

```
% Convenience script for running a single test.  
addpaths;  
test_result = runtest('an',struct(),'LinearMDP',...  
    'objectworld',struct('n',32,'determinism',0.7,'seed',1,'continuous',0),...  
    struct('training_sample_lengths',8,'training_samples',16,'verbosity',2));  
  
% Visualize solution.  
% printresult(test_result);  
visualize(test_result);
```

runtest(algorithm, algorithm-params, mdp-model, mdp, mdp-params, test-params)

algorithm = 'an' AN algorithm은 1%

algorithm_params = struct() algorithm-params는 1 struct.

mdp-model = 'LinearMDP' mdp-model은 LinearMDP은 1%

mdp = 'objectworld' mdp는 mdp는 'objectworld'

mdp-params = struct('n', 32, 'determinism', 0.7, 'seed', 1, 'continuous', 0)

mdp는 build된 8개의 16개의 mdp의 parameters.

test-params = struct('training_sample_lengths', 8, 'training_samples', 16, 'verbosity', 2)

test는 9개의 parameters.

```
% Run IRL test with specified algorithm and example.
function test_result = runtest(algorithm,algorithm_params, ...
    mdp_model,mdp,mdp_params,test_params)

% test_result - structure that contains results of the test:
%   see evaluateirl.m
% algorithm - string specifying the IRL algorithm to use; one of:
%   firl - NIPS 2010 FIRL algorithm
%   bfirl - Bayesian FIRL algorithm
% algorithm_params - parameters of the specified algorithm:
%   FIRL:
%       seed (0) - initialization for random seed
%       iterations (10) - number of FIRL iterations to take
%       depth_step (1) - increase in depth per iteration
%       init_depth (0) - initial depth
%   BFIRL:
%       seed (0) - initialization for random seed
% mdp_model - string specifying MDP model to use for examples:
% standardmdp - standard MDP model
% mdp - string specifying example to test on:
% gridworld
% mdp_params - string specifying parameters for example:
% Gridworld:
%       seed (0) - initialization for random seed
%       n (32) - number of cells along each axis
%       b (4) - size of macro cells
%       determinism (1.0) - probability of correct transition
%       discount (0.9) - temporal discount factor to use
% test_params - general parameters for the test:
%   test_models - models to test on
%   test_metrics - metrics to use during testing
%   training_samples (32) - number of example trajectories to query
%   training_sample_lengths (100) - length of each sample trajectory
%   true_features ([]) - alternative set of true features

% Make sure relevant paths are added.
addpath;

% Set default test parameters.
test_params = setdefaulttestparams(test_params);

```

mdpbuild을 이용해 mdp를 만드는 과정

```
% Construct MDP and features.
[mdp_data,r,feature_data,true_feature_map] = feval(strcat(mdp,'build'),mdp_params)
if isempty(test_params.true_features),
    true_feature_map = test_params.true_features;
end;
% Solve example.
true_feature_map을 사용
mdp_solution = feval(strcat(mdp_model,'solve'),mdp_data,r);
```

- test 향상하는 algorithm 이다
ex) ani_gpirl ...
- 각 algorithm에 필요한 parameter
struct 형태로 있음.
struct()로 다양한 경우 각 algorithm에
default parameter 사용.
ex) AN (이때 andefaultparams.m 참고)
Struct는 'seed', 'all_features', 'true-
features'로 구성.
default values 각각 0, 1.0
- mdp_model
향상하는 mdp의 형태.
standardmdp 또는 linearmdp 두 가지를
선택하는 듯 함.
- mdp
향상하는 mdp 알고리즘
ex) gridworld, Highway
- mdp_params
향상하는 mdp 예제를 정의하기 위한
parameter들.
- test_params
IRL 학습을 위해 주어지는 값
- feval (fun, variable) . fun 이름을 가진 함수를 variable을
입력으로 실행시킴.
String concatenate
feval(strcat(mdp,'build'), mdp_params) → mdpbuild 함수를 mdp_params
에 대입해 실행.
ex) mdp=gridworld 이런
strcat(mdp,'build') = gridworldbuild.

그리고 mdp_data와 r을 이용해 mdp를 만든다.

```
% Sample example trajectories.
if isempty(test_params.true_examples), test_params.true_examples 가 빈집
    example_samples = sampleexamples(mdp_model, mdp_data, mdp_solution, test_params);
else
    sampleexamples를 이용해 true_examples에 따른 example_sample 만들기.
    example_samples = test_params.true_examples;
end; else ok example-samples를 사용.

% Run IRL algorithm.
irl_result = feval(strcat(algorithm,'run'), algorithm_params, mdp_data, mdp_model, ...
    feature_data, example_samples, true_feature_map, test_params.verbosity);

% Evaluate result.
test_result = evaluateirl(irl_result, example_samples, mdp_data, mdp_params, ...
    mdp_solution, mdp, mdp_model, test_params.test_models, ...
    test_params.test_metrics, feature_data, true_feature_map); evaluateirl 결과를 irl_result에 정가.
test_result.algorithm = algorithm;
```

setDefaultTestParams.m

17. 4. 18 오후 7:22 C:\Users\JEON\Desktop\Re...setDefaultTestParams.m 1 / 1

```
% Set default general parameters for the test.  
function test_params = setDefaultTestParams(test_params)  
  
% Create default parameters.  
default_params = struct(...  
    'verbosity',2,...  
    'training_samples',32,...  
    'training_sample_lengths',100,...  
    'true_features',[],...  
    'true_examples',[],...  
    'test_models',{{'standardmdp','linearmdp'}},...  
    'test_metrics',{{'misprediction','policydist','featexp','value',...  
        'featvar','reward','rewarddemean','rewardmomentmatch'}});  
  
% Set parameters.  
test_params = fillDefaultParams(test_params,default_params);
```

test_params은 default_params의 field를 test_params에 있는 field와
default_params의 해당 field와 field 값을 차별화
즉, test_params에 있는 field는 default에 있는 field와 default 값은 차별화
되는 field가 있는 경우 structure로 정의됨.

