ニューラルネット

Report2

学籍番号 045713C:大城和也

提出日:平成 18 年 07 月 31 日 (月)

1 課題内容

http://www.eva.ie.u-ryukyu.ac.jp/ tnal/Job/NN/Readme.html を参照し以下をレポートとして提出すること.

Q1 プログラムの概説

Q2 実行結果の解説

- 得られた結果の検証
- パラメータ, 乱数シード変更により複数回実行すること
- と指数や配置を換えて実行してみること

Q3 Hopfield Net アルゴリズムの改良についての考察

2 プログラム

main 関数

```
/*
 * Traveling sales person problem.
 * example;
 * > tsp 1.0 1.0 2.0 394
 *
 */

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

#define CityNo    9

#define rnd()     ( (double)rand() / RAND_MAX )


double distance[CityNo][CityNo];
double unitout[CityNo][CityNo], Acoef, Bcoef, Dcoef;

/* city の配置座標 */
struct { double x, y; } cityxy[CityNo] =
{ { 3.0, 4.0 }, { 2.0, 7.0 }, { 4.0, 7.0 }, { 5.0, 5.0 },
    { 5.0, 3.0 }, { 4.0, 1.0 }, { 2.0, 1.0 }, { 1.0, 3.0 },
    { 1.0, 5.0 } };

main( argc, argv )
int argc;
char *argv[];
{
  int      no;
  double   en;
  unsigned seed;
  void     initialize(), update_state(), display_state();
  double   energy();

  /* Usage */
  if ( argc <= 1 )
    {
      printf(" Usage: tsp Acoef Bcoef Dcoef seed\n");
      exit(0);
    }
  /* 引数を浮動小数点に変換 */
  Acoef = atof( argv[1] );
  Bcoef = atof( argv[2] );
  Dcoef = atof( argv[3] );
  seed  = atof( argv[4] );

  srand( seed );           // rand の設定

  initialize();
  display_state( 0 );
  update_state();

  /* 学習の繰り返し */
  for( no = 1; no < 50; no++ )
    {
      update_state();
      display_state( no );
      en = energy();
      printf(" Energy = %f\n",en);
    }
}
```

main 関数では引数の処理．rand() の seed 設定．初期化を行なった後，update_state() 関数を繰り返すことで学習を行なっている．

初期化

```
void initialize()
{
  int    i, j;
  double dtotal;

  /* 都市間距離の算出 */
  dtotal = 0.0;
  for( i = 0; i < CityNo; i++ )
    for( j = 0; j < CityNo; j++ )
      {
distance[i][j] = 0.1 *
  sqrt( (cityxy[i].x - cityxy[j].x)*(cityxy[i].x - cityxy[j].x) +
       (cityxy[i].y - cityxy[j].y)*(cityxy[i].y - cityxy[j].y));
dtotal += distance[i][j];
      }

  /* 距離を割合に変換 */
  for( i = 0; i < CityNo; i++ )
    for( j = 0; j < CityNo; j++)
      distance[i][j] = 10.0*distance[i][j]/dtotal;

  /* ユニット間出力の初期値をランダムで設定 */
  for( i = 0; i < CityNo; i++ )
    for( j = 0; j < CityNo; j++ )
      unitout[i][j] = rnd();
}
```

ここでは初期化を行なっている．都市間の距離を求め，その距離を全体の割合として変換する．また，ユニットの出力をランダムで決めておくといった処理を行なっている．

値の更新 (学習)

```
void update_state()
{
  int    i, j, n, m, jm, jp;
  double un, unitin;
  double aterm, bterm, dterm;

  for( i = 0; i < CityNo; i++ )
    for( j = 0; j < CityNo; j++ )
      {
aterm = bterm = dterm = 0.0;
for( n = 0; n < CityNo; n++)
  aterm += unitout[i][n];
aterm = -Acoef*( aterm - unitout[i][j] );

for( m = 0; m < CityNo; m++)
  bterm += unitout[m][j];
bterm = -Bcoef*( bterm - unitout[i][j] );

if( j-1 == -1 ) jm = CityNo - 1;
else jm = j - 1;
if( j+1 == CityNo ) jp = 0;
else jp = j + 1;
for( m = 0; m < CityNo; m++)
  dterm += distance[i][m]*(unitout[m][jp] + unitout[m][jm]);
dterm = -Dcoef*dterm;

unitin = aterm + bterm + dterm + Acoef + Bcoef;
unitout[i][j] = 0.5*( 1.0 + tanh( unitin/0.5 ) );
      }
}
```

