Neural Networks

モデル化とシミュレーション特論
2023 年度前期
佐賀大学理工学研究科　只木進一

Introduction: Neurons and Brains

- Neural networks (神経回路網)
  - Generate collective responses to stimuli (刺激).
- Single cell organisms (単細胞生物)
  - Lacks neurons
  - Respond to external stimuli.
- Plants (植物)
  - Lacks neurons
- Multi-cell animals (多細胞動物)
  - Neurons through cell differentiation (細胞分化)

Neural systems in multi-cell animals

- Poriferan (海綿動物) and Placozoa (平板動物)
  - Lacks neurons
  - Minimum cell differentiation
- Animals with scattered neural systems (散在神経系)
  - Neural network on body surface
  - Not posses a central neural system (中枢神経)
  - Coelenterate (腔腸動物): jellyfish (くらげ), coral (さんご) etc.

Animals with cage-shaped neural system (かご型神経系をもつ動物)

- Ganglion (神経節) as a center of neural system at head
- Cage-shaped neural network on body surface
- Flatworm (扁形動物): Planaria (プラナリア) etc.

Animals with ladder-shaped neural system (はしご型神経系をもつ動物)

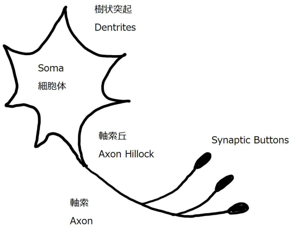
- Brains evolved through ganglion at heads
- Central neural system at abdomens (腹面)
- Ganglions at each somites (体節)
- Arthropod (節足動物): insects (昆虫) etc.
- Annelid (環形動物): earthworm (ミミズ) etc.

Animals with tubular neural system (管状神経系をもつ動物)

- Brains evolved through ganglion at heads
- Central neural system at body center
- Chordate (脊索動物)
- Vertebrate (脊椎動物)

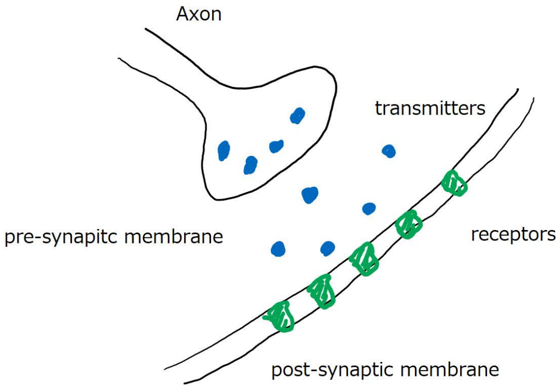
Neurons (神経細胞)

- Receive pulses from other neurons through synapses
- Electric pulses using ions
- Coding scheme are not clearly understood
- Two states: fire and rest
- Neuron fires if pulses from other neurons exceed some threshold



- Soma (細胞体)
    - Keep living activities such as normal cells
- Dendrite (樹状突起)
    - Receive pulses from other neurons
- Axon (軸索)
    - Send pulses to other neurons
    - Tip divided into 10,000 synaptic buttons for human neurons
- Cell division almost finished during early childhood
- Body cells keep dividing lifelong

Synapse


Axon
transmitters
pre-synapitc membrane
receptors
post-synaptic membrane

McCulloch-Pitts model

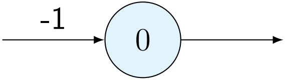
- Stimuli for neuron j from neurons i

$$x_j = \phi\left(\sum_i w_{ji}x_i - h_j\right) \tag{2.1}$$

- x_i: output from neuron i
- w_{ji}: synaptic connection
- h_j : threshold
- ϕ : response function, usually has a sigmoidal shape

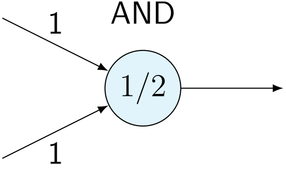
McCulloch-Pitts neuron as a logical gate

NOT

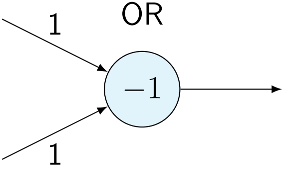


- Encode $\{\mathrm{T}, \mathrm{F}\}$ by $\{1, -1\}$
- Threshold values are shown as numerical values in nodes
- Assume step response functions.

