ニューロコンピューティング　レポート課題

学籍番号 1014210 吉田洸平

提出日：2017/01/16

1　はじめに

Hopfield ネットワークの連想記憶に関する課題を行い，結果を本レポートにまとめる．ただし本来であれば課題は I，II の 2 つ存在するが今回は I のみについてまとめる．

2　概要

課題 I の手順は

1. 数字のパターンを 2 つ選び，Hopfield ネットワークに記憶させ，その 2 つのパターンが平衡状態になっているかどうかしらべる．つまり，ネットワークを動作させたときに記憶させたパターンが正しく呼び出されるかを確認する．
2. 記憶させたパターン数を増やしていった場合に正しく読み出しを行えるかどうか調べる．

というものである．以上の手順を実行するために，CSharp を用いて Hopfield ネットワークの一連の動作を再現するプログラムを作成する．具体的には

- 初期状態を生成するための関数
- パターンとして使用するデータをまとめる関数
- 結合の強さを設定するための関数
- 非同期的状態変化を行うための関数

を実装する．初期状態はあるパターンを一つ選んで，確率 0.1 で 0，1 を反転させたものとする．データをまとめる関数はパターンを増やしても，変更が少ないように実装する．結合の強さは

$$w_{ij} = \sum_{\alpha=0}^{K} (2_i^{\alpha} - 1)(2_j^{\alpha} - 1) \quad (i \neq j)$$

$$w_{ii} = 0$$

$$\theta_i = 0$$

を使って近似的な自己想起的連想記憶を実現する．
非同期的状態変化は

$$\sum_{j=1}^{N} w_{ij}x_j \rangle 0 \text{ ならば } x_i = 1$$
$$\sum_{j=1}^{N} w_{ij}x_j \langle 0 \text{ ならば } x_i = 0$$
$$\sum_{j=1}^{N} w_{ij}x_j = 0 \text{ ならば } x_i \text{はそのまま}$$

上記の規則を用いて変化させる．

3　結果

3.1　2 章の手順 1 の結果

手順 1 で記憶させるパターンを 0 と 9 に設定する．初期状態に 9 を元として変化させたパターン (図 1) を用いてネットワークを動作させると記憶させたパターンである 9 が読みだされた (図 2)．