ここで，Hopfield net を使った学習を行なっている．

出力

```
void display_state( n )
int n;
{
  int    i, j;

  printf("    ###   Sequence     Cycle : %4d   ###\nCity ",n );
  for( i = 0; i < CityNo; i++ )
    printf(" %4d  ",i+1 );
  printf("\n");
  for( i = 0; i < CityNo; i++ )
    {
      printf("%4d  ",i+1 );
      for( j = 0; j < CityNo; j++ )
{
  printf("%5.2f  ",unitout[i][j] );
}
      printf("\n");
    }
}
```

ここでは、出力の動作を行なっている．出力するのは学習 n 回目の unit の出力である．

全体エネルギーの算出

```
double energy()
{
  int    i, j, m, jp, jm;
  double term1, term2, term3;

  term1 = term2 = term3;

  for( i = 0; i < CityNo; i++ )
    for( j = 0; j < CityNo; j++ )
      for( m = 0; m < CityNo; m++ )
if( m != j ) term1 += unitout[i][j]*unitout[i][m];
  term1 = 0.5*Acoef*term1;

  for( j = 0; j < CityNo; j++ )
    for( i = 0; i < CityNo; i++ )
      for( m = 0; m < CityNo; m++ )
if( m != i ) term2 += unitout[i][j]*unitout[m][j];
  term2 = 0.5*Bcoef*term2;

  for( i = 0; i < CityNo; i++ )
    for( j = 0; j < CityNo; j++ )
      {
if( j-1 == -1 ) jm = CityNo - 1;
else jm = j - 1;
if( j+1 == CityNo ) jp = 0;
else jp = j + 1;
       for( m = 0; m < CityNo; m++ )
 term3 += distance[i][m]*unitout[i][j]*(unitout[m][jp]+unitout[m][jm]);
      }
  term3 = 0.5*Dcoef*term3;

  return( term1 + term2 + term3 );
}
```

全体のエネルギーの値の算出を行なっている．

3　実行結果

Aconf と Bconf は 1.0,Dconf をと 2.0 し，seed 値を 394 としたときの実行結果を以下に示す.

始めに示すのが initialize() が終了した後の出力である．この出力は都市が 1〜9 が何番目に通るかを示しているものである．縦の 1〜9 が都市を表し，横の 1〜9 が順序となっている．ここでは初期値であるために，一つの都市に対していくつかの順序に発火している.

初期状態

```
reading args: Acoef=1.000000, Bcoef=1.000000, Dcoef=2.000000, seed=394
    ###   Sequence     Cycle :    0   ###
City     1      2      3      4      5      6      7      8      9
   1   0.00   0.83   0.71   0.71   0.91   0.27   0.54   0.47   0.64
   2   0.27   0.10   0.65   0.40   0.62   0.06   0.70   0.43   0.03
   3   0.07   0.34   0.49   0.59   0.06   0.59   0.39   0.61   0.23
   4   0.64   0.90   0.27   0.67   0.31   0.41   0.70   0.02   0.32
   5   0.26   0.03   0.47   0.94   0.33   0.12   0.73   0.82   0.70
   6   0.64   0.86   0.57   0.45   0.95   1.00   0.23   0.68   0.31
   7   0.55   0.60   0.69   0.68   0.23   0.86   0.43   0.78   0.42
   8   0.70   0.74   0.63   0.86   0.75   0.42   0.06   0.28   0.35
   9   0.17   0.04   0.72   0.99   0.75   0.73   0.04   0.46   0.16
```

以下は学習を行なったのちの最終の出力である．最終出力では都市に対して発火しているのは一つの順だけとなっていることが見て取れる.

