of output patterns.

- **That is, memory can be directly accessed by the content, rather than the physical address in the memory.**

Associative Memory & Pattern Association

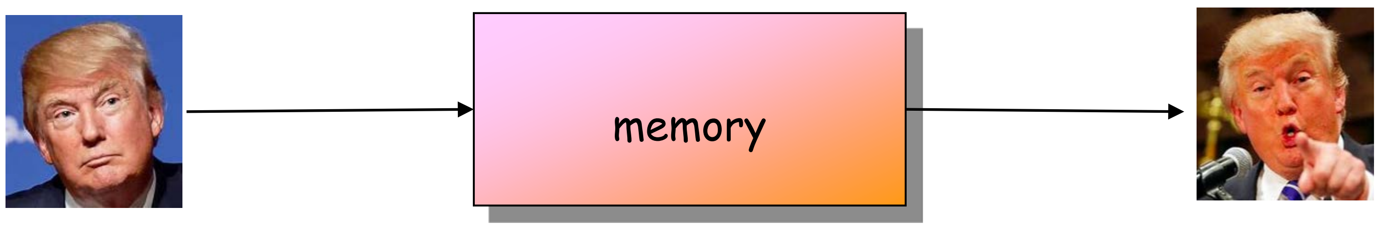
- Associative memory is often linked to pattern association
  - Associating patterns which are
    - similar
    - contrary
    - in close proximity (spatial)
    - in close succession (temporal)
  - Associative recall
    - evoke associated patterns
    - recall a pattern by part of it
    - evoke/recall with incomplete/noisy patterns

Associative Memories

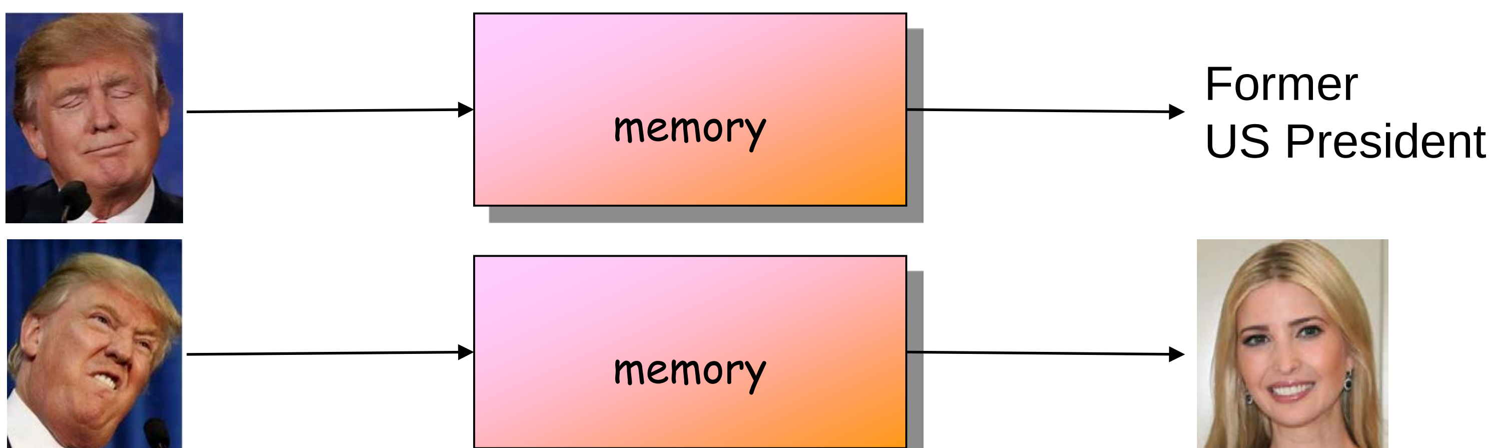
- **Two types of associative memory**: ***auto-associative*** and ***hetero-associative***.
- Auto-association
  - retrieves a previously stored pattern that most closely resembles the current pattern.
- Hetero-association
  - the retrieved pattern is, in general, different from the input pattern not only in content but possibly also in type and format.

