1

```
% NPC（学習プレイヤー）が行動を選択するためのプログラム

function [a]=action_test(policy,step,state3)
  % 最初のステップでは1マス目を選択
  if(step == 1)
    a=1;
  else

    % 政策policyに従いランダムに行動を選択
    while(1)
      random = rand;

      cprob = 0;
      for a=1:9
        cprob = cprob + policy(a);
        if(random < cprob)
          break;
        end
      end

      % 既にマスが埋まっていないかどうかを確認
      if state3(a)==0
        break;
      end
    end
  end

  % 書いている間は読み込みをロックする
  lock = fopen('./lock.m', 'w');

  % 行動を出力
  fp = fopen('./action.txt','w');
  fprintf(fp,'%d',a-1);
  fclose(fp);
  delete fp;

  % 書き込みが終わったのでロック解除
  fclose(lock);
  %munlock('./lock.m');
```

```
% NPC（学習プレイヤー）の行動選択と，学習する際の対戦相手となる人工知能プレイヤーのプログラム

function [action,reward,state3,fin]=action_train(policy,step,state3)
  reward = 0;

  % 学習プレイヤー
  % 最初のステップでは1マス目を選択
  if(step == 1)
    a=1;
  else
    % 政策policyに従いランダムに行動を選択
    while(1)
      random = rand;

      cprob = 0;
      for a=1:9
        cprob = cprob + policy(a);
        if(random < cprob)
          break;
        end
      end

      % 既にマスが埋まっていないかどうかを確認
      if state3(a)==0
        break;
      end
    end
  end

  action = a;
  state3(a) = 2;
  fin = check(state3);
  if(fin == 2)
    reward = 10;
    return;
  elseif(fin == 3)
    reward = 0;
    return;
  end

  % 人工知能プレイヤー
  reach = 0;
  pos = [1 2 3; 4 5 6; 7 8 9; 1 4 7; 2 5 8; 3 6 9; 1 5 9; 3 5 7];
  for i=1:max(size(pos))
    val = sum(state3(pos(i,:)));
    num = size(find(state3(pos(i,:))==0),2);
    if(val==2 & num==1)
      a = pos(i,state3(pos(i,:))==0);
      reach = 1;
      break;
    end
  end

  if(reach==0)
    while(1)
```

```
    a = floor(rand*9)+1;

    if state3(a)==0
      break;
    end
  end
end

state3(a) = 1;

fin = check(state3);
if(fin == 1)
  reward = -10;
  return;
elseif(fin==3)
  reward = 0;
  return;
end
```

```
% ゲームの勝敗をチェックするためのプログラム
% fin: 0-ゲーム続行, 1-人工知能プレイヤーの勝ち, 2-学習プレイヤーの勝ち, 3-引き分け

function fin=check(s)
  pos = [1 2 3; 4 5 6; 7 8 9; 1 4 7; 2 5 8; 3 6 9; 1 5 9; 3 5 7];

  for i=1:max(size(pos))
    val = prod(s(pos(i,:)));

    % 人工知能プレイヤーの勝ち
    if(val==1)
      fin = 1;
      return;

    % 学習プレイヤーの勝ち
    elseif(val == 8)
      fin = 2;
      return;
    end
  end

  % 引き分け
  if(prod(s))
    fin = 3;
    return;
  end

  % ゲーム続行
  fin = 0;
```

```
% 状態ベクトルから，状態スカラーへ変換するためのプログラム

function [state]=encode(state)
  % 8通りの変換
  convert = [1,2,3,4,5,6,7,8,9; %元の状態
             3,2,1,6,5,4,9,8,7; %変換(2)
             7,4,1,8,5,2,9,6,3; %変換(3)
             1,4,7,2,5,8,3,6,9; %変換(4)
             9,8,7,6,5,4,3,2,1; %変換(5)
             7,8,9,4,5,6,1,2,3; %変換(6)
             3,6,9,2,5,8,1,4,7; %変換(7)
             9,6,3,8,5,2,7,4,1  %変換(8)
           ];

  % 3進数から10進数へと変換するためのベクトル
  power = [3^8 3^7 3^6 3^5 3^4 3^3 3^2 3^1 3^0];

  % stateを8種類の変換を加えてから10進数へと変換する
  for i=1:size(convert,1)
    cands(i) = sum(state(convert(i,:))*power');
  end

  % 8個の候補のうち一番小さいものを選ぶ
  state = min(cands)+1;
```

```
function Q=MonteCarloPolicyIteration(L,M,options)
