بسمه تعالی

دانشگاه تحصیلات تکمیلی علوم پایه زنجان

گروه علوم رایانه و فناوری اطلاعات

درس طراحی سیستم های هوشمند

تمرین چهارم

Maze

محمدعمید عباسی 924468 - محمدسروش فرخزاده 924430 - مسعود فتحی 924429

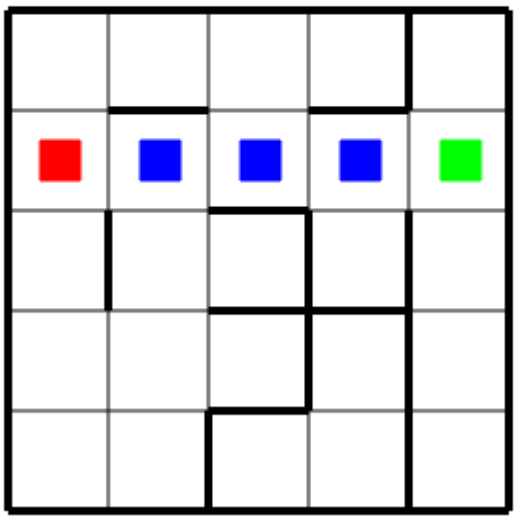
بهار96

فهرست

مقدمه

این پروژه با استفاده از تکنولوژی matlab پیاده سازی شده و شامل شش m file می باشد که برای اجرای الگوریتم value iteration فایل maze.m ، برای اجرای الگوریتم policy iteration فایل maze_policy.m و برای اجرای الگوریتم Q فایل Q2.m اجرا گردد؛ سه m file بعدی توابع استفاده شده در برنامه می باشند که در ادامه توضیح داده می شوند.

هدف از اجرای هر سه الگوریتم value iteration ، policy iteration و Q به دست آوردن تابع سیاست بهینه می باشد که بیانگر بهترین حرکت در هر حالت می باشد.



الگوریتم value iteration

Value iteration یکی از الگوریتم های حوزه ی یادگیری تقویتی در یادگیری ماشین می باشد که در آن هدف به دست آوردن تابع سیاست می باشد. در این الگوریتم توابع reward و sigma برای عامل شناخته شده می باشد.

1. For each state s, initialize $V(s) := 0$.
2. Repeat until convergence {

   For every state, update $V(s) := R(s) + \max_{a \in A} \gamma \sum_{s'} P_{sa}(s')V(s')$.

   }

This algorithm can be thought of as repeatedly trying to update the estimated value function using Bellman Equations (2).

Value iteration algorithm

در این الگوریتم ابتدا value های همه ی خانه ها با صفر مقداردهی اولیه می شوند، سپس تا زمانی که همگرایی رخ دهد، value هر خانه طبق فرمول آپدیت می شود.

منظور از همگرایی در این الگوریتم یکسان بودن ماتریس value در دو iteration متوالی می باشد.

فرمول value به این صورت می باشد که در هر دور برای هر خانه مقدار مجموع reward و gama ضربدر value خانه ی بعد را به ازای همه ی action های ممکن آن خانه حساب می کند و مقدار ماکزیمم را جایگزین value فعلی خانه می کند.