file defaultparams.m

17. 4. 18 오후 9:12 C:\Users\JEON\Desktop\Rese...\\filldefaultparams.m 1 / 1

```
% Fill in default parameters of a structure.  
function params = filldefaultparams(params,defaults)  
  
% Get default field names. ↗ structure의 field name을 저장한다.  
defaultfields = fieldnames(defaults); ↗ default parameter structure의 field name을 받아온다.  
  
% Step over all fields in the defaults structure.  
for i=1:length(defaultfields),  
    if ~isfield(params,defaultfields{i}),  
        params.(defaultfields{i}) = defaults.(defaultfields{i});  
    end;  
end; ↗ params에 defaultfield(i)가 있는지 없는지 체크.  
        ↗ params에 defaultfield(i) field를 만들고  
        ↗ defaults.(default fields(i)) 값을 저장.  
end;
```

```
% Construct the Objectworld MDP structures.
```

```
function [mdp_data,r,feature_data,true_feature_map] = objectworldbuild(mdp_params)
```

% mdp_params - parameters of the objectworld:

% seed (0) - initialization for random seed

% n (32) - number of cells along each axis

% placement_prob (0.05) - probability of placing object in each cell

% c1 (2) - number of primary "colors"

% c2 (2) - number of secondary "colors"

% determinism (1.0) - probability of correct transition

% discount (0.9) - temporal discount factor to use

% mdp_data - standard MDP definition structure with object-world details:

% states - total number of states in the MDP

% actions - total number of actions in the MDP

% discount - temporal discount factor to use

% sa_s - mapping from state-action pairs to states

% sa_p - mapping from state-action pairs to transition probabilities

% map1 - mapping from states to c1 colors

% map2 - mapping from states to c2 colors

% c1array - array of locations by c1 colors

% c2array - array of locations by c2 colors

% r - mapping from state-action pairs to rewards

% Fill in default parameters.

mdp_params = objectworlddefaultparams(mdp_params); *objectworlddefaultparams* 함수를 이용해 장애물 초기 mdp_params
의 default 값을 설정.

% Set random seed.

rand('seed', mdp_params.seed);

% Build action mapping.

action
state
probability
n
action

```
sa_s = zeros(mdp_params.n^2,5,5);
sa_p = zeros(mdp_params.n^2,5,5);
for y=1:mdp_params.n,
    for x=1:mdp_params.n,
        s = (y-1)*mdp_params.n+x;
        successors = zeros(1,1,5);
        successors(1,1,1) = s;
        successors(1,1,2) = (min(mdp_params.n,y+1)-1)*mdp_params.n+x;
        successors(1,1,3) = (y-1)*mdp_params.n+min(mdp_params.n,x+1);
        successors(1,1,4) = (max(1,y-1)-1)*mdp_params.n+x;
        successors(1,1,5) = (y-1)*mdp_params.n+max(1,x-1);
        sa_s(s,:,:)= repmat(successors,[1,5,1]);
        sa_p(s,:,:)= reshape(eye(5,5)*mdp_params.determinism +...
            (ones(5,5)-eye(5,5))*((1.0-mdp_params.determinism)/4.0),...
            1,5,5);
    end;
end;
```

% Construct map.

```
map1 = zeros(mdp_params.n^2,1);
```

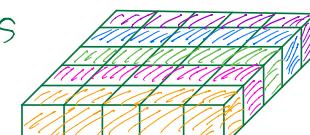
n²

0.6	0.1	0.1	0.1	0.1
0.1	0.6	0.1	0.1	0.1
0.1	0.1	0.6	0.1	0.1
0.1	0.1	0.1	0.6	0.1
0.1	0.1	0.1	0.1	0.6



0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.6	0.1	0.1
0.1	0.6	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.6

sa_s



sa_p



0.6	0.1	0.1	0.1	0.1
0.1	0.1	0.6	0.1	0.1
0.1	0.6	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.6

0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.6	0.1	0.1
0.1	0.6	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.6

0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.6	0.1	0.1
0.1	0.6	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.6

0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.6	0.1	0.1
0.1	0.6	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.1
0.1	0.1	0.1	0.1	0.6

```

map2 = zeros(mdp_params.n^2,1);
c1array = cell(mdp_params.c1,1); → C1이 92개인 경우 초기화된 cell 배열
c2array = cell(mdp_params.c2,1);

% Place objects in "rounds", with 2 colors each round.
% This ensures, for example, that increasing c1 from 2 to 4 results in all
% of the objects from c1=2 being placed, plus additional "distractor"
% objects. This prevents the situation when c1 is high of not placing any
% objects with c1=1 or c1=2 (which makes the example useless for trying to
% infer any meaningful reward).

for round=1:ceil(mdp_params.c1*0.5),
    initc1 = (round-1)*2;
    if initc1+1 == mdp_params.c1,
        % Always choose the leftover c1.
        prob = mdp_params.placement_prob*0.5;
        maxc1 = 1;
    else
        % Choose from two c1 colors.
        prob = mdp_params.placement_prob;
        maxc1 = 2;
    end;
    for s=1:mdp_params.n^2,
        if rand(1,1) < prob && map1(s) == 0,
            % Place object.
            c1 = initc1+ceil(rand(1,1)*maxc1);
            c2 = ceil(rand(1,1)*mdp_params.c2);
            map1(s) = c1;
            map2(s) = c2;
            c1array{c1} = [c1array{c1};s];
            c2array{c2} = [c2array{c2};s];
        end;
    end;
end;

% Create MDP data structure.
mdp_data = struct(...,
    'states', mdp_params.n^2, ...
    'actions', 5, ...
    'discount', mdp_params.discount, ...
    'determinism', mdp_params.determinism, ...
    'sa_s', sa_s, ...
    'sa_p', sa_p, ...
    'map1', map1, ...
    'map2', map2, ...
    'c1array', {c1array}, ...
    'c2array', {c2array});

```

주어진 mdp_params와 이를 통해 생성한 mdp structure의 mdp_data
를 바탕으로 feature map 생성.

