Hopfield model

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

- Neural network with mutual interaction
  - memory and retrieval
- Non-deterministic motion
- Hopfield model
- Boltzmann machine

https://github.com/modeling-and-simulation-mc-saga/Hopfield

Hopfield model

- N neurons: state $s_i = \{-1, 1\}$
- Interactions (symmetric because of theoretical reasons)

$$w_{ij} = w_{ji}, \quad w_{ii} = 0 \tag{2.1}$$

- Asynchronous updates
  - Select one neuron randomly and try to update its state
  - One Monte Carlo step consists of N update trials

$$s_i = \begin{cases} 1 & \text{if } h_i = \sum_j w_{ij} s_j \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{2.2}$$

Energy

- Energy

$$E = -\frac{1}{2}\sum_i \sum_j w_{ij} s_i s_j \tag{2.3}$$

- Energy varies by update of i-th neuron

$$\delta E = -\sum_j w_{ij} s_j \delta s_i \leq 0 \tag{2.4}$$

- δs_i always has the same sign of $h_i = \sum_j w_{ij} s_j$

- Energy monotonously decreases
  - Monotonously degreasing functions are called *Lyapunov* functions
- For $h_i = \sum_j w_{ij} s_j \geq 0$

$$\delta s_i = \begin{cases} 2 & \text{if} \quad s_i = -1 \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

- For $h_i = \sum_j w_{ij} s_j \leq 0$

$$\delta s_i = \begin{cases} -2 & \text{if} \quad s_i = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.6}$$

Hebb's learning rule

- For a learned pattern $\vec{\xi}$

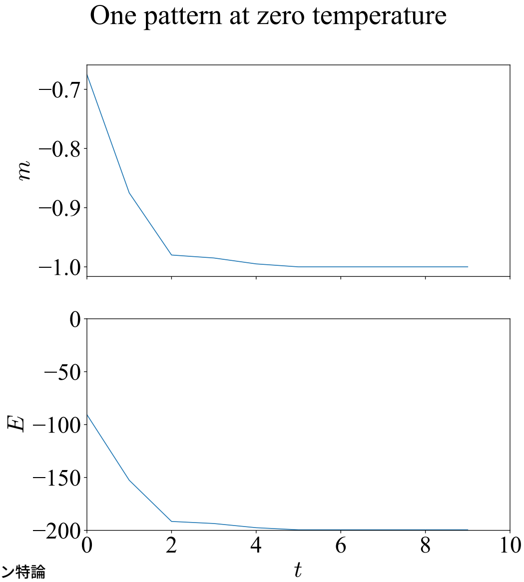
$$w_{ij} = \lambda \xi_i \xi_j, \qquad i \neq j \tag{2.7}$$

$$\begin{aligned} E &= -\frac{1}{2} \sum_i \sum_{j \neq i} \lambda \xi_i \xi_j s_i s_j \\ &= -\frac{\lambda}{2} \left[\left(\sum_i \xi_i x_i \right)^2 - \sum_i \xi_i^2 s_i^2 \right] = -\frac{\lambda}{2} \left[\left(\sum_i \xi_i x_i \right)^2 - N \right] \end{aligned} \tag{2.8}$$

Memoried patterns

- Two energy minima: $\vec{s} = \pm\vec{\xi}$
- Same as ferromagnetic (強磁性) systems
    - Memoried patterns are given as energy minima

Simulation with one pattern



P patterns

- Hebb' rule

$$w_{ij} = \lambda \sum_{\mu=0}^{P-1} \xi_i^\mu \xi_j^\mu \tag{2.9}$$

- Overlapping with μ-th pattern

$$m_\mu = \frac{1}{N} \sum_i \xi_i^\mu s_i \tag{2.10}$$

$$E = -\frac{\lambda}{2}N^2 \sum_{\mu} (m_\mu)^2 + \frac{\lambda}{2}NP \tag{2.11}$$

- Each pattern is represented by energy minima, if they are orthogonal
- There may be energy minima not representing memory patterns

Dynamics at finite temperature

At finite temperature T, transition probability is given by

$$P\left(\delta s_i = \pm 2\right) = \frac{1}{1 + e^{\mp 2\beta h_i}} \tag{3.1}$$

$$h_i = \sum_j w_{ij} s_j, \quad \beta = 1/T \tag{3.2}$$

- probability of $\delta s_i = -2$ $(s_i = 1)$
  - low temperature limit($\beta \to \infty$)

$$P(\delta s_i = -2) \to \begin{cases} 1 & h_i > 0 \\ 0 & h_i < 0 \end{cases} \tag{3.3}$$