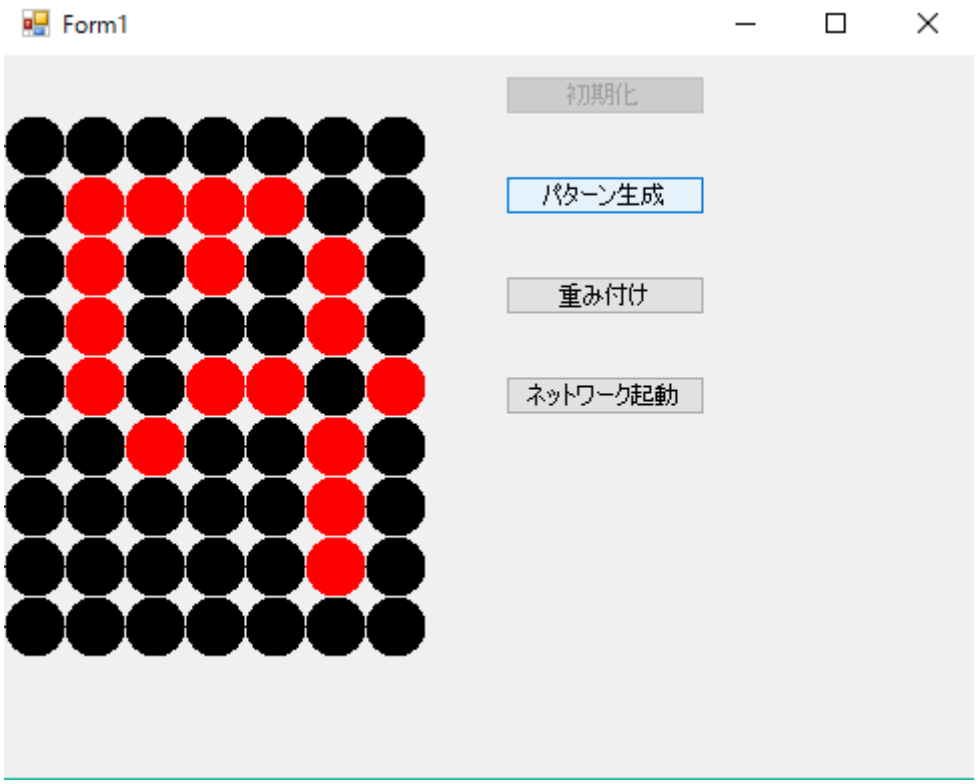

図 1　初期状態

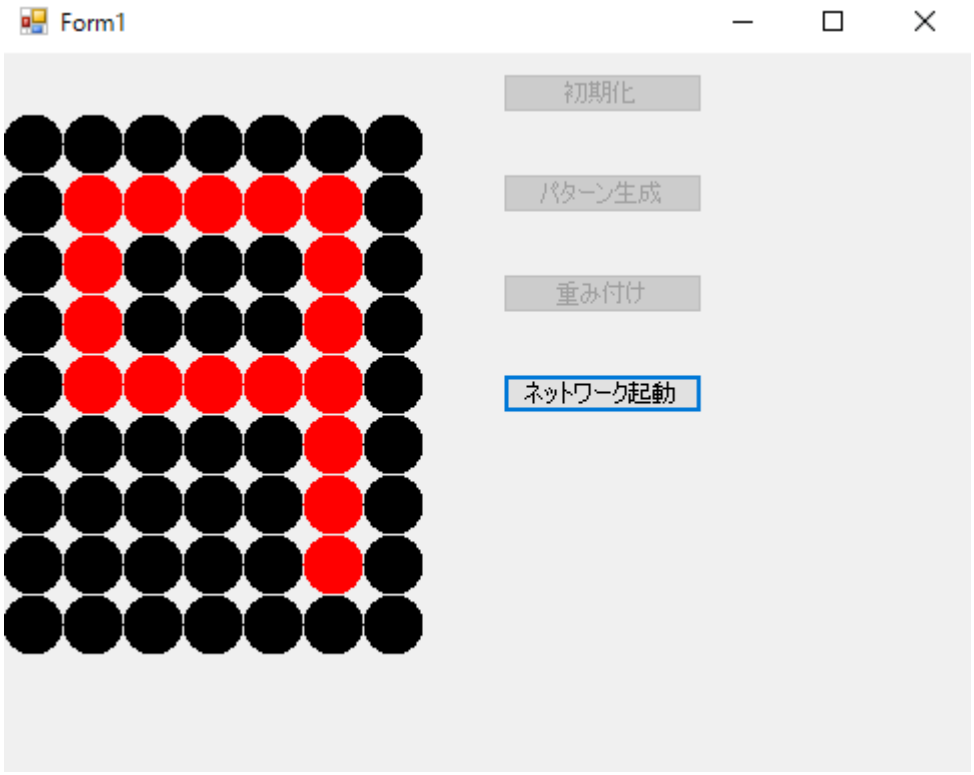

図 2　ネットワーク動作後

3.2　2 章の手順 2 の結果及び考察

手順 2 に従い，0，9 以外に 4 を追加し，初期状態を 9 を変化させたパターンを使用しネットワークを動作させた結果記憶させたパターンと一致することはなかった (図 3).
これは結合の強さを設定する際に，余計な部分に重荷が追加されたためと考えられる．つまり，元々パターンが 2 つしかなかったため結合重荷を設定しても余分な重荷が影響することがあまりなかったが，パターン数が増えていくことで重荷の値が負から正になる，正から負になるといった現象が起き，影響を与えた考えられる．またこれは，パターン数が増えていく毎にネットワークを動作させても記憶されたパターンとはかけ離れていくと考えられる．

