数値計算法自由レポート
スピングラスと連想記憶：ひこにゃんと地デジカの場合

OK

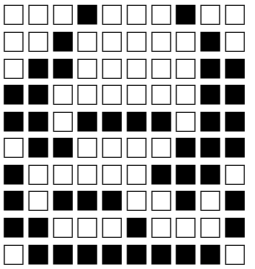
2011 年 8 月 1 日

概要

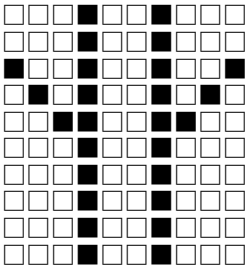
磁性を持たない金属に磁性体をわずかに混ぜ，冷やして固めると，スピンがバラバラな状態で固定される．これが非晶質のガラスのようだというので，スピングラスとよぶ．そのようなスピングラスのモデルを神経ネットワークに適用するような試みがある [1]．このレポートではそのような例を実際に計算機を動かして確かめてみる．

1 モデルの設定

10×10 のセルを用意する．そこにはそれぞれ上向きと下向きのどちらかのイジングスピンが入りうる．ここに，上向きのスピンを白，下向きのスピンを黒に対応させ，そこに以下のような 2 種類のパターンを埋め込む．



ひこにゃん[*1]の「かぶと」



地デジカ[*2]の「つの」

このパターンが，スピングラス模型によって再現されるかどうかを調べる．系のハミルトニアンの決め方はホップフィールド模型を採用する．

$$H = -\frac{1}{2}\sum_{(i,j)} J_{ij} S_i S_j$$

2 つのスピン i, j 間の相互作用は係数 J_{ij} で決まり，この係数は埋め込んだパターンによって決まる．(i, j) の選び方は相互作用の範囲を決めることに対応する．今回は周辺 4 つとだけ相互作用する場合と，全てのスピン間で相互作用する場合とを確認した．また，周期境界条件を採用した．