Associative Memories

Auto-association



Hetero-association



Associative Memories

Stored Patterns

$$(\mathbf{x}^1, \mathbf{y}^1)$$
$$(\mathbf{x}^2, \mathbf{y}^2)$$
$$\vdots$$
$$(\mathbf{x}^p, \mathbf{y}^p)$$

$$\mathbf{x}^i \equiv \mathbf{y}^i$$ Autoassociative

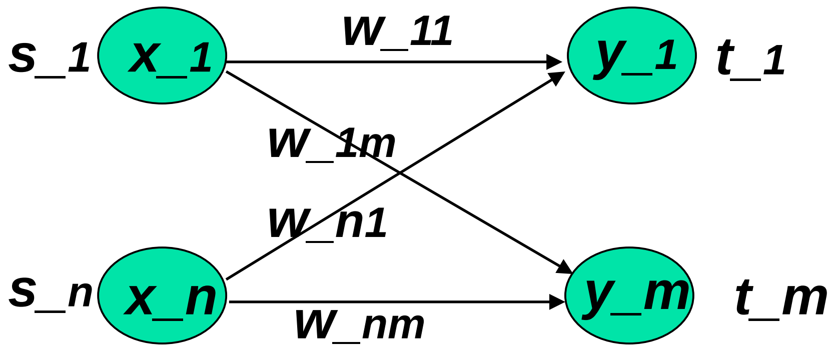
$$\mathbf{x}^i \neq \mathbf{y}^i$$ Heteroassociative

$$\mathbf{x}^i \in R^n$$

$$\mathbf{y}^i \in R^m$$

Simple AM

- Network structure: single layer
  - one output layer of non-linear units and one input layer
  - similar to the simple network for classification



- Goal of learning:
  - to obtain a set of weights ***w_ij***
  - from a set of training pattern pairs {***s*** **:** ***t***}
  - such that when ***s*** is applied to the input layer, ***t*** is computed at the output layer

Simple AM

- Similar to Hebbian learning for classification
- Algorithm: (bipolar or binary patterns)
  - For each training samples ***s*** **:** ***t*** $\Delta w_{ij} = s_i \cdot t_j$
  - Δw_{ij} increases if both input and output are ON (binary) or have the same sign (bipolar)
- If $\Delta w_{ij} = 0$ initially, then after updates for all P training patterns

$$w_{ij} = \sum_{p=1}^{P} s_i(p) t_j(p) \quad W = \{ w_{ij} \}$$

- Instead of obtaining ***W*** by iterative updates, it can be computed from the training set by calculating the outer product of ***s*** and ***t***.

Simple AM

- **Outer product**: Let ***s*** and ***t*** be **row** vectors.

  Then for a particular training pair ***s:t***

$$\Delta W(p) = s^T(p) \cdot t(p) = \begin{bmatrix} s_1 \\ \\ \\ s_n \end{bmatrix} [t_1, \ldots\ldots t_m] = \begin{bmatrix} s_1 t_1 \ldots\ldots s_1 t_m \\ s_2 t_1 \ldots\ldots s_2 t_m \\ \\ s_n t_1 \ldots\ldots s_n t_m \end{bmatrix} = \begin{bmatrix} \Delta w_{11} \ldots\ldots \Delta w_{1m} \\ \\ \\ \Delta w_{n1} \ldots\ldots \Delta w_{nm} \end{bmatrix}$$

  and $W(P) = \sum_{p=1}^{P} s^T(p) \cdot t(p)$

- It involves 3 nested loops p, i, j (order of p is irrelevant)

```
p= 1 to P                    /* for every training pair */
    i = 1 to n               /* for every row in W      */
        j = 1 to m           /* for every element j in row i */
```

$$w_{ij} := w_{ij} + s_i(p) \cdot t_j(p)$$

Simple AM

- Does this method provide a good association?
  - Recall with training samples (after the weights are learned or computed)
  - Apply $\boldsymbol{s}(\boldsymbol{k})$ to one layer, hope $\boldsymbol{t}(\boldsymbol{k})$ appear on the other, e.g. $f(\boldsymbol{s}(\boldsymbol{k})W) = \boldsymbol{t}(\boldsymbol{k})$
  - May not always succeed (each weight contains some information from all samples)

$$\boldsymbol{s}(\boldsymbol{k})W = \boldsymbol{s}(\boldsymbol{k})\sum_{p=1}^{P}\boldsymbol{s}^T(\boldsymbol{p})\boldsymbol{t}(\boldsymbol{p}) = \sum_{p=1}^{P}\boldsymbol{s}(\boldsymbol{k})\cdot\boldsymbol{s}^T(\boldsymbol{p})\cdot\boldsymbol{t}(\boldsymbol{p})$$

$$= \boldsymbol{s}(\boldsymbol{k})\boldsymbol{s}^T(\boldsymbol{k})\boldsymbol{t}(\boldsymbol{k}) + \sum_{p\neq k}\boldsymbol{s}(\boldsymbol{k})\boldsymbol{s}^T(\boldsymbol{p})\boldsymbol{t}(\boldsymbol{p})$$

