HW3

Ising Model with Visualization

陳凱騫

2024-12-15

Table of contents

簡介

本次報告實現 2D Ising 模型，透過 **Metropolis 演算法** 模擬格點演化，並在不同時間步驟視覺化格點狀態。此外，分析能量和

模型邏輯與程式碼

1. 載入必要套件

```
# 載入視覺化套件
library(ggplot2)
library(gridExtra)

# 載入矩陣操作套件
library(reshape2)
```

---

2. 參數設定與初始化

```
# 設定初始參數
L <- 20            # 格點大小 (L x L)
T <- 2.5           # 溫度
J <- 1             # 交換作用強度
steps <- c(0, 1, 4, 32, 100, 1000) # 時間步驟
beta <- 1 / T      # 反溫度
```

```r
# 初始化隨機自旋配置 (每個格點的值為 -1 或 1)
initialize_lattice <- function(L) {
  matrix(sample(c(-1, 1), L * L, replace = TRUE), L, L)
}

# 計算能量變化 (ΔE)
delta_energy <- function(lattice, i, j, L) {
  spin <- lattice[i, j]
  neighbors <- lattice[(i %% L) + 1, j] +
               lattice[(i - 2) %% L + 1, j] +
               lattice[i, (j %% L) + 1] +
               lattice[i, (j - 2) %% L + 1]
  return(2 * spin * neighbors)
}
```

註解： 1. initialize_lattice 函數生成隨機初始自旋矩陣，格點值為 (-1) 或 (1)。 2. delta_energy 函數計算自旋反轉帶來的能量變化 (□E)，透過考慮最近鄰居的影響。

---

3. Metropolis 演算法：自旋更新

```r
# Metropolis 演算法的單步更新
metropolis_step <- function(lattice, beta, L) {
  for (k in 1:(L * L)) { # 更新所有格點
    i <- sample(1:L, 1)  # 隨機選取行
    j <- sample(1:L, 1)  # 隨機選取列
    dE <- delta_energy(lattice, i, j, L)

    # 接受或拒絕更新
    if (dE <= 0 || runif(1) < exp(-dE * beta)) {
      lattice[i, j] <- -lattice[i, j] # 反轉自旋
    }
  }
  return(lattice)
}
```

邏輯： 1. 隨機選取格點 (i, j)。 2. 計算自旋反轉的能量變化 dE。 3. 使用 Metropolis 接受準則：
- 如果 dE <= 0 (能量下降)，接受更新。
- 如果 dE > 0，以概率 (e^{-□□E}) 接受更新。

---

4. 格點狀態的演化與視覺化

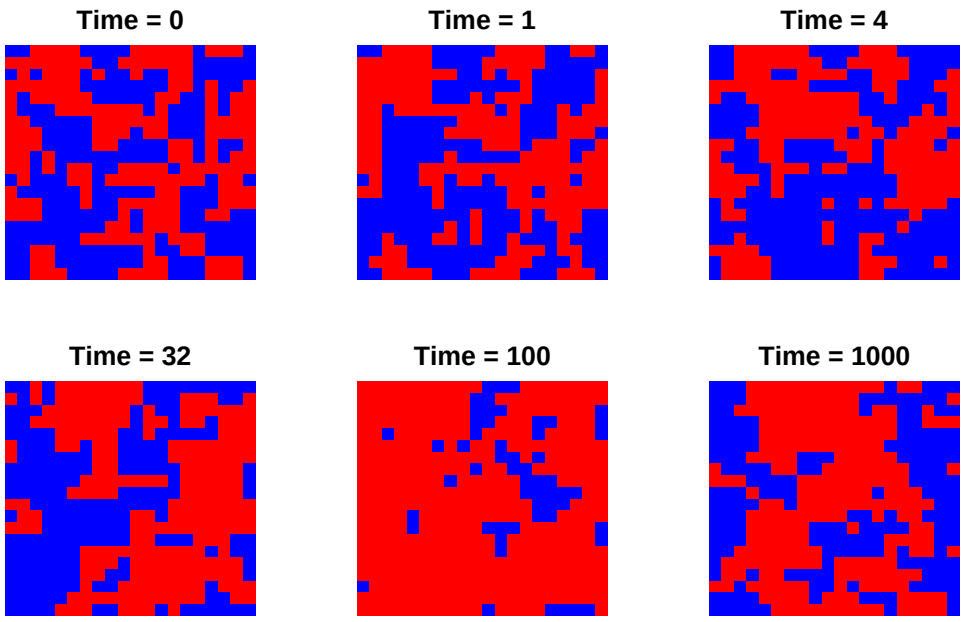
```r
# 繪製不同時間步驟的格點狀態
plot_lattice_evolution <- function(L, steps, beta) {
  lattice <- initialize_lattice(L) # 初始化格點
  par(mfrow = c(2, 3), mar = c(2, 2, 2, 2)) # 2x3 圖片布局

  for (t in 1:length(steps)) {
    step <- steps[t]
    for (s in 1:step) { # 執行 Metropolis 演算法 step 次
      lattice <- metropolis_step(lattice, beta, L)
```

```
  }

  # 視覺化格點狀態
  image(1:L, 1:L, t(lattice), col = c("blue", "red"), axes = FALSE,
        main = paste("Time =", step))
 }
}

# 執行視覺化
plot_lattice_evolution(L, steps, beta)
```



---

5. 能量與磁化率模擬

```
# 計算系統能量
calc_energy <- function(lattice, J) {
 E <- 0
 L <- nrow(lattice)
 for (i in 1:L) {
  for (j in 1:L) {
   neighbors <- lattice[(i %% L) + 1, j] +
                lattice[(i - 2) %% L + 1, j] +
                lattice[i, (j %% L) + 1] +
                lattice[i, (j - 2) %% L + 1]
   E <- E - J * lattice[i, j] * neighbors
  }
 }
 return(E / 2) # 避免重複計算
}

# 計算平均能量與磁化率
simulate_ising <- function(L, T, steps_eq, steps_mc) {
```

```
  beta <- 1 / T
  lattice <- initialize_lattice(L)

  # 平衡過程
  for (step in 1:steps_eq) {
    lattice <- metropolis_step(lattice, beta, L)
  }

  # 蒙地卡羅模擬
  energy <- magnetization <- 0
  for (step in 1:steps_mc) {
    lattice <- metropolis_step(lattice, beta, L)
    energy <- energy + calc_energy(lattice, J)
    magnetization <- magnetization + abs(sum(lattice))
  }

  list(Energy = energy / steps_mc,
       Magnetization = magnetization / (steps_mc * L^2))
}
```

結論

上述程式碼分別模擬 2D Ising 模型的演化過程、能量與磁化率的變化，並在不同時間步驟視覺化系統狀態。