Objectworld default params.m

17. 4. 19 오전 10:47 C:\Users\JEON\Desktop\objectworld\defaultparams.m 1 / 1

```
% Fill in default parameters for the objectworld example.
function mdp_params = objectworlddefaultparams(mdp_params)

% Create default parameters.
default_params = struct(... → objectworld를 build하기 위한 기본적인 Mdp_params의 default 값.
    'seed', 0, ...
    'n', 32, ...
    'placement_prob', 0.05, ...
    'c1', 2, ...
    'c2', 2, ...
    'continuous', 0, ...
    'determinism', 1.0, ...
    'discount', 0.9); → mdp_params의 각각의 field에 대한 기본값을 지정하는 것.

% Set parameters. →
mdp_params = filldefaultparams(mdp_params, default_params);

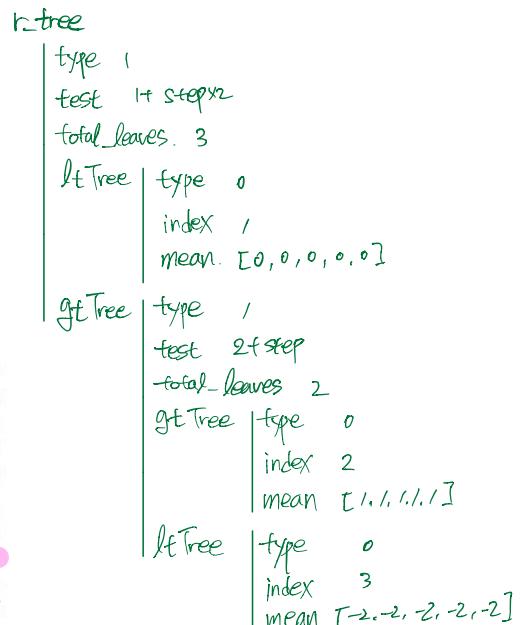
% Construct default reward tree.
c1 = mdp_params.c1; number of primary color
c2 = mdp_params.c2; number of secondary color.
step = c1+c2;
%{
r_tree = struct('type', 1, 'test', 1+step*2, 'total_leaves', 3, ... % Test distance to c1 1 shape
    'gtTree', struct('type', 0, 'index', 1, 'mean', [-2, -2, -2, -2, -2]), ... % Penalty for being close to c1 ↵
shape
    'ltTree', struct('type', 1, 'test', 2+step*1, 'total_leaves', 2, ... % Test distance to c1 2 shape
        'gtTree', struct('type', 0, 'index', 2, 'mean', [1 1 1 1 1]), ... % Reward for being close
        'ltTree', struct('type', 0, 'index', 3, 'mean', [0 0 0 0 0])); % Neutral reward for any other ↵
state.
%}
r_tree = struct('type', 1, 'test', 1+step*2, 'total_leaves', 3, ... % Test distance to c1 1 shape
    'ltTree', struct('type', 0, 'index', 1, 'mean', [0, 0, 0, 0, 0]), ... % Neutral reward for being elsewhere
    'gtTree', struct('type', 1, 'test', 2+step*1, 'total_leaves', 2, ... % Test distance to c1 2 shape
        'gtTree', struct('type', 0, 'index', 2, 'mean', [1 1 1 1 1]), ... % Reward for being close
        'ltTree', struct('type', 0, 'index', 3, 'mean', [-2 -2 -2 -2 -2])); % Penalty otherwise

% Create default parameters.
default_params = struct(... →
    'r_tree', r_tree);

% Set parameters.
mdp_params = filldefaultparams(mdp_params, default_params);
reward tree는 초기화 되었을 때 mdp_params로 설정됨.
```

7.1 Objectworld Experiments

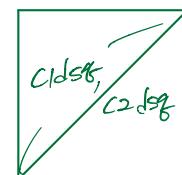
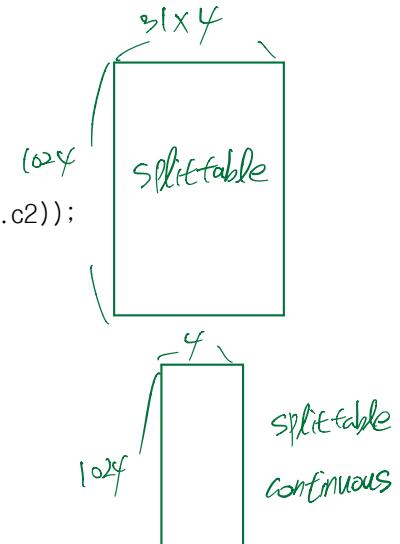
The objectworld is an $N \times N$ grid of states with five actions per state, corresponding to steps in each direction and staying in place. Each action has a 30% chance of moving in a different random direction. Randomly placed objects populate the objectworld, and each is assigned one of C inner and outer colors. Object placement is randomized in the transfer environments, while N and C remain the same. There are $2C$ continuous features, each giving the Euclidean distance to the nearest object with a specific inner or outer color. In the discrete feature case, there are $2CN$ binary features, each one an indicator for a corresponding continuous feature being less than $d \in \{1, \dots, N\}$. The true reward is positive in states that are both within 3 cells of outer color 1 and 2 cells of outer color 2, negative within 3 cells of outer color 1, and zero otherwise. Inner colors and all other outer colors are distractors. The algorithms were provided example paths of length 8, and the number of examples and colors was varied to determine their ability to handle limited data and distractors.



