```
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])
```