$$= \|\boldsymbol{s}(\boldsymbol{k})\|^2\,\boldsymbol{t}(\boldsymbol{k}) + \sum_{p\neq k}\boldsymbol{s}(\boldsymbol{k})\boldsymbol{s}^T(\boldsymbol{p})\boldsymbol{t}(\boldsymbol{p})$$

principal term

cross-talk term

Simple AM

- Principal term gives the association between ***s*(*k*)** and ***t*(*k*)**.
- Cross-talk represents correlation between ***s*(*k*):*t*(*k*)** and other training pairs. When cross-talk is large, ***s*(*k*)** will recall something other than ***t*(*k*)**.
- If all ***s*(*p*)** are orthogonal to each other, then $\mathbf{s}(\mathbf{k})\cdot\mathbf{s}^{T}(\mathbf{p})=0$, no sample other than ***s*(*k*):*t*(*k*)** contribute to the result.
- However, there are at most ***n*** orthogonal vectors in an *n*-dimensional space.
- Cross-talk increases when *P* increases.

Example 1: hetero-associative

- Binary pattern pairs ***s*** **:** ***t*** with **|*s*|** = 4 and **|*t*|** = 2.
- Total weighted input to output units: $y\_in_j = \sum_i x_i w_{ij}$
- Activation function: threshold

$$y_j = \begin{cases} 1 & \text{if} \quad y\_in_j > 0 \\ 0 & \text{if} \quad y\_in_j \le 0 \end{cases}$$

- Weights are computed by Hebbian rule (sum of outer products of all training pairs)

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

- Training samples:

	s(p)	t(p)
p=1	(1 0 0 0)	(1, 0)
p=2	(1 1 0 0)	(1, 0)
p=3	(0 0 0 1)	(0, 1)
p=4	(0 0 1 1)	(0, 1)

Example 1: hetero-associative

$$s^T(1)\cdot t(1) = \begin{pmatrix}1\\0\\0\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} = \begin{pmatrix}1 & 0\\0 & 0\\0 & 0\\0 & 0\end{pmatrix} \qquad s^T(2)\cdot t(2) = \begin{pmatrix}1\\1\\0\\0\end{pmatrix}\begin{pmatrix}1 & 0\end{pmatrix} = \begin{pmatrix}1 & 0\\1 & 0\\0 & 0\\0 & 0\end{pmatrix}$$

$$s^T(3)\cdot t(3) = \begin{pmatrix}0\\0\\0\\1\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix} = \begin{pmatrix}0 & 0\\0 & 0\\0 & 0\\0 & 1\end{pmatrix} \qquad s^T(4)\cdot t(4) = \begin{pmatrix}0\\0\\1\\1\end{pmatrix}\begin{pmatrix}0 & 1\end{pmatrix} = \begin{pmatrix}0 & 0\\0 & 0\\0 & 1\\0 & 1\end{pmatrix}$$

Computing the weights $W = \sum_{p=1}^{P} s_i^T(p)t_j(p)$ $\qquad W = \begin{pmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{pmatrix}$

Example 1: hetero-associative

x=(1 0 0 0)

$$\begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}\begin{pmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{pmatrix}=\begin{pmatrix}2 & 0\end{pmatrix}$$

$$y_1 = 1, \quad y_2 = 0$$

x=(0 1 0 0) similar to S(1) and S(2)

$$\begin{pmatrix}0 & 1 & 0 & 0\end{pmatrix}\begin{pmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{pmatrix}=\begin{pmatrix}1 & 0\end{pmatrix}$$

$$y_1 = 1, \quad y_2 = 0$$

x=(0 1 1 0)

$$\begin{pmatrix}0 & 1 & 1 & 0\end{pmatrix}\begin{pmatrix}2 & 0\\1 & 0\\0 & 1\\0 & 2\end{pmatrix}=\begin{pmatrix}1 & 1\end{pmatrix}$$

$$y_1 = 1, \quad y_2 = 1$$

(1 0 0 0), (1 1 0 0) class (1, 0)

(0 0 0 1), (0 0 1 1) class (0, 1)

(0 1 1 0) is not sufficiently similar to any class

Example 2: auto-associative

- Same as hetero-associative nets, except ***t*(*p*) = *s* (*p*)**.
- Used to recall a pattern by its noisy or incomplete version. (**pattern completion / pattern recovery**)
- A single pattern ***s*** = (1, 1, 1, -1) is stored (weights computed by Hebbian rule – outer product)