Objectworld features.m

17. 4. 19 오후 11:53 C:\Users\JEON\Desktop\Re...\Objectworldfeatures.m 1 / 2

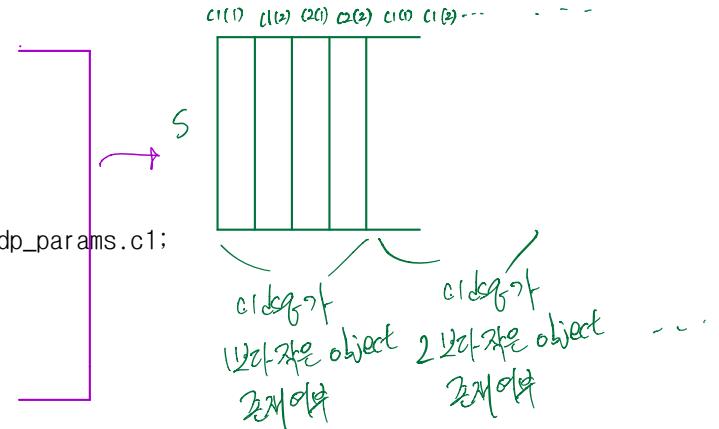
```
1 % Construct the raw features for the objectworld domain.
2 function [feature_data,true_feature_map,r] = objectworldfeatures(mdp_params,mdp_data)
3
4 % mdp_params - definition of MDP domain
5 % mdp_data - generic definition of domain
6 % feature_data - generic feature data:
7 %   splittable - matrix of states to features
8 %   stateadjacency - sparse state adjacency matrix
9
10 % Fill in default parameters.
11 mdp_params = objectworlddefaultparams(mdp_params);
12
13 % Construct adjacency table.
14 stateadjacency = sparse([],[],[],mdp_data.states,mdp_data.states,..
15     mdp_data.states*mdp_data.actions);
16 for s=1:mdp_data.states,
17     for a=1:mdp_data.actions,
18         stateadjacency(s,mdp_data.sa_s(s,a,1)) = 1;
19     end;
20 end;
21
22 % Construct discrete and continuous split tables.
23 splittable = zeros(mdp_data.states,(mdp_params.n-1)*(mdp_params.c1+mdp_params.c2));
24 splittablecont = zeros(mdp_data.states,mdp_params.c1+mdp_params.c2);
25 for s=1:mdp_data.states,
26     % Get x and y positions.
27     y = ceil(s/mdp_params.n); → 
28     x = s-(y-1)*mdp_params.n; → 
29
30     % Determine distances to each type of object.
31     c1dsq = sqrt(2*(mdp_params.n^2))*ones(mdp_params.c1,1);
32     c2dsq = sqrt(2*(mdp_params.n^2))*ones(mdp_params.c2,1);
33     for i=1:mdp_params.c1, → c1을 이용해 사방을 갖는 object의 벡터.
34         for j=1:length(mdp_data.c1array{i}), → i를 통해 각각의 object를 갖는 object의 벡터.
35             cy = ceil(mdp_data.c1array{i}(j)/mdp_params.n);
36             cx = mdp_data.c1array{i}(j)-(cy-1)*mdp_params.n;
37             d = sqrt((cx-x)^2 + (cy-y)^2); → State s-(i,j) object 사이의 거리
38             c1dsq(i) = min(c1dsq(i),d);
39         end; → State s-(i,j) object 사이의 거리를 c1dsq(i)로 정한다.
40     end; → State s-(i,j) object 사이의 거리를 c1dsq(i)로 정한다.
41     for i=1:mdp_params.c2, → 기준으로 object 사이의 거리를 c1dsq(i)로 정한다.
42         for j=1:length(mdp_data.c2array{i}), → 기준으로 object 사이의 거리를 c1dsq(i)로 정한다.
43             cy = ceil(mdp_data.c2array{i}(j)/mdp_params.n);
44             cx = mdp_data.c2array{i}(j)-(cy-1)*mdp_params.n;
45             d = sqrt((cx-x)^2 + (cy-y)^2);
46             c2dsq(i) = min(c2dsq(i),d);
47         end; → 기준으로 object 사이의 거리를 c1dsq(i)로 정한다.
48     end; → 기준으로 object 사이의 거리를 c1dsq(i)로 정한다.
49
50     % Build corresponding feature table (discrete).
```



```

51 for d=1:mdp_params.n-1,
52     strt = (d-1)*(mdp_params.c1+mdp_params.c2);
53     for i=1:mdp_params.c1,
54         splittable(s,strt+i) = c1dsq(i) < d;
55     end;
56     strt = (d-1)*(mdp_params.c1+mdp_params.c2)+mdp_params.c1;
57     for i=1:mdp_params.c2,
58         splittable(s,strt+i) = c2dsq(i) < d;
59     end;
60 end;
61
62 % Build corresponding feature table (continuous).
63 splittablecont(s,1:mdp_params.c1) = c1dsq;
64 splittablecont(s,mdp_params.c1+1:mdp_params.c1+mdp_params.c2) = c2dsq;
65 end;
66
67 % Return feature data structure.
68 feature_data = struct('stateadjacency',stateadjacency,'splittable',splittable);
69
70 % Construct true feature map.
71 true_feature_map = sparse([],[],[],mdp_data.states, ...
72     mdp_params.r_tree.total_leaves,mdp_data.states);
73 for s=1:mdp_data.states,
74     % Determine which leaf state belongs to.
75     [leaf,~] = cartcheckleaf(mdp_params.r_tree,s,feature_data);
76     true_feature_map(s,leaf) = 1;
77 end;
78
79 % Fill in the reward function.
80 R_SCALE = 5;
81 r = cartaverage(mdp_params.r_tree,feature_data)*R_SCALE;
82
83 % Optionally, replace splittable.
84 if mdp_params.continuous,
85     feature_data.splittable = splittablecont;
86 end;
87

```



```

1 % Check which leaf of tree contains leaf.
2 function [leaf, val] = cartcheckleaf(tree, s, feature_data)
3
4 % Check if this is a leaf.
5 if tree.type == 0,
6     % Return result.
7     leaf = tree.index;
8     val = tree.mean;
9 else
10    % Recurse.
11    if feature_data.splittable(s, tree.test) == 0,
12        branch = tree.ltTree;
13    else
14        branch = tree.gtTree;
15    end;
16    [leaf, val] = cartcheckleaf(branch, s, feature_data);
17 end;
18
19 r_tree.type = 1

```

r-tree

type 1
test 1+stepx2 Step = C1+C2
total_leaves 3

ltTree	type 0 index 1 mean [0, 0, 0, 0, 0]
--------	---

leaf
!types

gtTree	type 1 test 2+step total_leaves 2
--------	---

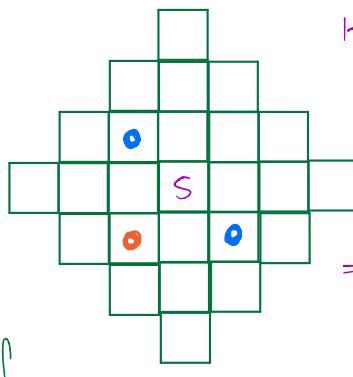
gtTree	type 0 index 2 mean [1, 1, 1, 1, 1]
--------	---

leaf
!types

ltTree	type 0 index 3 mean [-2, -2, -2, -2, -2]
--------	--

else

feature_data.splittable(s, r_tree.test) \Rightarrow 3개의 action이 2개인 A와 B로 나뉨
If stepx2 = 9 \Rightarrow 11번째 행렬에 C(1)이 object가 있다면 A를 선택