AND



OR



logic gates

- Step function

$$\theta(x) = \begin{cases} 1 & x \geq 0 \\ -1 & x < 0 \end{cases} \tag{2.2}$$

- NOT

$$\theta\left(-x - 0\right) = \begin{cases} -1 & x = 1 \\ 1 & x = -1 \end{cases} \tag{2.3}$$

- AND

$$\theta\left(x + y - \frac{1}{2}\right) = \begin{cases} 1 & x = 1,\ y = 1 \\ -1 & \text{otherwise} \end{cases} \tag{2.4}$$

- OR

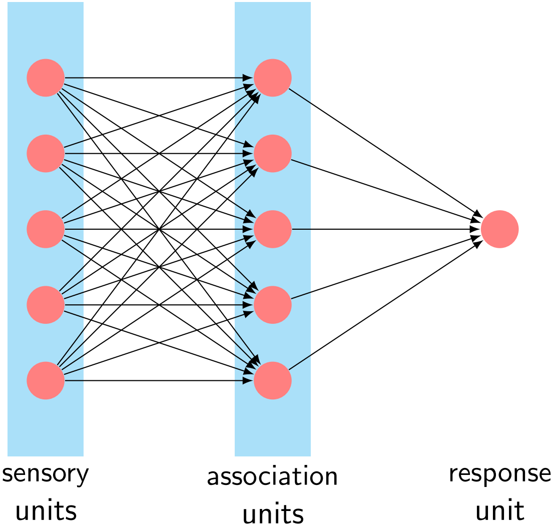
$$\theta\left(x + y - 1\right) = \begin{cases} -1 & x = 0,\ y = 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.5}$$

Classes in `model` package

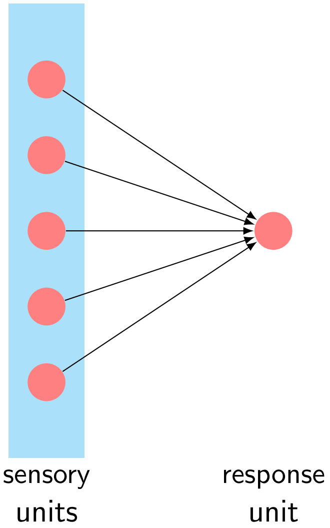
- `Neuron` Class
    - Constructor with weight and response function
    - `response()` with input
- `McCullochPitts` class
    - Show response of `AND`, `OR`, and `NOT` gates.
- `CorrectResponse` class
- `AbstractMultiLayer` class

Perceptron : Rosenblatt (1966)

Learning and recognition by neural networks



Two-layer perceptron



Two-layer perceptron

- Response unit receives $\{a_i\}$ from sensory units
- Response unit outputs η

$$\eta = \theta\left(\sum_j \xi_j a_j\right) \tag{3.1}$$

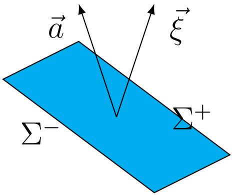
- ξ_j: weight for input
- The threshold is placed as the last element of the weight. And the last element of input is constant.
- θ : Step function

$$\theta\left(\vec{\xi}\cdot\vec{a}\right) = \begin{cases} 1 & \vec{\xi}\cdot\vec{a} \geq 0 \\ -1 & \vec{\xi}\cdot\vec{a} < 0 \end{cases} \tag{3.2}$$

Linearly Separability

- Consider a input space spanned by $\vec{a}$
- Divide the space by a hyper space normal to the weight vector $\vec{\xi}$

$$\begin{cases} \vec{\xi} \cdot \vec{a} \geq 0 & \vec{a} \in \Sigma^{+} \\ \vec{\xi} \cdot \vec{a} < 0 & \vec{a} \in \Sigma^{-} \end{cases} \tag{3.3}$$



Learning by error-correction

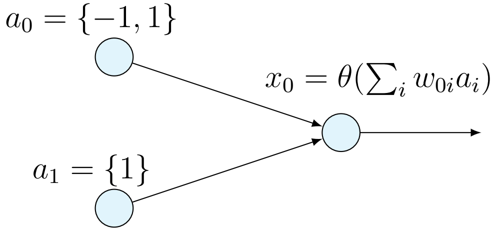
$$\begin{cases} \vec{\xi} \to \vec{\xi} + c\vec{a} & \text{if } \eta = -1 \text{ for } \vec{a} \in \Sigma^+ \\ \vec{\xi} \to \vec{\xi} - c\vec{a} & \text{if } \eta = 1 \text{ for } \vec{a} \in \Sigma^- \end{cases} \tag{3.4}$$

- Correct reusponce η_{corerct}

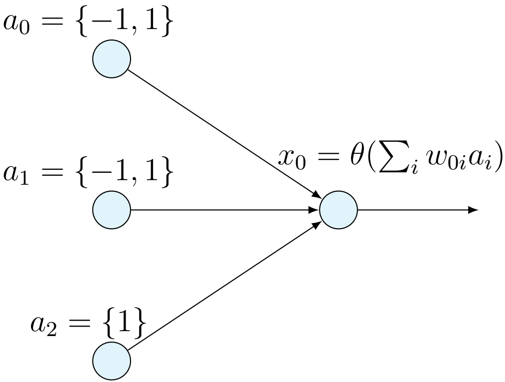
$$\eta_{\text{correct}} = \begin{cases} 1 & \vec{a} \in \Sigma^+ \\ -1 & \vec{a} \in \Sigma^- \end{cases} \tag{3.5}$$

$$\vec{\xi} \to \vec{\xi} - \frac{c}{2} \left(\eta - \eta_{\text{correct}} \right) \vec{a} \tag{3.6}$$

