MODEL ISINGA 2D

ab

10 grudnia 2016

TEMATYKA

- Model spinów Isinga
- Hamiltonian i suma statystyczna modelu
- Metoda Monte-Carlo. Algorytm Metropolisa.
- Obserwable

Hamiltonian, energia

$$E = -J \sum_{[ij]} s_i s_j - H \sum_i s_i$$

Tutaj

$$s_i = \begin{cases} +1\,, & \text{spin } \textit{do góry} \\ -1\,, & \text{spin } \textit{do dołu} \end{cases}$$

Pierwszy człon w E jest energią oddziaływania międzyspinowego, drugi opisuje oddziaływanie spinów z polem (magnetycznym) zewnętrznym H.

Magnetyzacja

$$M = \sum_i s_i$$

Spiny preferują kierunek pola magnetycznego.

W temperaturze niższej niż pewna temperatura kT_c, zwana temperaturą Curie, występuje spontaniczne uporządkowanie spinów (przy zerowym polu H) – przejście fazowe typu *paramagnetyk – magnetyk*. W temperaturach $kT > kT_c$ spiny są ustawione chaotycznie. Pole magnetyczne H modyfikuje zachowanie się układu spinów.

Suma statystyczna układu

$$Z = \sum_{\{E\}} \exp\left(-\beta E\right) = \sum_{\{E\}} \exp\left[-\beta(-J \sum_{[ij]} s_i s_j - H \sum_i s_i)\right]$$

$$\beta = \frac{1}{kT}$$

Obliczenia Z dla sieci jedno- i dwuwymiarowych, można otrzymać analitycznie. W przypadku sieci 3-wymiarowych pozostają obliczenia numeryczne oparte na metodzie Monte-Carlo (nawet dla sieci skończonych liczba konfiguracji rośnie bardzo szybko i obliczenia numerycznie dokładne są niemożliwe do wykonania).

Numeryczna suma ststystyczna nie jest dobrym punktem wyjścia dla obliczeń termodynamicznych (różniczkowanie jest praktycznie niewykonalne).

Metoda polega na generowaniu losowych konfiguracji, gdzie konfiguracja oznacza zbiór spinów przypadkowo skierowanych. Obliczone obserwable uśrednia się po zespole wygenerowanych konfiguracji.

Obserwable, np. magnetyzację, oblicza się zgodnie z rozkładem Boltzmanna

$$\langle M \rangle = \frac{\sum_{konfig} M \exp(-E/kT)}{\sum_{konfig} \exp(-E/kT)} .$$

Czynnik boltzmannowski zmienia się szybko z energią. Można to wykorzystać i generować konfiguracje najważniejsze, zgodne z rozkładem przwdopodobieństwa

$$p(s_1, s_2, \ldots s_n) = \frac{\exp(-E/kT)}{\sum_{konfig} \exp(-E/kT)} .$$

Nie znamy jednak mianownika tego wyrażenia.

W tym wypadku średnie wartości energii i magnetyzacji układu spinów obliczamy wg. wzorów

$$\langle E \rangle = \frac{1}{M} \sum_{k=1}^{M} E(k\text{-ta konfiguracja})$$

$$\langle M \rangle = \frac{1}{M} \sum_{k=1}^{M} M(k\text{-ta konfiguracja})$$

gdzie k-ta konfiguracja $= \{s_1^{(k)}, \ldots, s_n^{(k)}\}$, a M oznacza liczbę wygenerowanych konfiguracji.

Pierwszy algorytm wyboru konfiguracji w tego rodzaju problemach podali Metropolis i inni [2]. Jego realizacja jest następująca:

- Dla bieżącej konfiguracji wybierz spin s_i i przyjmij $s_{i,\,prba} = -s_i$
  - oblicz zmianę energii układu

$$\delta E = E(s_1, \ldots, s_{i,\,prba}, \ldots, s_n) - E(s_1, \ldots, s_i, \ldots, s_n)$$

  - jeżeli

$$p = e^{-\delta E / kT} > \gamma$$

    gdzie γ jest liczbą losową z rozkładu jednorodnego $(0, 1)$, wówczas $s_i \leftarrow s_{i,\,prba}$ (zamień spin na przeciwny).
- powtórz wszystko dla każdego spinu sieci

Zaczynając od konfiguracji początkowej, wykonujemy wstępną serię kroków MC – termalizacja, a następnie serię kroków gdzie obliczamy obserwable układu. Po tym obliczamy średnie ich wartości.

Kramers i Wannier [3] pokazali, że temperatura krytyczna dla kwadratowej sieci 2-wymiarowej wynosi $kT = 2.269$.