\Rightarrow feature_data.splittable(s, r_tree.test) = 1

i- feature_data.splittable(s, r_tree.test) == 0

branch = r_tree.ltTree.
type 0
index 1
mean [0, 0, 0, 0, 0]

Cartcheckleaf(r_tree.ltTree, s, feature_data)

\Rightarrow leaf = r_tree.ltTree.index = 1

val = r_tree.ltTree.mean = [0, 0, 0, 0, 0]

else

feature_data.splittable(s, r_tree.test) = 1

branch = r_tree.gtTree
type 1
test 2+step
total_leaves 2

cartcheckleaf (r-tree.getTree, s, feature_data)

r-tree.getTree.type = 1

else.

if feature_data.splittable (s, r-tree.getTree.test) == 0
2 step

↳ 2 step action은 도달할 수 있는 경로에 바깥쪽 생략이

C(2)인 object 존재 여부

branch = r-tree.getTree.ltTree

type 0

index 3

mean [-2, -2, -2, -2, -2]

cartcheckleaf (r-tree.getTree.ltTree, s, feature_data)

⇒ leaf = r-tree.getTree.ltTree.index = 3

val = r-tree.getTree.ltTree.mean = [-2, -2, -2, -2, -2]

else

feature_data.splittable (s, r-tree.getTree.test) = 1

branch = r-tree.getTree.gtTree

type 0

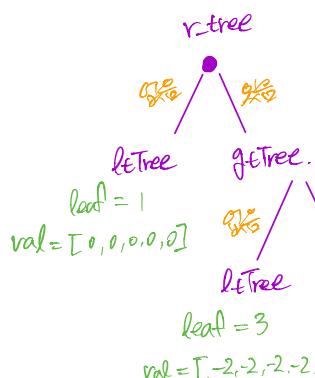
index 2

mean [1, 1, 1, 1, 1]

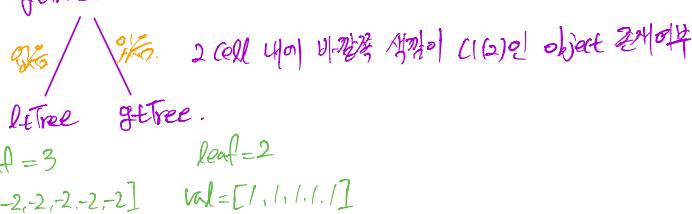
cartcheckleaf (r-tree.getTree.gtTree, s, feature_data)

⇒ leaf = r-tree.getTree.gtTree.index = 2

val = r-tree.getTree.gtTree.mean = [1, 1, 1, 1, 1]



3 cell 내에 바깥쪽 생략이 C(1)인 object 존재 여부



2 cell 내에 바깥쪽 생략이 C(2)인 object 존재 여부

Cartaverage.m

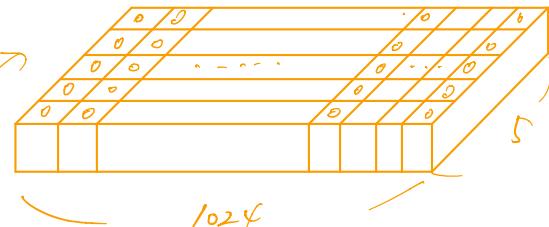
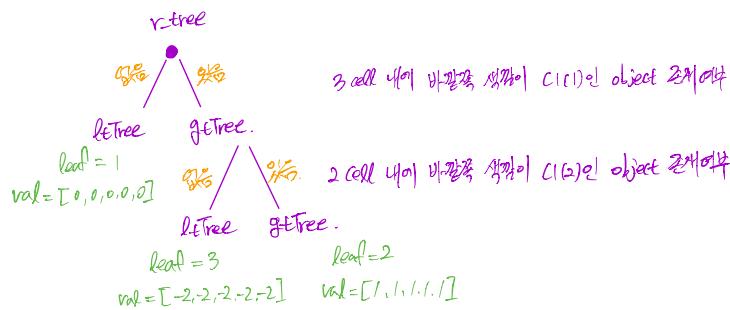
17. 4. 20 오전 4:08 C:\Users\WEON\Desktop\Research\W\cartaverage.m

1 / 1

```

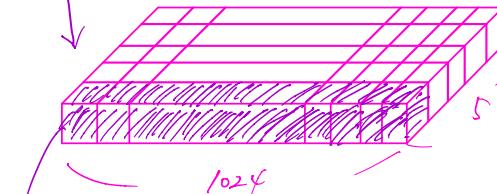
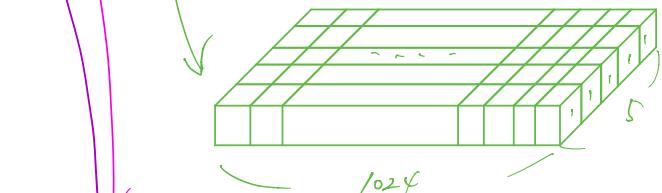
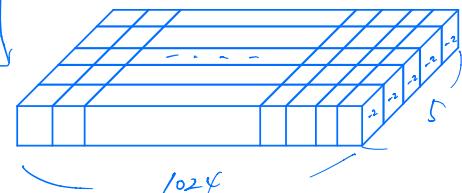
1 % Return average reward for given regression tree.
2 function R = cartaverage(tree, feature_data)
3
4 if tree.type == 0,
5     % Simply return the average.
6     R = repmat(tree.mean, size(feature_data.splittable, 1), 1);
7 else
8     % Compute reward on each side.
9     ltr = cartaverage(tree.ltTree, feature_data);
10    gtr = cartaverage(tree.gtTree, feature_data);
11
12     % Combine.
13     ind = repmat(feature_data.splittable(:, tree.test), 1, size(ltr, 2));
14     R = (1-ind).*ltr + ind.*gtr;      1024 x 1
15 end;  2 cell 나의 값이 2 cell 나의 값이
16
17 ltr
18 gtr.