Example: Perceptron learning NOT gate



Example: Perceptron learning AND and OR gates

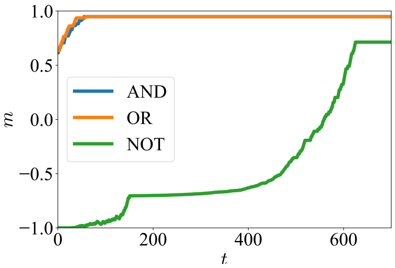


Classes in `twoLayer` package

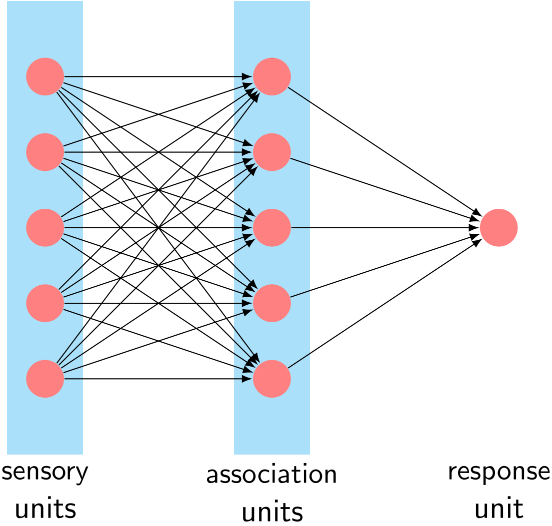
- `TwoLayer` class: general two layer model
- `LearningLogicGate` class: general learning model
- `LearningAndGate`, `LearningOrGate`, and `LearningNotGate`

Experiments

1. $\vec{\xi}_0$: Correct answers
2. $\vec{\xi}$: Initial random vectors
3. At each learning step
   1. learning
   2. normalize vector
   3. evaluation: $m = \vec{\xi}_0 \cdot \vec{\xi}$



3-layer perceptron



3-layer perceptron

- Outputs from sensory units: $\{a_i\}$
- Outputs from association units: $\{x_i\}$