  - high temperature limit ($\beta \to 0$)

$$P(\delta s_i = -2) \to 1/2 \tag{3.4}$$

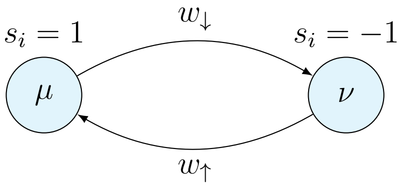
- probability of $\delta s_i = 2$ $(s_i = -1)$
  - low temperature limit($\beta \to \infty$)

$$P(\delta s_i = 2) \to \begin{cases} 1 & h_i < 0 \\ 0 & h_i > 0 \end{cases} \tag{3.5}$$

  - high temperature limit

$$P(\delta s_i = 2) \to 1/2 \tag{3.6}$$

Equilibrium



$$\frac{P_\mu}{P_\nu} = \frac{w_\uparrow}{w_\downarrow} = \frac{1+e^{-2\beta h_i}}{1+e^{2\beta h_i}} = e^{-2\beta h_i} = \frac{e^{-\beta h_i}}{e^{\beta h_i}}$$

$$= \frac{e^{-\beta E(\mu)}}{e^{-\beta E(\nu)}} \tag{3.7}$$

$$P_\mu \propto e^{-\beta E(\mu)} \tag{3.8}$$

$$\begin{aligned} h_i &= \sum_{j \neq i} w_{ij} 1 \times s_j \\ &= \frac{1}{2} \sum_{j \neq i} w_{ij} 1 \times s_j + \frac{1}{2} \sum_{j \neq i} w_{ij} s_j \times 1 + \frac{1}{2} \sum_{j \neq i} \sum_{k \neq i} w_{jk} s_j s_k \\ &= E(\mu) \end{aligned} \tag{3.9}$$

$$\begin{aligned} -h_i &= \sum_{j \neq i} w_{ij} (-1) \times s_j \\ &= \frac{1}{2} \sum_{j \neq i} w_{ij} (-1) \times s_j + \frac{1}{2} \sum_{j \neq i} w_{ij} s_j \times (-1) + \frac{1}{2} \sum_{j \neq i} \sum_{k \neq i} w_{jk} s_j s_k \\ &= E(\nu) \end{aligned} \tag{3.10}$$

Simulation

- Setting 10 Kanji patterns: not orthogonal!
- Change the number of patterns
- Zero or finite temperature
- Observe how the system find one of patterns

model package

- Neuron class:states of a neuron
- Hopfield class
  - generate weight vectors from patterns
  - update states at zero and finite temperature
  - evaluate overlapping with patterns
  - evaluate the energy
- AbstractPatterns

simpleExample package

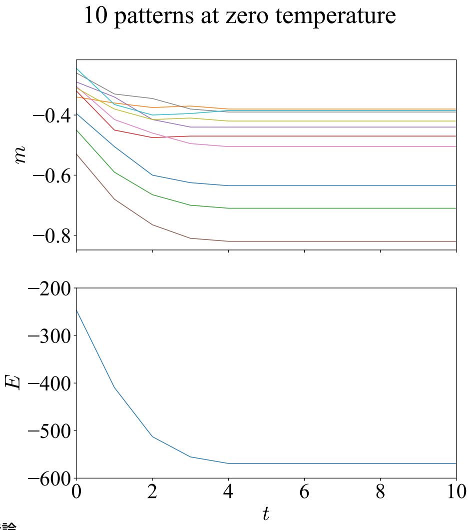
- SimplePattern class: define the memorized pattern
- SimplePatternMain class
    - Run 10 Monte Carlo steps
    - Output overlapping and energy

simpleExample package

- `KanjiPatterns` class: define 10 kanji patterns
- `KanjiPatternsMain` class
  - Run 1000 Monte Carlo steps
  - Output overlapping and energy
  - $T = 0$ and annealing from $T = 10$ cases

Zero temperature

Overlapping with patterns and energy at $T = 0$



Finite temperature

Overlapping with patterns and energy through simulated annealing