Onsager [4] rozwiązał model 2-wymiarowy dokładnie, w granicy termodynamicznej $n \to \infty$ dla $H = 0$.

```
/******************************************************************************
 *  Compilation:  javac Ising.java
 *  Execution:    java Ising N kT
 *
 *  Create an N-by-N grid of sites. Each site has spin "up" (+1)
 *  or "down" (-1).
 *
 *  % java Ising N 2.26918  // critical temperature
 *
 ******************************************************************************/

import java.awt.Color;

public class Ising {
    private int N;                  // N-by-N grid of sites
    private double J = 1.0;         // strength of spin-spin interaction
                                    // (ferromagnetic when J is positive)
    private double kT;              // temperature (say between 1 and 4)
    private boolean[][] spin;       // up (true) or down (false)
    private double p=.5;            // occupation probability

    // N-by-N grid, kT = temperature, p = probability of up spin
```

PRZYKŁAD PROGRAMU II

```
public Ising(int N, double kT, double p) {
    this.N    = N;
    this.kT   = kT;
    this.spin = new boolean[N][N];
    this.p = p;
}

void start() {
    // initialize random matrix of boolean values
    for (int i = 0; i < N; i++)
        for (int j = 0; j < N; j++)
            spin[i][j] = (Math.random() < p);

    // set scale and turn on animation mode
    StdDraw.setXscale(0, N);
    StdDraw.setYscale(0, N);
    StdDraw.enableDoubleBuffering();

    // loop over configurations
    for (int t = 0; true; t++) {
        phase();
        draw();
        StdDraw.show();
```

```
            StdDraw.pause(50);
        }
    }

    // total average magnetization (between -1 and 1)
    public double magnetization() {
        int M = 0;
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                if (spin[i][j]) M++;
                else            M--;
            }
        }
        return 1.0 * M / (N * N);
    }

    // total energy of site (i, j), using periodic boundary conditions
    // assumes 0 <= i, j < N
    private double energy(int i, int j) {
        double E = 0.0;
        if (spin[i][j] == spin[(i+1)%N][j])   E++;
        else                                  E--;
        if (spin[i][j] == spin[i][(j+1)%N])   E++;
```

```
        else                                    E--;
        if (spin[i][j] == spin[(i-1+N)%N][j]) E++;
        else                                    E--;
        if (spin[i][j] == spin[i][(j-1+N)%N]) E++;
        else                                    E--;
        return -J * E;
    }

    // total energy, using periodic boundary conditions
    public double energy() {
        double E = 0.0;
        for (int i = 0; i < N; i++)
            for (int j = 0; j < N; j++)
                // divide by two to mitigate double-counting
                E += 0.5 * energy(i, j);
        return E;
    }

    // one Monte Carlo step
    public void step(int i, int j) {
        double deltaE = -2 * energy(i, j);
        // flip if energy decreases or get lucky
        if ((deltaE <= 0) || (Math.random() <= Math.exp(-deltaE / kT)))
```

```
        spin[i][j] = !spin[i][j];
}

// one Monte Carlo phase - N^2 steps
public void phase() {
    for (int steps = 0; steps < N*N; steps++) {
        int i = (int) (Math.random() * N);
        int j = (int) (Math.random() * N);
        step(i, j);
    }
}

// plot it
public void draw() {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < N; j++) {
            if (spin[i][j]) StdDraw.setPenColor(StdDraw.WHITE);
            else            StdDraw.setPenColor(StdDraw.BLUE);
            StdDraw.filledSquare(i + 0.5, j + 0.5, .5);
        }
    }

    // draw lines
```

```
        StdDraw.setPenColor(StdDraw.BLACK);
        for (int i = 0; i < N; i++) {
            StdDraw.line(i, 0, i, N);
            StdDraw.line(0, i, N, i);
        }
    }

    // string representation
    public String toString() {
        String NEWLINE = System.getProperty("line.separator");
        String s = "";
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < N; j++) {
                if (spin[i][j]) s += "< ";
                else            s += "> ";
            }
            s += NEWLINE;
        }
        return s;
    }

    public static void main(String[] args) {
        if (args.length < 2) {
```

```
            System.out.println("use: java Ising L kT");
            System.exit(1);
        }
        int N = Integer.parseInt(args[0]);          // N-by-N lattice
        double kT = Double.parseDouble(args[1]);    // temperature
        Ising ising = new Ising(N, kT, 0.5);

        ising.start();
    }
}
```


R. Feynman.
Mechanika statystyczna. PWN. Warszawa.


N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller.
J. Chem. Phys. 21, 1087 (1953).


H.A. Kramers and G.H. Wannier.
Phys. Rev. 60, 252 (1941).


Lars Onsager.
Phys. Rev. 65, 117 (1944).

Thank you!

PRZYKŁAD

PRZYKŁAD