```



$ltr = \text{Cartaverage}(r_tree.ltTree, feature_data)$

$gtr = \text{Cartaverage}(r_tree.gtTree, feature_data)$



각 상태에서 2 cell 이내에 바깥쪽 설정 C(2)인 object가 존재하는지 여부.

linearmdpsolve.m

17. 4. 20 오후 10:53 C:\Users\JEON\Desktop\Research...\linearmdpsolve.m 1 / 1

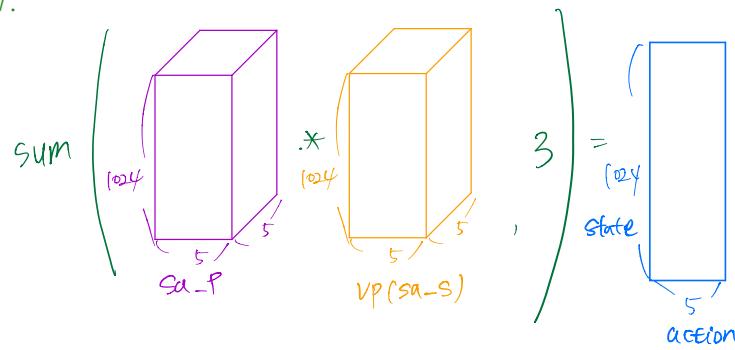
```
1 % Solve a linear MDP and return value function, Q function, and policy.
2 function mdp_solution = linearmdpsolve(mdp_data,r)
3
4 % Run value iteration to compute the value function and policy.
5 [v,q,p] = linearvalueiteration(mdp_data,r);
6
7 % Return solution.
8 mdp_solution = struct('v',v,'q',q,'p',p);
9
    value   Q   policy
    function function
```

```

1 % Run value iteration to solve a linear MDP.
2 function [v,q,p,logp] = linearvalueiteration(mdp_data,r,vinit)
3
4 states = mdp_data.states; % 1024
5 actions = mdp_data.actions; % 5
6 VITR_THRESH = 1e-4;
7 %VITR_THRESH = 1e-10;
8
9 % Compute log state partition function V.
10 if (nargin == 3)
11     v = vinit;
12 else
13     v = zeros(states,1);
14 end;
15 diff = 1.0;
16 while diff >= VITR_THRESH,
17     % Initialize new v.
18     vp = v;
19
20     % Compute q function.
21     q = r + mdp_data.discount * sum(mdp_data.sa_p.*vp(mdp_data.sa_s),3); %  $= R_{t+1} + \gamma \sum_{s'} (P(s'|s,a) \cdot V(s'))$ 
22
23     % Compute softmax.
24     y = maxentsoftmax(q);
25     update V
26     diff = max(abs(v-vp));
27 end;
28
29 % Compute policy.
30 logp = q - repmat(v,1,actions);
31 p = exp(logp); %  $\pi(a|s)$ 
32

```

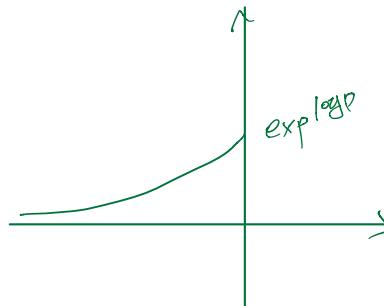
q의 값이가 가장 작은
action을 최종의 policy로
 \Rightarrow 실제로는 확률분포로 나타낸다.



$$= R_{t+1} + \gamma \sum_{s'} (P(s'|s,a) \cdot V(s'))$$

s_t 에서 a_t 를 취했을 때
수행되는 다음 상태 s'
으로 이동할 확률.

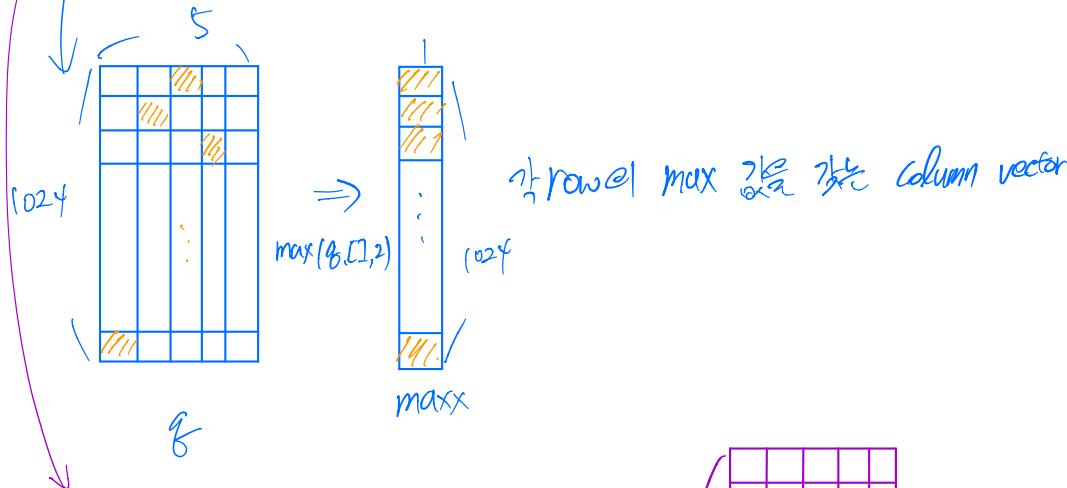
s_t 에서 a_t 를 취했을 때 s'
로 이동할 확률은 $P(s'|s,a)$
이다. $V(s')$



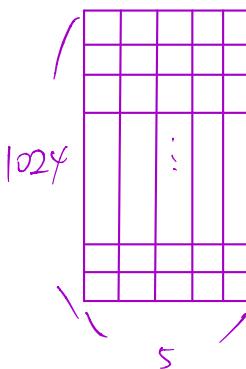
```

1 % Compute column-wise softmax of a matrix.
2 function v = maxentsoftmax(q)
3
4 % Find maximum elements.
5 maxx = max(q,[],2); →  $\max_a E(R_{t+1} + \gamma \sum_s (p(s, t, a) \cdot V(s)) )$ 
6
7 % Compute safe softmax.
8 v = maxx + log(sum(exp(q - repmat(maxx, 1, size(q, 2))), 2));
9