  nstates = 3^9;   % 状態数
  nactions = 9;    % 行動数
  T = 5;           % 最大ステップ数

  % Q関数の初期化
  Q=zeros(nstates,nactions);
  Q=sparse(Q);

  % 政策反復
  for l=1:L
    visits = ones(nstates,nactions);    % (s,a)の出現回数
    results = zeros(M,1);               % ゲームの結果
    rand('state',1);                    % seedの初期化

    % エピソード
    for m=1:M
      state3 = zeros(1,9);

      % ステップ
      for t=1:T

        % 状態のエンコード
        state = encode(state3);

        % 政策の生成
        policy = zeros(1,nactions);

        switch(options.pmode)
          case 1 % greedy
            [v,a] = max(Q(state,:));
            policy(a) = 1;

          case 2 % e-greedy
            [v,a] = max(Q(state,:));
            policy = ones(1,nactions)*options.epsilon/nactions;
            policy(a) = 1-options.epsilon+options.epsilon/nactions;

          case 3 % softmax
            policy=exp(Q(state,:)/options.tau)/sum(exp(Q(state,:)./options.tau));
        end

        % 行動の選択および実行
        [action,reward,state3,fin] = action_train(policy,t,state3);

        % 状態, 行動, 報酬, 出現回数の更新
        states(m,t)      = state;
        actions(m,t)     = action;
        rewards(m,t)     = reward;
        visits(state,action)     = visits(state,action) + 1;

        % ゲーム終了
        if(fin>0)
          results(m) = fin;
```

```
          % 割引き報酬和の計算
          drewards(m,t) = rewards(m,t);
          for pstep=t-1:-1:1
            drewards(m,pstep) = options.gamma * drewards(m,pstep+1);
          end
          break;
        end
      end
    end

    % 状態行動価値関数の計算
    Q=zeros(nstates,nactions);
    Q=sparse(Q);
    for m=1:M
      for t=1:size(states,2)
        s = states(m,t);
        a = actions(m,t);
        if(s==0)
          break;
        end
        Q(s,a) = Q(s,a) + drewards(m,t);
      end
    end

    Q = Q./visits;

    % 勝率の計算
    rate(l) = size(find(results==2),1)./M;

    % 標準出力
    fprintf(1,'%d) Win=%d/%d, Draw=%d/%d, Lose=%d/%d¥n',l, size(find(results==2),1),M,size(find↙
(results==3),1),M,size(find(results==1),1),M);
    % fflush(stdout);
  end

  %  グラフの出力
  figure(1)
  clf
%  axes('FontSize',15,'LineWidth',2.0);
  games = M:M:M*L;
  g=plot(games,rate);
  set(g,'LineWidth',2);
  g=xlabel('ゲーム数');
  set(g,'FontSize',14);
  g=ylabel('勝率');
  set(g,'FontSize',14);
  axis([M,M*L,0.4,1])
```

```
% NPC（学習プレイヤー）が自分の状態を把握するためのプログラム

function [state, state3, fin]=observe_test
  % 状態, 報酬, ゲームの状況をファイルから読み込む
  fp=fopen('./state.txt','r');
  while(fp == -1)
    pause(0.01);
  end
%{
  % ロックファイルがあるなら書き込み中なので待つ
  while(exist('lock.m'))
    pause(0.01);
  end
%}
  % 状態
  [tok,rem]=strtok(fgets(fp),':');
  state3(1) = str2num(tok);
  cnt = 2;
  while(true)
    [tok,rem]=strtok(rem,':');
    if isspace(tok(1)) == 1
        break;
    end
    state3(cnt) = str2num(tok);
    cnt = cnt + 1;
  end

  % ゲームの状況
  fin = str2num(fgets(fp));

  fclose(fp);
  %munlock('./state.txt');
  %====================

  %状態のエンコード（3進数から10進数）
  state = encode(state3);