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

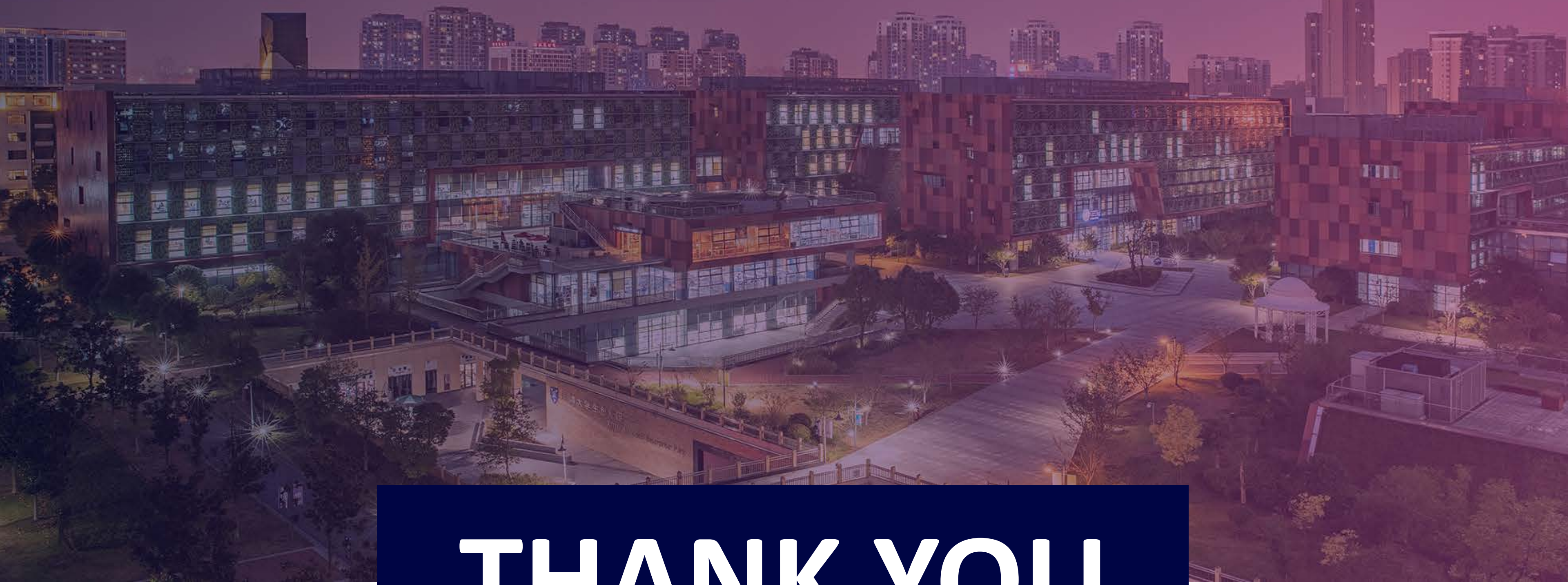
training pat.	$(1 1 1 -1) \cdot \boldsymbol{W} = (4\ 4\ 4\ -4) \rightarrow (1 1 1 -1)$
noisy pat	$(-1 1 1 -1) \cdot \boldsymbol{W} = (2\ 2\ 2\ -2) \rightarrow (1 1 1 -1)$
missing info	$(0\ 0\ 1 -1) \cdot \boldsymbol{W} = (2\ 2\ 2\ -2) \rightarrow (1 1 1 -1)$
more noisy	$(-1 -1 1 -1) \cdot \boldsymbol{W} = (0\ 0\ 0\ 0)$ not recognized

Example 2: auto-associative

- *W* is always a symmetric matrix
- Diagonal elements will dominate the computation when multiple patterns are stored (= P).
- When P is large, *W* is close to an identity matrix. This causes output = input, which may not be any stoned pattern. The pattern correction power is lost.
- Replace diagonal elements by zero:

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

$$\begin{aligned}
(1 \;\; 1 \;\; 1 \;\; -1)W' &= (3 \;\; 3 \;\; 3 \;\; -3) \rightarrow (1 \;\; 1 \;\; 1 \;\; -1) \\
(-1 \;\; 1 \;\; 1 \;\; -1)W' &= (3 \;\; 1 \;\; 1 \;\; -1) \rightarrow (1 \;\; 1 \;\; 1 \;\; -1) \\
(0 \;\; 0 \;\; 1 \;\; -1)W' &= (2 \;\; 2 \;\; 1 \;\; -1) \rightarrow (1 \;\; 1 \;\; 1 \;\; -1) \\
(-1 \;\; -1 \;\; 1 \;\; -1)W' &= (1 \;\; 1 \;\; -1 \;\; 1) \rightarrow \; wrong
\end{aligned}$$



THANK YOU

VISIT US

WWW.XJTLU.EDU.CN

FOLLOW US

@XJTLU

Xi'an Jiaotong-Liverpool University
西交利物浦大學