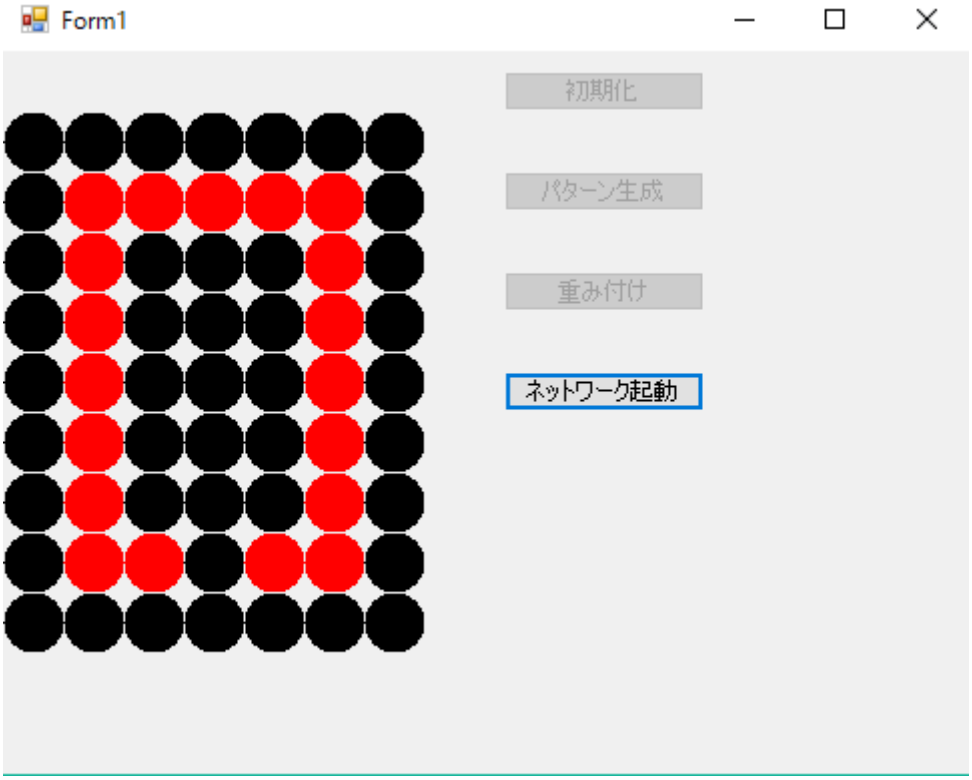

図 3　ネットワーク動作後

4　感想

思った以上に課題 I を実行するのに時間がかかってしまい，課題 II に取り掛かることが出来ず残念だった．しかし，課題 I については概ね満足のいくものが出来上がってよかった．今回は近似的に自己連想的記憶を実現したため，手順 2 のような結果になったのかと思ったため，これを安定させるためにはどうすればよいかについて考えていくことが今後の展望である．

付録 (プログラムのソースコード)

(※ソースコードは手順 1 で行ったもの．手順 2 を行うときは MakePattern 関数にパターンを追加，PatternIndex 関数に for 文を追加，PutWeight 関数の変数 k の値を変更する)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace HPNAp
{
    public partial class Form1 : Form
    {

        Button bt1;
        Button bt2;
        Button bt3;
        Button bt4;

        Random r = new Random();
        Random i = new Random();
        Random j = new Random();

        int[] num = new int[63];
        int[,] ptn = new int[9, 63];
        int[,] w = new int[63, 63];
        int [] xp = new int[63];

        int[] zero = new int[]{  0,  0,  0,  0,  0,  0,  0,
              0, +1, +1, +1, +1, +1,  0,
              0, +1,  0,  0,  0, +1,  0,
              0, +1,  0,  0,  0, +1,  0,
              0, +1,  0,  0,  0, +1,  0,
              0, +1,  0,  0,  0, +1,  0,
              0, +1,  0,  0,  0, +1,  0,
              0, +1, +1, +1, +1, +1,  0,
              0,  0,  0,  0,  0,  0,  0};


        int[] one = new int[]{  0,  0,  0,  0,  0,  0,  0,
             0,  0,  0, +1,  0,  0,  0,
             0,  0, +1, +1,  0,  0,  0,
             0,  0,  0, +1,  0,  0,  0,
             0,  0,  0, +1,  0,  0,  0,
```

```
     0,  0,  0, +1,  0,  0,  0,
     0,  0,  0, +1,  0,  0,  0,
     0, +1, +1, +1, +1, +1,  0,
     0,  0,  0,  0,  0,  0,  0};


int[] two = new int[]{  0,  0,  0,  0,  0,  0,  0,
     0, +1, +1, +1, +1, +1,  0,
     0,  0,  0,  0,  0, +1,  0,
     0,  0,  0,  0,  0, +1,  0,
     0, +1, +1, +1, +1, +1,  0,
     0, +1,  0,  0,  0,  0,  0,
     0, +1,  0,  0,  0,  0,  0,
     0, +1, +1, +1, +1, +1,  0,
     0,  0,  0,  0,  0,  0,  0};