```



$$\text{repmat}(\maxx, 1, \text{size}(q, 2)) =$$



```
% Sample example tranjectories from the state space of a given MDP.
function example_samples = sampleexamples(mdp_model, mdp_data, mdp_solution, test_params)

% Allocate training samples.
N = test_params.training_samples; → sample 수
T = test_params.training_sample_lengths; → sample의 길이
example_samples = cell(N,T); → 각각의 [state, action]의 저장

% Sample trajectories.
for i=1:N,
    % Sample initial state.
    s = ceil(rand(1,1)*mdp_data.states);

    % Run sample trajectory.
    for t=1:T,
        % Compute optimal action for current state.
        a = feval(strcat(mdp_model,'action'), mdp_data, mdp_solution, s); mdp_solution이 state s에 대한
        % Store example. optimal action 선택.
        example_samples{i,t} = [s;a]; → state와 action sample.
        % Move on to next state.
        s = feval(strcat(mdp_model,'step'), mdp_data, mdp_solution, s, a);
    end;
end;
```

linearmdpaction.m

17. 4. 23 오후 11:54 C:\Users\JEON\Desktop\Research\linearmdpaction.m 1 / 1

```
1 % Return optimal action given the mdp solution.  
2 function a = linearmdpaction(mdp_data, mdp_solution, s)  
3  
4 samp = rand(1,1);  
5 total = 0;  
6 for a=1:mdp_data.actions,  
7     total = total+mdp_solution.p(s,a);  
8     if total >= samp,  
9         return;  
10    end;  
11 end;  
12
```

rand(1,1)은最优动作 선택하는가?

mdp_solution.p

0.1	0.15	0.6	0.1	0.05
-----	------	-----	-----	------

ex) samp = 0.8
→ a=3

linearmdpstep.m

17. 4. 23 오후 11:54 C:\Users\JEON\Desktop\Research...\linearmdpstep.m 1 / 1

```
1 % Take a single step with the specified action.  
2 function s = linearmdpstep(mdp_data,~,s,a)  
3  
4 % Random sample for stochastic step.  
5 r = rand(1,1);  
6 sm = 0;  
7 for k=1:size(mdp_data.sa_p,3),  
8     sm = sm+mdp_data.sa_p(s,a,k);  
9     if sm >= r,  
10         s = mdp_data.sa_s(s,a,k);  
11         return;  
12     end;  
13 end;  
14  
15 % Should never reach here.  
16 fprintf(1,'ERROR: MDP data specifies transition distribution for state %i action %i that does not  
sum to 1!\n',...  
17     s,a);  
18 s = -1;  
19
```

State s 에 대한 action a 의 확률 분포

0.1
0.1
0.6
0.1
0.1

s'
s'_1
s'_2
s'_3
s'_4

mdp_data.sa_p

mdp_data.sa_s

```
1 % Abbeel & Ng algorithm implementation (projection version).
2 function irl_result = anrun(algorithm_params,mdp_data,mdp_model, ...
3     feature_data,example_samples,true_features,verbosity)
4
5 % algorithm_params - parameters of the MMP algorithm:
6 %     seed (0) - initialization for random seed
7 %     all_features (0) - use all features as a basis
8 %     true_features (0) - use true features as a basis
9 % mdp_data - definition of the MDP to be solved
10 % example_samples - cell array containing examples
11 % irl_result - result of IRL algorithm, generic and algorithm-specific:
12 %     r - inferred reward function
13 %     v - inferred value function.
14 %     q - corresponding q function.
15 %     p - corresponding policy.
16 %     time - total running time
17
18 % Fill in default parameters.
19 algorithm_params = andefaultparams(algorithm_params);
20 seed=0, all-features=1, true-features=0 = struct()
21 % Set random seed.
22 rand('seed',algorithm_params.seed);
23 tic;
24
25 % Initialize variables.
26 [states,actions,transitions] = size(mdp_data.sa_p); = [1024 5 5]
27 [N,T] = size(example_samples); = [16 8] 1024x5x5.
28 mdp_solve = str2func(strcat(mdp_model,'solve'));
29 linearMpsolve.
30 % Build feature membership matrix.
31 if algorithm_params.all_features,
32     F = feature_data.splitable;
33     % Note that we add a row of 1s to the feature matrix to ensure that we
34     % can control the reward at every state.
35     F = horzcat(F,ones(states,1)); 1024x125
36 elseif algorithm_params.true_features,
37     F = true_features;
38 else
39     F = eye(states);
40 end;
41
42 % Count features.
43 features = size(F,2); = 125
44 F = F';
45
46 % Construct state expectations.
47 muE = zeros(states,1); 1024x1
48 ex_s = zeros(N,T); 16x8
49 ex_a = zeros(N,T); 16x8
50 for i=1:N,
```

```

51 for t=1:T,
52     ex_s(i,t) = example_samples{i,t}(1);
53     ex_a(i,t) = example_samples{i,t}(2);
54     muE(ex_s(i,t)) = muE(ex_s(i,t)) + mdp_data.discount^(t-1);
55 end;
56 end;
57 muE = muE/N;
58 Fmu = F*muE;  $\Rightarrow \hat{\mu}_E = \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^{T-1} \gamma^t \phi(S_t^{(i)})$ 
59
60 % Generate random policy.
61 w = rand(features,1);  $\rightarrow 125 \times 1$ 
62 r = repmat(F'*w, 1, actions);  $R = W^T \phi + \text{각 state } y_t$ 
63 soln = standardmdpsove(mdp_data,r);  $\text{reward.}$ 
64 weights = {w};  $1024 \times 5$ 
65 solutions = {soln};
66 mus = {};
67 mu_bars = {};
68 itr = 1;
69
70 % Initialize t.
71 t = 100.0;
72 told = 0.0;
73
74 while 1,
75     told = t;
76     % Compute feature expectations under the last policy.
77     om = standardmdpfrequency(mdp_data,solutions{itr});
78     mu = F*om;  $\rightarrow$  state expectation ?
79
80     mus{itr} = mu;
81     if itr == 1,
82         mu_bars{itr} = mu;  $\rightarrow \bar{\mu}^{(0)} = \mu^{(0)}$ 
83     end;
84
85     % Increment iteration count.
86     itr = itr + 1;
87
88     % Compute t and w using projection.
89     if itr == 2,
90         % use existing mu.
91         mu_bar = mu_bars{itr-1};  $\rightarrow \bar{\mu}^{(0)} = \mu^{(0)}$ 
92         w = Fmu - mu_bar;
93         t = norm(Fmu - mu_bar);
94     else
95         mu_bar_prev = mu_bars{itr-2};  $N^{(i-2)}$ 
96         num = (mu - mu_bar_prev)' * (Fmu - mu_bar_prev);  $(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})$ 
97         denom = (mu - mu_bar_prev)' * (mu - mu_bar_prev);  $(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})$ 
98         ratio = num/denom;
99         mu_bar = mu_bar_prev + ratio*(mu - mu_bar);
100         $\bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})} (N^{(i-1)} - \bar{\mu}^{(i-2)})$ 