係数 J_{ij} は以下のヘッブ則で決める．

$$J_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu$$

で決める．これは埋め込んだパターンごとに和をとる．p はパターン数．

2 プログラム

すべてのスピン間で相互作用がある場合の C プログラムは以下のとおりである．

[*1] 彦根と滋賀をメジャーにしたおそるべきゆるきゃら．http://hikone-hikonyan.jp/

[*2] 地デジ化を推進するために送り込まれてきた刺客． http://www.nab.or.jp/chidejika/

```
#include<stdio.h>
#include<stdlib.h>
#include<math.h>
#include<time.h>

int main(){
  int cell[10][10];
  int oldcell[10][10];
  int kabuto[10][10];
  int tuno[10][10];
  int J[10][10];
  int i,j,k,l,m,n;
  int H1,H2;
  FILE *f1,*f2;

  f1=fopen("initial.txt","w");
  f2=fopen("final.txt","w");

  srand((unsigned)time(NULL));          /* 乱数の初期化 http://www.nmn.jp/~hidai/c/ */

  for(i=0;i<10;i++){
    for(j=0;j<10;j++){
      cell[i][j]=(rand() %2)*2-1;
    }
  }

  for(i=0;i<10;i++){
    for(j=0;j<10;j++){
      kabuto[i][j]=-1;
      tuno[i][j]=-1;
    }
  }

  j=0;
  kabuto[3][j]=1;
  kabuto[4][j]=1;
  kabuto[6][j]=1;
  kabuto[7][j]=1;
  kabuto[8][j]=1;
  tuno[2][j]=1;
  j=1;
  kabuto[2][j]=1;
```

```
kabuto[3][j]=1;
kabuto[4][j]=1;
kabuto[5][j]=1;
kabuto[8][j]=1;
kabuto[9][j]=1;
tuno[3][j]=1;
j=2;
kabuto[1][j]=1;
kabuto[2][j]=1;
kabuto[5][j]=1;
kabuto[7][j]=1;
kabuto[9][j]=1;
tuno[4][j]=1;
j=3;
kabuto[0][j]=1;
kabuto[4][j]=1;
kabuto[7][j]=1;
kabuto[9][j]=1;
for(i=0; i<10; i++){
  tuno[i][j]=1;
}

j=4;
kabuto[4][j]=1;
kabuto[7][j]=1;
kabuto[9][j]=1;
kabuto[9][j]=1;
j=5;
kabuto[4][j]=1;
kabuto[8][j]=1;
kabuto[9][j]=1;
j=6;
kabuto[4][j]=1;
kabuto[6][j]=1;
kabuto[9][j]=1;
j=7;
kabuto[0][j]=1;
kabuto[5][j]=1;
kabuto[6][j]=1;
kabuto[7][j]=1;
kabuto[9][j]=1;
j=8;
```

```
  kabuto[1][j]=1;
  kabuto[2][j]=1;
  kabuto[3][j]=1;
  kabuto[4][j]=1;
  kabuto[5][j]=1;
  kabuto[6][j]=1;
  kabuto[9][j]=1;
  j=9;
  kabuto[2][j]=1;
  kabuto[3][j]=1;
  kabuto[4][j]=1;
  kabuto[5][j]=1;
  kabuto[7][j]=1;
  kabuto[8][j]=1;

  for(i=0;i<10;i++){
    for(j=0;j<10;j++){
      tuno[i][9-j]=tuno[i][j];
    }
  }

  for(i=0;i<10;i++){
    for(j=0;j<9;j++){
      fprintf(f1,"%d ",cell[i][j]);
    }
   fprintf(f1,"%d\n",cell[i][9]);
  }

  /* ここまで初期化　ここから計算 */

  for(i=0;i<10;i++){
    for(j=0;j<10;j++){
      oldcell[i][j]=cell[i][j];
    }
  }

i=0;
  while(i<99999){
    k=rand()%10;
    l=rand()%10;

    H1=0;
```

```
      H2=0;

      for (m=0;m<10;m++){
        for (n=0;n<10;n++){
          J[m][n]=kabuto[k][l]*kabuto[m][n]+tuno[k][l]*tuno[m][n];
          H1=H1+J[m][n]*cell[k][l]*oldcell[m][n];
          H2=H2+J[m][n]*(-cell[k][l]*oldcell[m][n]);
        }
      }

      H1=H1-J[k][l]*cell[k][l]*oldcell[k][l自分自身との積を除く];//
      H2=H2-J[k][l]*(-cell[k][l])*oldcell[k][l]; 同じく//

        if(H2>H1)
          cell[k][l]=-cell[k][l];
    i++;
}

  for(i=0;i<10;i++){
    for(j=0;j<9;j++){
      fprintf(f2,"%d ",cell[i][j]);
    }
    fprintf(f2,"%d\n",cell[i][9]);
  }



}
```

はじめ，すべてのセルの状態をランダムになっているとする．そこで，ランダムにセルを抽出し，その状態のハミルトニアンとそのセルのスピンをひっくり返した状態のハミルトニアンを計算して，後者が小さければスピンを反転させる．これを100000 回繰り返した．これを実行すると 1 と-1 で書かれたファイルが出力されるので，それを

```
File.open("initial.txt"){|file|
while line =file.gets
 line1=line.gsub□("-1","")
 line2=line1.gsub■("1","")
puts line2
end
}
puts "--"
File.open("final.txt"){|file|
while line =file.gets
```

```
 line1=line.gsub□("-1","")
 line2=line1.gsub■("1","")
puts line2
end
}
```

のような Ruby プログラムで□と■に変換した.

3 結果

相互作用が全部の場合だと，有限の回数で必ずどちらかのパターンに落ち着いた．相互作用がまわり 4 つのみだと，パターンが想起されることはなかった.

あとがき

このプログラムは，西森秀稔『スピングラスと連想記憶』を参考にし，それを実際にプログラムを組んで確かめてみたものである．実はこれは，2009 年のオナーセミナーで発表した内容であるが [2]，当時はプログラムは数理生物学を専攻する先輩に書いてもらい，僕は中身を見ていなかったので，今回自分で書いてみた，当時は相互作用範囲の条件がより多かったが，今回は時間不足のため条件は少なくなっている.

参考文献

[1] 西森秀稔『スピングラスと連想記憶』

[2] 平成 21 年度オナーセミナー http://www.sci.osaka-u.ac.jp/honors/pdf/report/kouki_report_0330_bangai.pdf