最終表示

```
    ###   Sequence     Cycle :   99   ###
City     1      2      3      4      5      6      7      8      9
   1   0.03   0.03   0.03   0.06   0.98   0.06   0.03   0.03   0.03
   2   0.06   0.99   0.08   0.03   0.01   0.01   0.00   0.00   0.01
   3   0.02   0.07   0.99   0.08   0.02   0.01   0.00   0.00   0.01
   4   0.01   0.02   0.06   0.99   0.10   0.02   0.01   0.01   0.01
   5   0.01   0.01   0.01   0.02   0.10   0.99   0.06   0.02   0.01
   6   0.01   0.00   0.00   0.01   0.02   0.08   0.99   0.07   0.02
   7   0.01   0.00   0.00   0.01   0.01   0.03   0.08   0.99   0.06
   8   0.06   0.02   0.01   0.01   0.01   0.02   0.02   0.07   0.99
   9   0.99   0.07   0.02   0.02   0.01   0.01   0.01   0.02   0.06
 Energy = 6.627321
```

この時の TSP の回答は図 1 のようになることがわかる．また、このときのエネルギーは図 2 の様に推移している．5 回をすぎるあたりでは既に収束していることがわかる.

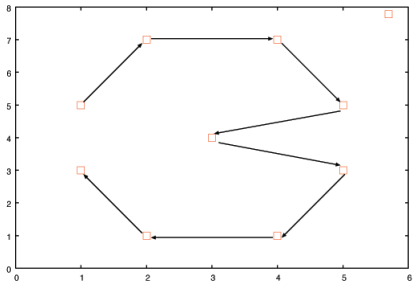

図 1: TSP 経路

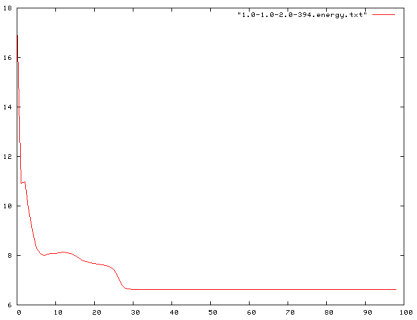

図 2: エネルギー推移

3.1 パラメータの変更

Seed 値を 394 に保ちつつ Acoef，Bcoef，Dcoef の値を変更し，計算を行った．変更は図 3 が Acoef=2，Bcoef=2，Dcoef=1。図 4 が Acoef=1, Bcoef=2,Dcoef=4 として行なった。

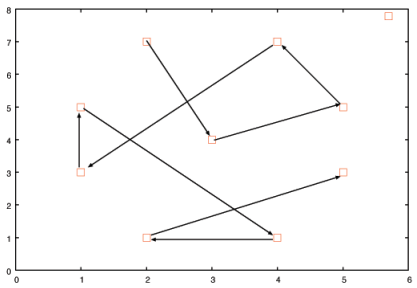

図 3: 2.0 2.0 1.0 394

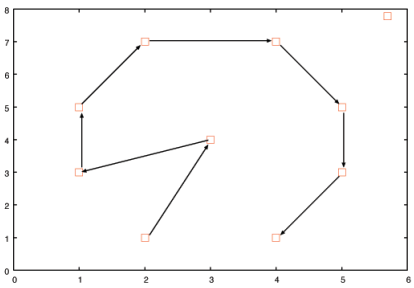

図 4: 1.0 2.0 4.0 394

また，この時のエネルギーの推移は以下のようになった．

パラメータを変更すると，学習の様子が大きく変化する．図 3 は最適解には遠く，局所解へと陥ったと思われる．一方，図 4 は初めとは違った最適解が導かれている．いくつかのパラメータで調べたが，パラメータ Acoef と Bcoef を大きくし，Dcoef を小さくすると，局所解になることが多く，Dcoef を大きくすると最適解に近づくが，TSP として間違った解 (同じ都市を回るなど) が見られることが多くなった．これより，パラメータの設定は相互の作用によって変化することがわかる．