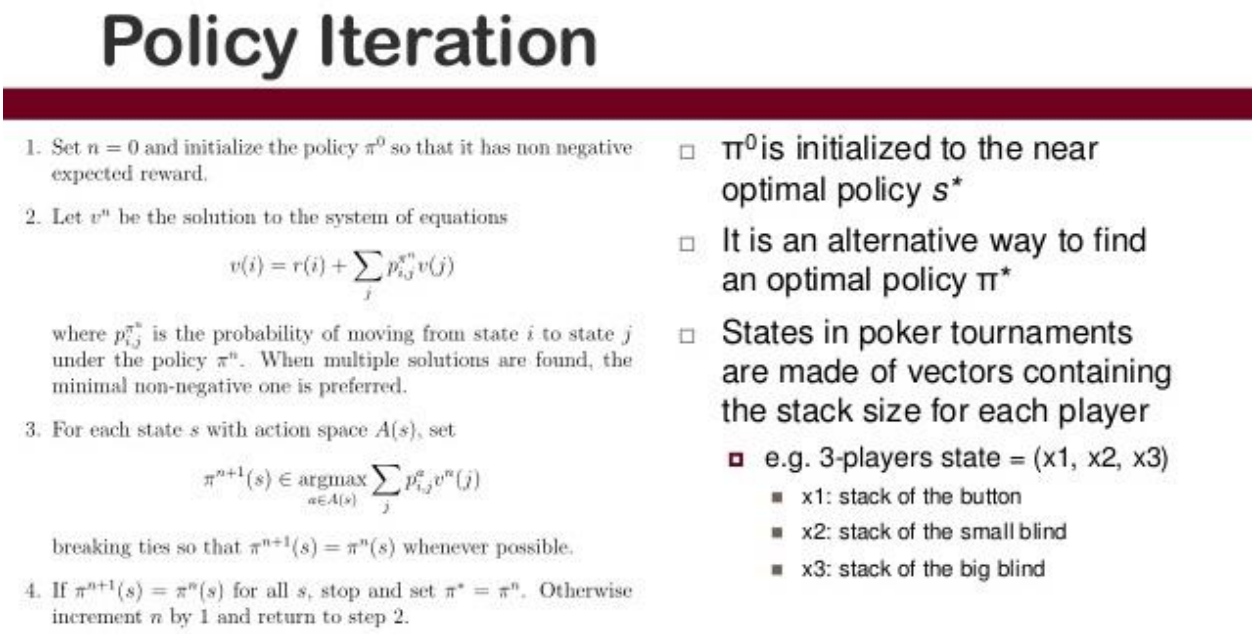
پیاده سازی این الگوریتم به صورت زیر می باشد:

```
while ~(all(G2 == G1))
    G1 = value;
    for i = 1:(s*s)
            if any(goal == i)
                value(i)=0;
            else
                if B(i,1)
                temp_val(1) = reward(i,'N',goal,s) + gama*value(sigma(i,'N',s));
                end
                if B(i,2)
                temp_val(2) = reward(i,'E',goal,s) + gama*value(sigma(i,'E',s));
                end
                if B(i,3)
                temp_val(3) = reward(i,'S',goal,s) + gama*value(sigma(i,'S',s));
                end
                if B(i,4)
                temp_val(4) = reward(i,'W',goal,s) + gama*value(sigma(i,'W',s));
                end
                [value(i),policy(i)] = max(temp_val);
                temp_val = zeros(4,1);
            end
    end
    G2 = value;
    iter = iter + 1;

end
```

الگوریتم policy iteration

policy iteration یکی از الگوریتم های حوزه ی یادگیری تقویتی در یادگیری ماشین می باشد که در آن هدف به دست آوردن تابع سیاست می باشد. در این الگوریتم توابع reward و sigma برای عامل شناخته شده می باشد.



Policy iteration algorithm

در این الگوریتم ابتدا مقدار policy برای همه ی خانه ها به صورت رندوم مقداردهی اولیه می شود و سپس تا زمانی که همگرایی رخ دهد دو عمل زیر تکرار می شود:

عمل یک: به دست آوردن value هر خانه با توجه به policy مرحله ی قبل. که به دو صورت قابل پیاده سازی می باشد:

1-روش ریاضی: در این روش با استفاده از policy مرحله ی قبل معادله ای از valueها به دست می آوریم و معادله را حل می کنیم. برای حل این معادله می توان از linear regression استفاده کرد. که فرمول آن به صورت زیر می باشد:

$X = (A^TA)^{-1} * A^T * B$

که در این فرمول ماتریس X ماتریس valueها و ماتریس A ، ماتریس ضرایب و B ماتریس اعداد طرف دیگر معدله می باشد.

2-روش استفاده از simplified value iteration: در این روش با استفاده از value iteration ساده شده مقدار valueها را با توجه به policyهای مرحله ی قبل به دست می آوریم؛ که ما در این برنامه از این روش استفاده کرده ایم.