```

```
function Q=Qlearning(M,options)
  nstates = 3^9;          % 状態数
  nactions = 9;       % 行動数
  results = zeros(M,1);   % 勝敗結果
  eM =1000;          % 評価を行うエピソード数

  % Q関数の初期化
  Q=zeros(nstates,nactions);

  for m=1:M
    rand('state',mod(m,eM))
    t = 1;
    state3 = zeros(1,9);

    while(1)
      % 状態, 報酬, ゲーム状況の観測
      state = encode(state3);

      %====================
      % 政策の生成
      policy = zeros(1,nactions);

      switch(options.pmode)
        case 1 % greedy
          [v,a] = max(Q(state,:));
          policy(a) = 1;

        case 2 % e-greedy
          [v,a] = max(Q(state,:));
          policy = ones(1,nactions)*options.epsilon/nactions;
          policy(a) = 1-options.epsilon+options.epsilon/nactions;

        case 3 % softmax
          policy=exp(Q(state,:)/options.tau)/sum(exp(Q(state,:)./options.tau));
      end
      %====================

      % 行動の選択および実行
      [action,reward,state3,fin] = action_train(policy,t,state3);

      %====================
      % Q関数の更新 (Q学習)
      % 1ステップ前の状態, 行動のQ値を更新
      if t > 1
        Q(pstate,paction) = Q(pstate,paction) + options.alpha*(reward - Q(pstate,paction) + options.
gamma*max(Q(state,:)));
      end

      % ゲーム終了
      if(fin>0)
        results(m) = fin;
        break;
      end

      % 状態と行動の記録
```

```
      pstate = state;
      paction = action;

      t = t + 1;
    end

    if(mod(m,eM)==0)
      fprintf(1,'%d) Win=%d/%d, Draw=%d/%d, Lose=%d/%d¥n',m, size(find(results(m-eM+1:m)==2),1),eM,size↙
(find(results(m-eM+1:m)==3),1),eM,size(find(results(m-eM+1:m)==1),1),eM);
    end

    fflush(stdout);
  end

  %  グラフの出力
  results2(results~=2)=0;
  results2(results==2)=1;
  res =reshape(results2,eM,M/eM);
  rate = sum(res)./eM;
  figure(3)
  clf
%  axes('FontSize',15,'LineWidth',2.0);
  games = eM:eM:M;
  g=plot(games,rate);
  set(g,'LineWidth',2);
  g=xlabel('ゲーム数');
  set(g,'FontSize',14);
  g=ylabel('勝率');
  set(g,'FontSize',14);
  axis([eM,M,0.4,1])
```

```
// NPC（学習プレイヤー）と対戦して遊ぶための，GUIのソースプログラム
//--------------------------------------------------------------------------------------------
/*
    ¥brief  sanmoku game interface
    ¥author Tsubasa Fukuyama
    ¥date   Mar.31 2008
*/
//--------------------------------------------------------------------------------------------
#define _CRT_SECURE_NO_WARNINGS
#define WIN32
#include "FL/Fl.H"
#include "FL/Fl_Window.H"
#include "FL/Fl_Box.H"
#include "FL/Fl_Button.H"
#include "FL/Fl_Menu_Bar.H"
#include "FL/Fl_Menu_Item.H"
#include "FL/Fl_ask.H"
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <Windows.h>

// ボタン配列
Fl_Button *b[9];

// 状態配列：0-無し，1-人間（O），2-学習プレイヤー（X）
int state[9] = {0, 0, 0, 0, 0, 0, 0, 0, 0};

//先攻，1-学習プレイヤー，0-人間
int player = 1;

// ゲームの状況：0-続行，1-終了
int gameflag = 0;

Fl_Window *window = new Fl_Window(170,190,"OX Game");

//--------------------------------------------------------------------------------------------
/*
    ¥brief 状態をファイルstate.txtに出力.