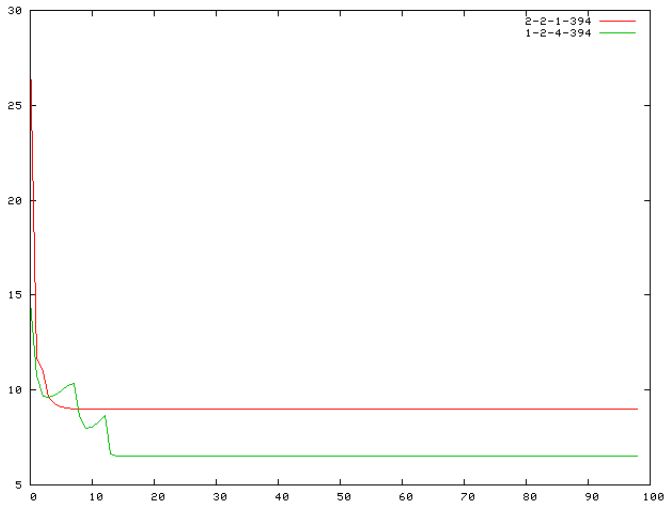

図 5:

3.2 シード値の変更

今回は始めに示したパラメータの設定，Acoef=1.0 Bcoef=1.0 Dcoef=2.0 としてシード値を 300，100 と変化させてみた．

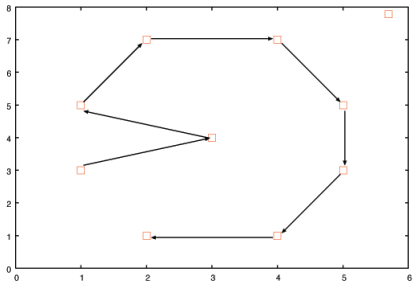

図 6: 1.0 1.0 2.0 300

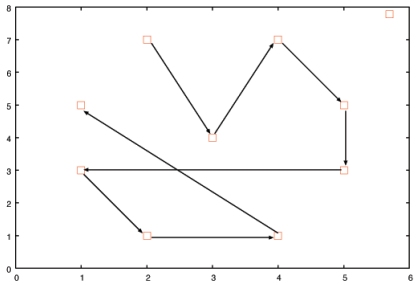

図 7: 1.0 1.0 2.0 100

図 6 は形は違ってもしっかり解が求められている．図 7 は一部が違うものの学習していることは伺える．これをパラメータを変化したときと比べてみるとこっちの方が安定しており，適切なパラメータを選ぶことが学習をうまくさせることに繋がっていることがわかる．

3.3　都市数，配置の変更

次に都市の数，及びその配置を変更した時の Hopfield net での学習を行なってみる．図 8 は都市の変更で，菱形の形をとるようにしてみた．また，図 9 は都市数を 9 個から 12 個に変えたものである．

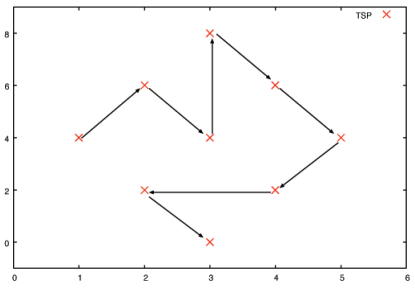

図 8: 都市の配置の変更

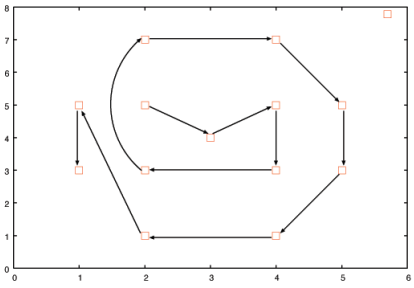

図 9: 都市数の増加

都市の位置を変更，あるいは増加してもある程度の解は導きだされているようだ．図??の推移をみると都市数を増加したときのエネルギーは大きい．これは全体の距離自体が長くなったためだと思われる．

4　アルゴリズムの改良についての考察

Hopfield Net アルゴリズムは最急降下法を扱っているために局所解へ陥りやすい．これを考慮したアルゴリズムがボルツマンマシンと言われる方法で，Hopfield net のアルゴリズムにゆらぎ温度と呼ばれる値を用いることで，局所解に陥ることを少なくすることができる．あるいは他のアルゴリズムと併用することで防ぐことが出来る．

また，Hopfield Net は問題が対称結合ネットワークの形になることが必要なので，構造化が難しく．適応できる問題が，誤差伝播などの方法に比べ少ないなどの欠点もある．

参考文献

[1] Neural Network and AL Terminology
”http://staff.aist.go.jp/utsugi-a/Lab/term-j.html”

[2] ニューラルネットワークとは
”http://obog.ome.meisei-u.ac.jp/ tuchiyaob/new.html”