تفاوت Value iteration ساده شده با value itartion اصلی در این می باشد که در value iteration اصلی در هر iteration ماکزیمم را انتخاب می کنیم ولی در value iteration ساده شده در هر iteration ، value را با توجه به policy مرحله ی قبل انتخاب می کنیم و نه مقدار ماکزیمم را!!!

عمل دوم: به روز رسانی مقدار policy می باشد. که در این عمل policy یا actionای انتخاب می شود که مقدار زیر را max کند:

Reward(s,a) + gama*v(sigma(s,a))

پیاده سازی:

```
while ~(all(G2 == G1))
    G1 = policy;
    for i = 1:(s*s)
            for j = 1:(s*s)
                if any(goal == j)
                    value(j) = 0;
                else
                    if B(j,policy(j))
                    value(j) = reward(j,policy_ac(policy(j)),goal,s) + gama*value(sigma(j,policy_ac(policy(j)),s));
                    end
                end
            end
            if any(goal == i)
                value(i)=0;
            else
                if B(i,1)
                temp_val(1) = reward(i,'N',goal,s) + gama*value(sigma(i,'N',s));
                end
                if B(i,2)
                temp_val(2) = reward(i,'E',goal,s) + gama*value(sigma(i,'E',s));
                end
                if B(i,3)
                temp_val(3) = reward(i,'S',goal,s) + gama*value(sigma(i,'S',s));
                end
                if B(i,4)
                temp_val(4) = reward(i,'W',goal,s) + gama*value(sigma(i,'W',s));
                end
                [~,policy(i)] = max(temp_val);
                temp_val = zeros(4,1);
            end
    end
    G2 = policy;
   iter = iter + 1;
end
```

الگوریتم یادگیری Q

Q یکی از الگوریتم های حوزه ی یادگیری تقویتی در یادگیری ماشین می باشد که در آن هدف به دست آوردن تابع سیاست می باشد. در این الگوریتم توابع reward و sigma برای عامل شناخته شده نمی باشد و عامل فقط sigma و reward لحظه ای را می داند.

Q learning algorithm

For each s, a initialize the table entry $\hat{Q}(s, a)$ to zero.
Observe the current state s
Do forever:

- Select an action a and execute it
- Receive immediate reward r
- Observe the new state s'
- Update the table entry for $\hat{Q}(s, a)$ as follows:

$$\hat{Q}(s, a) \leftarrow r + \gamma \max_{a'} \hat{Q}(s', a')$$

- $s \leftarrow s'$

Q-learning algorithm

در این الگوریتم یک جدول Q به اندازه تعداد عمل ها ضربدر تعداد حالت ها می سازیم و آن را با صفر مقدار دهی اولیه می کنیم سپس یک حالت رندوم را انتخاب می کنیم و وارد حلقه ی بی نهایت می شویم:

یک عمل را انتخاب می کنیم و reaward ناشی از آن را از محیط دریافت می کنیم و حالت S′ ناشی از آن را مشاهده می کنیم؛سپس جدول Q را مطابق فرمول به روز رسانی می کنیم و وارد state جدید می شویم. زمانی که به حالت goal رسیدیم یک episode تمام می شود. و برای episode جدید دوباره الگوریتم را تکرار می کنیم.

پیاده سازی:

```
for i = 0:maxiter
    %ka = ka + 1;
%     if i>250000
%         ka = 10;
%     end
    if(any(current_state == goal))
        if(h~=1)
            for u = 1:h-1
                Q_table(w(1,u)) = w(2,u);
            end
            h = 1;
        end
        current_state =  randi([1 s*s],1);
    else
        for l = 1 : 4
           aa = aa + ka^Q_table((current_state-1)*4+l);
        end
        for l = 1:4
            kaa(l) = ka^Q_table((current_state-1)*4+l)/aa;
        end
        action =(2*eps)*floor(rand/(2*eps));
        aa = 0;
```

```

        if (0 <= action) && (action < kaa(1))
            action = 1;
        elseif (kaa(1) <= action) && (action < (kaa(1) + kaa(2)))
            action = 2;
        elseif ((kaa(1) + kaa(2)) <=action) && (action < (kaa(1) + kaa(2) + kaa(3)))
            action = 3;
        else
            action = 4;
        end
        %action = randi([1 4],1);
        if(B(current_state,action))
                    w(1,h) = ((current_state - 1)*4) + action;
                    w(2,h) = reward(current_state,policy_ac(action),goal,s) + ...
                    gama*max(Q_table(((sigma(current_state,policy_ac(action),s)-1)*4 + 1):((sigma(current_state,policy_ac(action),s)-1)*4)+ 4));
                    current_state = sigma(current_state,policy_ac(action),s);
                    h = h + 1;
        elseif (g>2000)
                    current_state =  randi([1 s*s],1);
                    g  = 0;
                    h = 1;
        else
                    g = g + 1;
        end
    end
end
```