*/
void output(){
  int k;
  FILE *fp, *lock;

  // ロックファイルの生成
  lock = fopen("lock.m", "w");

  // 状態ベクトルを出力
  fp = fopen("state.txt","w");
  for(k=0;k<9;k++){
    fprintf(fp,"%d:",state[k]);
  }

  // ゲームの状況を出力
  fprintf(fp,"¥n%d",gameflag);

  fclose(fp);
  fclose(lock);

  // ロックファイルの削除
  remove("lock.m");
}
//--------------------------------------------------------------------------------------------

//--------------------------------------------------------------------------------------------
/*
    ¥brief 勝敗確認.
*/
int judge(int pos, int turn) {
  int i,j,k;
  char message[256] = "";

  i = (int)pos / 3;
  j = pos % 3;

  for(k=0; k<3; k++)
    if(state[i*3+k] != turn)
      break;

  if(k == 3) {
    for(k=0; k<3; k++)
      b[i*3+k]->color(turn==1?FL_RED:FL_BLUE);
      window->redraw();

    sprintf(message, "%s Win%s!!", turn==1?"You":"COM", turn==1?"":"s");
    fl_alert(message);
    return 1;
  }

  for(k=0; k<3; k++)
    if(state[j+k*3] != turn)
      break;

  if(k == 3) {
    for(k=0; k<3; k++)
      b[j+k*3]->color(turn==1?FL_RED:FL_BLUE);
    window->redraw();

    sprintf(message, "%s Win%s!!", turn==1?"You":"COM", turn==1?"":"s");
    fl_alert(message);
    return 1;
  }

  if(i==j) {
    for(k=0; k<3; k++)
```

```
106       if(state[4*k] != turn)
107         break;
108     if(k == 3) {
109       for(k=0; k<3; k++)
110         b[4*k]->color(turn==1?FL_RED:FL_BLUE);
111       window->redraw();
112 
113       sprintf(message, "%s Win%s!!", turn==1?"You":"COM", turn==1?"":"s");
114       fl_alert(message);
115       return 1;
116     }
117   }
118   if((2-i)==j) {
119     for(k=0; k<3; k++)
120       if(state[2*(k+1)] != turn)
121         break;
122     if(k == 3) {
123       for(k=0; k<3; k++)
124         b[2*(k+1)]->color(turn==1?FL_RED:FL_BLUE);
125       window->redraw();
126 
127       sprintf(message, "%s Win%s!!", turn==1?"You":"COM", turn==1?"":"s");
128       fl_alert(message);
129       return 1;
130     }
131   }
132   return 0;
133 }
134 //--------------------------------------------------------------------------------------------
135 
136 
137 //--------------------------------------------------------------------------------------------
138 /*
139     ¥brief 行動をファイルaction.txtから入力.
140 */
141 void action(void*) {
142   FILE *fp, *lock;
143   errno_t error;
144   errno_t error2;
145   char cdir[255];
146   GetCurrentDirectory(255, cdir);
147   // action.txtがあってもロックファイルがある間は書き込み中なので待つ
148   if((error=fopen_s(&fp,"action.txt","r")) != NULL || (error2=fopen_s(&lock, "lock.m", "r")) != NULL){
149     Fl::repeat_timeout(2.0,action);
150   }else{
151 
152     // 行動のマスに×を置き，状態を2とする
153     char str[10];
154     fgets(str,10,fp);
155     int num = atoi(str);
156     fclose(lock);
157 
158     b[num]->label("X");
159     state[num] = 2;
160     player = 0;
161     fclose(fp);
162     remove("action.txt");
163 
164     if(gameflag = judge((int)num, 2)){
165       output();
166       return;
167     }
168 
169   }
170 }
171 //--------------------------------------------------------------------------------------------
172 
173 //--------------------------------------------------------------------------------------------
174 /*
175     ¥brief マス上のボタンがクリックされた時のイベント.