int[] three = new int[]{  0,  0,  0,  0,  0,  0,  0,
       0, +1, +1, +1, +1, +1,  0,
       0,  0,  0,  0,  0, +1,  0,
       0,  0,  0,  0,  0, +1,  0,
       0,  0,  0, +1, +1, +1,  0,
       0,  0,  0,  0,  0, +1,  0,
       0,  0,  0,  0,  0, +1,  0,
       0, +1, +1, +1, +1, +1,  0,
       0,  0,  0,  0,  0,  0,  0};


int[] four = new int[]{ 0,  0,  0,  0,  0,  0,  0,
     0, +1,  0,  0,  0,  0,  0,
     0, +1,  0,  0,  0,  0,  0,
     0, +1,  0, +1,  0,  0,  0,
     0, +1, +1, +1, +1,  0,  0,
     0,  0,  0, +1,  0,  0,  0,
     0,  0,  0, +1,  0,  0,  0,
     0,  0,  0, +1,  0,  0,  0,
     0,  0,  0,  0,  0,  0,  0};


int[] five = new int[]{ 0,  0,  0,  0,  0,  0,  0,
```

```
     0, +1, +1, +1, +1, +1,  0,
     0, +1,  0,  0,  0,  0,  0,
     0, +1,  0,  0,  0,  0,  0,
     0, +1, +1, +1, +1, +1,  0,
     0,  0,  0,  0,  0, +1,  0,
     0,  0,  0,  0,  0, +1,  0,
     0, +1, +1, +1, +1, +1,  0,
     0,  0,  0,  0,  0,  0,  0};


int[] six = new int[]{ 0,  0,  0,  0,  0,  0,  0,
    0, +1, +1, +1, +1, +1,  0,
    0, +1,  0,  0,  0,  0,  0,
    0, +1,  0,  0,  0,  0,  0,
    0, +1, +1, +1, +1, +1,  0,
    0, +1,  0,  0,  0, +1,  0,
    0, +1,  0,  0,  0, +1,  0,
    0, +1, +1, +1, +1, +1,  0,
    0,  0,  0,  0,  0,  0,  0};


int[] seven = new int[]{ 0,  0,  0,  0,  0,  0,  0,
      0, +1, +1, +1, +1, +1,  0,
      0,  0,  0,  0,  0, +1,  0,
      0,  0,  0,  0, +1,  0,  0,
      0,  0,  0, +1,  0,  0,  0,
      0,  0, +1,  0,  0,  0,  0,
      0, +1,  0,  0,  0,  0,  0,
      0, +1,  0,  0,  0,  0,  0,
      0,  0,  0,  0,  0,  0,  0};


int[] eight = new int[]{ 0,  0,  0,  0,  0,  0,  0,
      0, +1, +1, +1, +1, +1,  0,
      0, +1,  0,  0,  0, +1,  0,
      0, +1,  0,  0,  0, +1,  0,
      0, +1, +1, +1, +1, +1,  0,
      0, +1,  0,  0,  0, +1,  0,
      0, +1,  0,  0,  0, +1,  0,
      0, +1, +1, +1, +1, +1,  0,
```

```
       0,  0,  0,  0,  0,  0,  0};


int[] nine = new int[]{ 0,  0,  0,  0,  0,  0,  0,
     0, +1, +1, +1, +1, +1,  0,
     0, +1,  0,  0,  0, +1,  0,
     0, +1,  0,  0,  0, +1,  0,
     0, +1, +1, +1, +1, +1,  0,
     0,  0,  0,  0,  0, +1,  0,
     0,  0,  0,  0,  0, +1,  0,
     0,  0,  0,  0,  0, +1,  0,
     0,  0,  0,  0,  0,  0,  0};

public Form1()
{
    InitializeComponent();

    this.Width = 500;
    this.Height = 400;

    bt1 = new Button();
    bt1.Text = "初期化";
    this.bt1.Size = new Size(100, 20);
    this.bt1.Location = new Point(250, 10);
    bt1.Click += new EventHandler(bt1_Click);

    bt2 = new Button();
    bt2.Text = "パターン生成";
    this.bt2.Size = new Size(100, 20);
    this.bt2.Location = new Point(250, 60);
    bt2.Click += new EventHandler(bt2_Click);

    bt3 = new Button();
    bt3.Text = "重み付け";
    this.bt3.Size = new Size(100, 20);
    this.bt3.Location = new Point(250, 110);
    bt3.Click += new EventHandler(bt3_Click);

    bt4 = new Button();
    bt4.Text = "ネットワーク起動";
```

```
    this.bt4.Size = new Size(100, 20);
    this.bt4.Location = new Point(250, 160);
    bt4.Click += new EventHandler(bt4_Click);

    bt1.Parent = this;
    bt2.Parent = this;
    bt3.Parent = this;
    bt4.Parent = this;
    this.Paint += frame_paint;