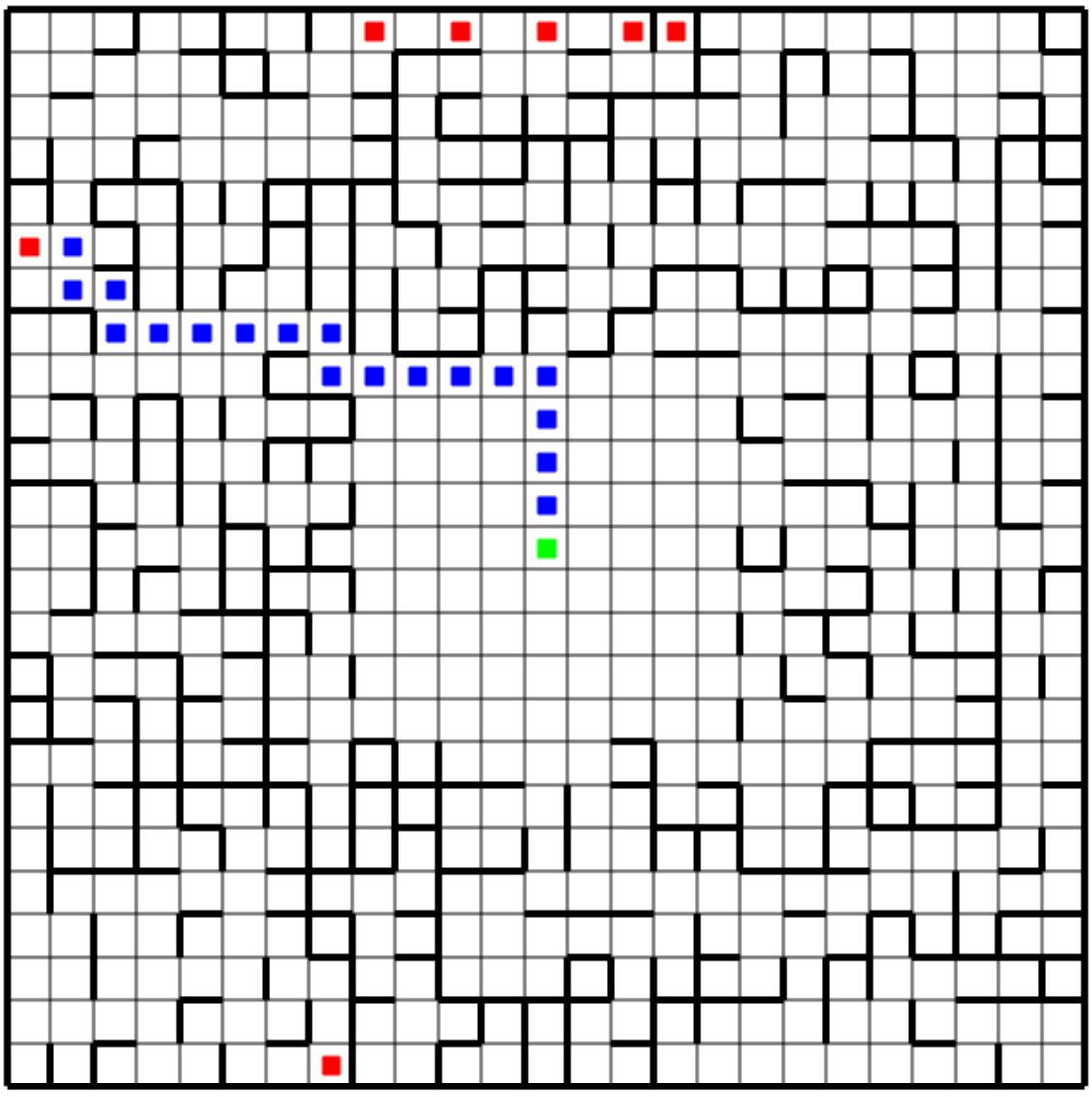
ابتدای هر m file

در ابتدای هر m file، فایل های ورودی برنامه خوانده می شوند و در یک آرایه ذخیره می شوند.

```
fileid = fopen('maze_25.txt','r');
A = fscanf(fileid,'%u');
s = A(1); %size of matrix
if s == 5
    start = A(103);
    goal = A(104:end);
    A = A(3:102);
    B = reshape(A,[4,25]);
    B = B';
elseif s == 25
    start = A(2503);
    goal = A(2504:end);
    A = A(3:2502);
    B = reshape(A,[4,625]);
    B = B';
end
```

انتهای هر m file

در انتهای هر m file، با استفاده از policy به دست آمده maze مربوطه کشیده می شود؛ به این صورت که با توجه به فایل ورودی خط های عمودی و افقی ترسیم می گردد. سپس با توجه به نقطه ی شروع و پایان(ها) و policy مسیر شروع تا پایان کشیده می شود.



مقایسه ی الگوریتم های policy و value

بدون در نظر گرفتن بحث محاسباتی هر دو الگوریتم اثبات همگرایی دارند و هر دو به جواب بهینه همگرا می شوند.

الگوریتم value iteration در هر iteration فقط همسایه های خانه هایی را که مقدار دارند به روز می کند، در صورتی که policy iteration در هر iteration تمام خانه ها را به روز رسانی می کند، که این باعث می شود در بعضی از مسائل در iteration های کمتری به جواب بهینه برسد.

Policy تا حدودی به مقدار دهی اولیه هم وابستگی دارد.

درکل برای مسائل کوچک policy ممکن است بهتر از value عمل کند ولی برای مسائل بزرگ value بهتر است چون policy ممکن است به جواب نرسد.

مشکلات الگوریتم Q

1-بزرگ بودن جدول Q .

2- رندوم انتخاب کردن عملها.