$$x_i = f\left(\sum_j w_{ij} a_j\right) \tag{4.1}$$

w_{ij}: weight

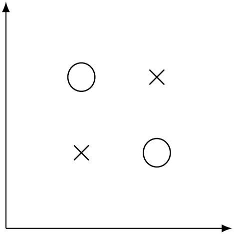
- Output from response unit: η

$$\eta = g\left(\sum_k s_k x_i\right) \tag{4.2}$$

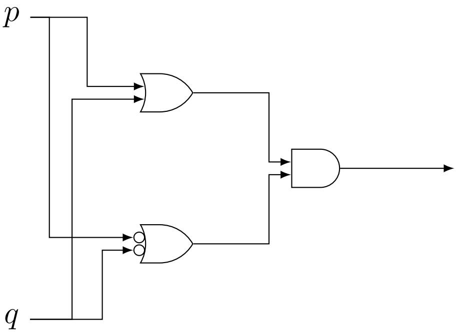
s_k: weight

Example XOR
Note that XOR is not liniearly separable

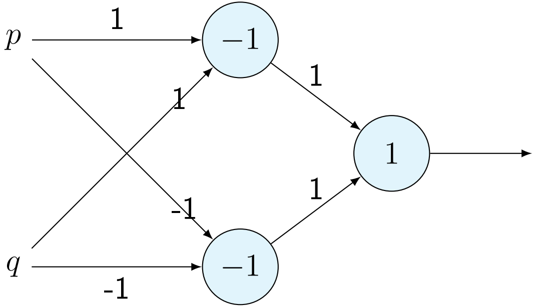
p	q	p XOR q
0	0	0
0	1	1
1	0	1
1	1	0



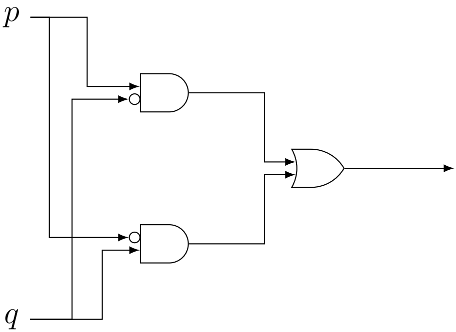
Example: XOR logical circuit



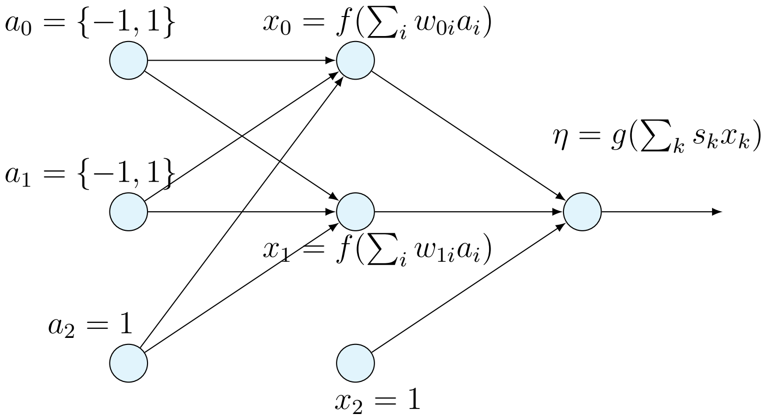
Implement as 3-layer perceptron of XOR



Example: XOR logical circuit: another version



Real implementation



Back propagation method

- Minimize square error

$$E = \frac{1}{2}\left(\eta - \eta_{\mathrm{correct}}\right)^2 \tag{4.3}$$

- Continuous output from any elements

$$x_i = f\left(\sum_j w_{ij} a_j\right) \tag{4.4}$$

$$\eta = g\left(\sum_k s_k x_k\right) \tag{4.5}$$

Update weights in response unit

$$
\begin{aligned}
\frac{\partial E}{\partial s_k} &= (\eta - \eta_{\mathrm{correct}})\, g'\left(\sum_j s_j x_j\right) x_k \\
&\equiv r x_k
\end{aligned}
\tag{4.6}
$$

$$
s_k \to s_k - c r x_k \tag{4.7}
$$

Update weights in association unit

$$\frac{\partial E}{\partial w_{ij}} = (\eta - \eta_{\mathrm{correct}})\, g'\left(\sum_k s_k x_k\right) s_i f'\left(\sum_\ell w_{i\ell} a_\ell\right) a_j$$

$$\equiv \tilde{r}_i a_j \tag{4.8}$$

$$\tilde{r}_i \equiv r s_i f'\left(\sum_\ell w_{i\ell}\right) \tag{4.9}$$

$$w_{ij} \to w_{ij} - c\tilde{r}_i a_j \tag{4.10}$$

- Errors seem to propagate backwardly from the response unit to the association units
- Updates possibly be trapped at local minima

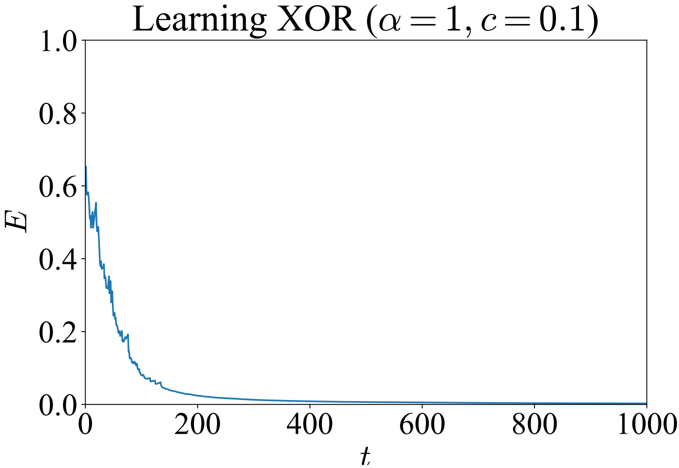
Simulation setups

- Response function

$$f(x) = g(x) = \tanh(\alpha x) \tag{4.11}$$

- random initial value: $\{w_{ij}\}$、$\{s_k\}$
- Observe error at every step

$$\bar{E} = \frac{1}{4} \sum_{a_0=\{-1,1\}} \sum_{a_1=\{-1,1\}} \frac{1}{2} \left(\eta - \eta_{\text{correct}}\right)^2 \tag{4.12}$$



Learning XOR ($\alpha = 1, c = 0.1$)
1.0
0.8
0.6
0.4
0.2
0.0
E
0
200
400
600
800
1000
t

Responce of output unit

p	q	o
-1	-1	-1
-1	1	-0.94
1	-1	-0.94
1	1	0.98

Output unit works like AND gate.

Responces of associative units

p	q	o_1	o_2
-1	-1	-0.92	1
-1	1	0.85	0.84
1	-1	0.86	0.85
1	1	1	-0.92

The 1st unit works like $p \vee q$. and 2nd one $\neg p \vee \neg q$.