176 */
177 void button_change(Fl_Widget* o, void *i) {
178   Fl_Button *button = (Fl_Button *)o;
179   int num;
180 
181   if(gameflag)
182     return;
183 
184   // 学習プレイヤーの順番の時は，何もしないで戻す
185   if(player != 0)
186     return;
187 
188   // 押されたボタンをマーク（○）に変更
189   if(state[(int)i] != 0)
190     return;
191   button->label("O");
192   state[(int)i] = 1;
193 
194   // ゲーム終了の場合は，状態を出力して終わり
195   if(gameflag = judge((int)i, 1)){
196     output();
197     return;
198   }
199 
200   // ゲーム続行の場合は，次は学習プレイヤーの番
201   for(num=0; num<9; num++)
202     if(state[num] == 0)
203       break;
204   if(num == 9)
205     return;
206 
207   // 状態を出力
208   output();
209   player = 1;
210 
```

```
211   // 行動ファイルを削除
212   remove("action.txt");
213
214   // タイマー
215   Fl::add_timeout(1.0,action);
216 }
217 //--------------------------------------------------------------------------------
218
219 //--------------------------------------------------------------------------------
220 /*
221     ¥brief resetボタンがクリックされた時のイベント.
222 */
223 void reset(Fl_Widget* o, void *i) {
224   int k;
225
226   // ボタンの初期化
227   for(k=0; k<9; k++) {
228     state[k] = 0;
229     b[k]->label("");
230     b[k]->color(FL_WHITE);
231   }
232
233   // フラグの初期化
234   gameflag = 0;
235
236   // 状態出力
237   output();
238
239   // タイマー開始
240   Fl::add_timeout(1.0,action);
241 }
242 //--------------------------------------------------------------------------------
243
244 //--------------------------------------------------------------------------------
245 /*
246     ¥brief 三目並べゲームのメイン.
247 */
248 int main(int argc, char **argv) {
249   remove("action.txt");
250   remove("state.txt");
251
252   // メニューバー
253   Fl_Menu_Bar *menubar = new Fl_Menu_Bar(0,0,300,20,0);
254   int k;
255
256   // ボタンの配置
257   for(k=0; k<9; k++) {
258     b[k] = new Fl_Button(5+55*(k%3),25+55*(int)(k/3),50,50,"");
259     b[k]->labelsize(36);
260     b[k]->labeltype(FL_SHADOW_LABEL);
261     b[k]->callback(button_change, (void *)k);
262     b[k]->color(FL_WHITE);
263   }
264
265   // メニューの追加
266   menubar->add("File/Restart", "", reset, 0);
267   window->end();
268   window->show(argc, argv);
269
270   // 学習プレイヤーが先行の場合
271   if(player==1){
272     output();
273     Fl::add_timeout(1.0,action);
274   }
275
276   return Fl::run();
277 }
```

```
function Q=SARSAPolicyIteration(L,M,options)
  nstates = 3^9;   % 状態数
  nactions = 9;    % 行動数
  T = 5;           % 最大ステップ数

  % Q関数の初期化
  Q=zeros(nstates,nactions);
  Q=sparse(Q);

  for l=1:L
    results = zeros(M,1);           % ゲームの結果
    rand('state',1);                % seedの初期化
    newQ=zeros(nstates,nactions);   % 更新用価値関数の初期化

    for m=1:M
      state3 = zeros(1,9);

      for t=1:T
        % 状態, 報酬, ゲーム状況の観測
        state = encode(state3);

        % 政策の生成
        policy = zeros(1,nactions);

        switch(options.pmode)
          case 1 % greedy
            [v,a] = max(Q(state,:));
            policy(a) = 1;

          case 2 % e-greedy
            [v,a] = max(Q(state,:));
            policy = ones(1,nactions)*options.epsilon/nactions;
            policy(a) = 1-options.epsilon+options.epsilon/nactions;

          case 3 % softmax
            policy=exp(Q(state,:)/options.tau)/sum(exp(Q(state,:)./options.tau));
        end

        % 行動の選択および実行
        [action,reward,state3,fin] = action_train(policy,t,state3);

        % 1ステップ前の状態, 行動のQ値を更新
        if t > 1
          newQ(pstate,paction) = newQ(pstate,paction) + options.alpha*(reward - newQ(pstate,paction) ↙
options.gamma*max(newQ(state,:)));
        end