راه حل: استفاده از راه حل احتمالاتی. با استفاده از این روش احتمال انتخاب خانه هایی که مقدار آن ها کمتر می باشد یعنی آپدیت نشده اند را در اوایل اجرا که نیاز به explore داریم و احتمال انتخاب خانه هایی را که مقدار بیشتری دارند یعنی آپدیت شده اند را در اواخر اجرا که نیاز به explit داریم افزایش می دهیم.

K عددی است که در اوایل مسئله مقدارش کم و رفته رفته زیاد می شود.

فرمول استفاده شده :

Exploration versus Exploitation

- The Q-learning algorithm doesn't say how we could choose an action
- If we choose an action that maximises our estimate of Q we could end up not exploring better alternatives
- To converge on the true Q values we must favour higher estimated Q values but still have a chance of choosing worse estimated Q values for exploration (see the convergence proof of the Q-learning algorithm in [Mitchell, sec. 13.3.4.]). An action selection function of the following form may employed, where k>0:

$$P(a_k \mid s) = \frac{k^{\hat{Q}(s,a_k)}}{\sum_{k'} k^{\hat{Q}(s,a_{k'})}}$$

3-در هر iteration فقط یک خانه رو آپدیت می کند.

راه حل: همه ی خانه های یک episode را ذخیره و زمانی که به goal رسید update می کنیم بدین صورت تعداد iterationها را کاهش می دهیم

پاسخ سوالات

الف) بله چون در هر قدم عملی که value را بیشینه می کند، انتخاب می گردد. Value هم برابر است با جمع ریوارد های تخفیف داده شده تا هدف.

ب) مانند قسمت الف.

ج) در قسمت مقایسه ی دو الگوریتم policy و value ذکر گردید.

چ) مانند الف.

د) در قسمت مشکلات الگوریتم Q ذکر گردید.