}

private void bt1_Click(object sender, EventArgs e)
{
    //throw new NotImplementedException();

    MakePattern(nine, ref num);
    Refresh();
    bt1.Enabled = false;

}

private void bt2_Click(object sender, EventArgs e)
{
    //throw new NotImplementedException();

    PatternIndex(nine, zero, ref ptn);
    bt2.Enabled = false;

}

private void bt3_Click(object sender, EventArgs e)
{
    //  throw new NotImplementedException();

    PutWeight(ptn, ref w);
    bt3.Enabled = false;
}

private void bt4_Click(object sender, EventArgs e)
```

```
{
    Timer tm = new Timer();
    tm.Interval = 250;
    tm.Start();
    tm.Tick += new EventHandler(timer_Click);
}

private void timer_Click(object sender, EventArgs e)
{
    //throw new NotImplementedException();

    RunNetwork(ref num);
    Refresh();
}

private void frame_paint(object sender, PaintEventArgs e)
{
    //throw new NotImplementedException();

    Graphics g = e.Graphics;
    SolidBrush sb1 = new SolidBrush(Color.Black);
    SolidBrush sb2 = new SolidBrush(Color.Red);
    int w = 0;
    int h = 0;

    for (int i = 0; i < 63; i++)
    {
        if (i % 7 == 0)
        {

            w = 0;
            h++;

        }

        if (num[i] == 0)
        {

            g.FillEllipse(sb1, 30 * w, 30 * h, 30, 30);
            w++;
```

```
        }
        else if (num[i] == 1)
        {

            g.FillEllipse(sb2, 30 * w, 30 * h, 30, 30);
            w++;

        }

    }


}

//初期状態を生成するための関数
public void MakePattern(int [] number, ref int[] defaultP)
{

    int rndm;

    for (int i = 0; i < 63; i++)
    {

        defaultP[i] = number[i];

        rndm = r.Next(0, 9);
        if (rndm == 0 && defaultP[i] == 0)
        {
            defaultP[i] = 1;
        }
        else if (rndm == 0 && defaultP[i] == 1)
        {
            defaultP[i] = 0;
        }

    }

}
```

```
// パターンとして使用するデータをまとめる関数
public void PatternIndex(int[] num1, int[] num2, /*int [] num3, */ref int[,] pattern)
{
    for (int a = 0; a < 63; a++)
    {
        pattern[0, a] = num1[a];
    }
    for (int b = 0; b < 63; b++)
    {
        pattern[1, b] = num2[b];
    }
/*     for (int c = 0; c < 63; c++)
    {
        pattern[2, c] = num3[c];
    } */
}

// 結合の強さを設定するための関数
public void PutWeight(int[,] pattern, ref int[,] w)
{

    for (int ni = 0; ni < 63; ni++)
    {
        for (int nj = 0; nj < 63; nj++)
        {

            if (ni == nj)
            {
                w[ni, nj] = 0;
            }
            else if (ni != nj)
            {
                for (int k = 0; k < 2; k++)
                {
                    w[ni, nj] += (2 * pattern[k, ni] - 1) * (2 * pattern[k, nj] - 1);
                }
            }
        }
    }
```

```
        }

        // 非同期的状態変化を行うための関数
        public void RunNetwork(ref int[] x)
        {

            int ni;

            ni = i.Next(0, 62);

            for (int nj = 0; nj < 63; nj++)
            {
                xp[ni] += w[ni, nj] * x[nj];
            }

            if (xp[ni] > 0)
            {
                x[ni] = 1;
            }
            else if (xp[ni] < 0)
            {
                x[ni] = 0;
            }

          //  Console.WriteLine(ni.ToString()+":"+xp[ni]);

        }

    }
}
```