        % ゲーム終了
        if(fin>0)
          results(m) = fin;
          break;
        end

        % 状態と行動の記録
        pstate = state;
```

```
        paction = action;
      end
    end

    Q = newQ;

    % 勝率の計算
    rate(l) = size(find(results==2),1)./M;

    % 標準出力
    fprintf(1,'%d) Win=%d/%d, Draw=%d/%d, Lose=%d/%d¥n',l, size(find(results==2),1),M,size(find↙
(results==3),1),M,size(find(results==1),1),M);
    fflush(stdout);
  end

  %  グラフの出力
  figure(1)
  clf
%  axes('FontSize',15,'LineWidth',2.0);
  games = M:M:M*L;
  g=plot(games,rate);
  set(g,'LineWidth',2);
  g=xlabel('ゲーム数');
  set(g,'FontSize',14);
  g=ylabel('勝率');
  set(g,'FontSize',14);
  axis([M,M*L,0.4,1])
```

2:1:0:2:1:2:0:1:0:
1

```
function Q=TDLambdaPolicyIteration(L,M,options)
  nstates = 3^9;   % 状態数
  nactions = 9;    % 行動数
  T = 5;           % 最大ステップ数

  % Q関数の初期化
  Q=zeros(nstates,nactions);
  Q=sparse(Q);

  for l=1:L
    results = zeros(M,1);           % ゲームの結果
    rand('state',1);                % seedの初期化
    newQ=zeros(nstates,nactions);   % 価値関数の初期化
    e=zeros(nstates,nactions);      % 適格度の初期化

    for m=1:M
      state3 = zeros(1,9);

      for t=1:T
        % 状態, 報酬, ゲーム状況の観測
        state = encode(state3);

        % 政策の生成
        policy = zeros(1,nactions);

        switch(options.pmode)
          case 1 % greedy
            [v,a] = max(Q(state,:));
            policy(a) = 1;

          case 2 % e-greedy
            [v,a] = max(Q(state,:));
            policy = ones(1,nactions)*options.epsilon/nactions;
            policy(a) = 1-options.epsilon+options.epsilon/nactions;

          case 3 % softmax
            policy=exp(Q(state,:)/options.tau)/sum(exp(Q(state,:)./options.tau));
        end

        % 行動の選択および実行
        [action,reward,state3,fin] = action_train(policy,t,state3);

        % 1ステップ前の状態, 行動のQ値を更新
        if t > 1
          % 適格度の更新
          e = e.*options.gamma*options.lambda;
          e(pstate, paction) = e(pstate, paction) + 1;

          newQ = newQ + options.alpha * e * (reward - newQ(pstate,paction) + options.gamma * newQ
(state,action));
        end

        % ゲーム終了
        if(fin>0)
          results(m) = fin;
```

```
            break;
          end

          % 状態と行動の記録
          pstate = state;
          paction = action;
        end
      end

      Q = newQ;

      % 勝率の計算
      rate(l) = size(find(results==2),1)./M;

      % 標準出力
      fprintf(1,'%d) Win=%d/%d, Draw=%d/%d, Lose=%d/%d¥n',l, size(find(results==2),1),M,size(find(results==3),1),M,size(find(results==1),1),M);
      fflush(stdout);
    end

    %  グラフの出力
    figure(1)
    clf
    %axes('FontSize',15,'LineWidth',2.0);
    games = M:M:M*L;
    g=plot(games,rate);
    set(g,'LineWidth',2);
    g=xlabel('ゲーム数');
    set(g,'FontSize',14);
    g=ylabel('勝率');
    set(g,'FontSize',14);
    axis([M,M*L,0.4,1])
```

```
% 学習した政策を利用するためのプログラム
function test(Q,options)
  nactions = 9;  % 行動数
  step = 1;      % ステップカウンター

  while(1)
    % 状態, 報酬, ゲーム状況の観測
    [state, state3, fin]=observe_test;

    % 政策の生成
    policy = zeros(1,nactions);
    switch(options.pmode)
      case 1 % greedy
        [v,a] = max(Q(state,:));
        policy(a) = 1;

      case 2 % e-greedy
        [v,a] = max(Q(state,:));
        policy = ones(1,nactions)*options.epsilon/nactions;
        policy(a) = 1-options.epsilon+options.epsilon/nactions;

      case 3 % softmax
        policy=exp(Q(state,:)/options.tau)/sum(exp(Q(state,:)/options.tau));
      end

      % 行動の選択および実行
      [a] = action_test(policy,step,state3);
      step = step + 1;
  end
```