```

$$\hat{\mu}_E = \frac{1}{m} \sum_{i=1}^m \sum_{t=0}^{T-1} \gamma^t \phi(S_t^{(i)})$$

states
1024
feature
125
x
1024
 $F \cdot w = R$

State expectation · muE.

i=1

225	225	225	251	251	251	251	251
-----	-----	-----	-----	-----	-----	-----	-----

t=1

$$\phi(S_0^{(1)}) = \phi(225) = F(:, 225)$$

$$\mu_E(225) = \mu_E(225) + 0.9^0 = 1$$

t=2

$$\mu_E(225) = \mu_E(225) + 0.9^1 = 1.9$$

t=3

$$\mu_E(225) = \mu_E(225) + 0.9^2 = 1.9 + 0.81 = 2.71$$

t=4

$$\mu_E(225) = \mu_E(225) + 0.9^3 = 0.9^3$$

:

t=8

$$\mu_E(225) = 0.9^3 + 0.9^4 + 0.9^5 + 0.9^6 + 0.9^7$$

주어진 policy에 대해 feature expectation은

구함.

주어진 policy에 대해 feature expectation은

$$E\left[\sum_{t=0}^{\infty} \gamma^t \right] ?$$

주어진 policy에 대해 feature expectation은

$$\bar{\mu}^{(0)} = \mu^{(0)}$$

end;

84

% Increment iteration count.

itr = itr + 1;

87

% Compute t and w using projection.

if itr == 2,

% use existing mu.

$$mu_bar = mu_bars{itr-1}; \rightarrow \bar{\mu}^{(0)} = \mu^{(0)}$$

w = Fmu - mu_bar;

t = norm(Fmu - mu_bar);

else

$$mu_bar_prev = mu_bars{itr-2}; N^{(i-2)}$$

$$num = (mu - mu_bar_prev)' * (Fmu - mu_bar_prev); (\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})$$

$$denom = (mu - mu_bar_prev)' * (mu - mu_bar_prev); (\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})$$

ratio = num/denom;

mu_bar = mu_bar_prev + ratio*(mu - mu_bar);

$$\bar{\mu}^{(i-1)} = \bar{\mu}^{(i-2)} + \frac{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu_E - \bar{\mu}^{(i-2)})}{(\mu^{(i-1)} - \bar{\mu}^{(i-2)})^T (\mu^{(i-1)} - \bar{\mu}^{(i-2)})} (N^{(i-1)} - \bar{\mu}^{(i-2)})$$

```

101     w = Fmu - mu_bar;  $w = \mu_E - \bar{\mu}^{(i-1)}$ 
102     t = norm(Fmu - mu_bar);
103     mu_bars{itr-1} = mu_bar;
104 end;
105
106 % Recompute optimal policy using new weights.
107 r = repmat(F'*w,1,actions);
108 soln = standardmdpsolve(mdp_data,r);
109 weights{itr} = w;
110 solutions{itr} = soln;
111
112 % Check convergence.
113 if (abs(t-told) <= 0.0001),
114     break;
115 end;
116
117 % Print t.
118 if verbosity ~= 0,
119     fprintf(1,'Completed IRL iteration, t=%f\n',t);
120 end;
121 end;
122
123 % Compute mu for last policy.
124 om = standardmdpfrequency(mdp_data,solutions{itr});
125 mu = F*om;
126 mus{itr} = mu;
127
128 % Construct matrix.
129 mu_mat = zeros(features,itr);
130 for i=1:itr,
131     mu_mat(:,i) = mus{i};
132 end;
133
134 % Solve optimization to determine lambda weights.
135 cvx_begin
136     if verbosity ~= 0,
137         cvx_quiet(false);
138     else
139         cvx_quiet(true);
140     end;
141     variable mu(features);
142     variable lambda(itr);
143     minimize(sum_square(mu-Fmu));
144     subject to
145         mu == mu_mat*lambda;
146         lambda >= zeros(itr,1);
147         lambda'*ones(itr,1) == 1;
148 cvx_end
149
150 % In Abbeel & Ng's algorithm, we should use the weights lambda to construct

```

```
151 % a stochastic policy. However, here we are evaluating IRL algorithms, so
152 % we must return a single reward. To this end, we'll simply pick the reward
153 % with the largest weight lambda.
154 [~,idx] = max(lambda);
155 w = weights{idx};
156
157 % Compute reward.
158 r = repmat(F'*w,1,actions);
159
160 % Compute policies.
161 soln = mdp_solve(mdp_data,r);
162 v = soln.v;
163 q = soln.q;
164 p = soln.p;
165 r_itr = cell(1,1);
166 tree_r_itr = cell(1,1);
167 p_itr = cell(1,1);
168 tree_p_itr = cell(1,1);
169 wts_itr = cell(1,1);
170 r_itr{1} = r;
171 tree_r_itr{1} = r;
172 p_itr{1} = soln.p;
173 tree_p_itr{1} = p_itr{1};
174 wts_itr{1} = w;
175 time = toc;
176
177 % Construct returned structure.
178 irl_result = struct('r',r,'v',v,'p',p,'q',q,'r_itr',{r_itr}, 'model_itr',{wts_itr},...
179     'model_r_itr',{tree_r_itr}, 'p_itr',{p_itr}, 'model_p_itr',{tree_p_itr},...
180     'time',time);
181
```

andefaultparams.m

17. 4. 18 오전 3:56 C:\Users\JEON\Desktop\Research\andefaultparams.m 1 / 1

```
% Fill in default parameters for the Abbeel & Ng algorithm.  
function algorithm_params = andefaultparams(algorithm_params)  
  
% Create default parameters.  
default_params = struct(...  
    'seed',0,...  
    'all_features',1,...  
    'true_features',0);  
  
% Set parameters.  
algorithm_params = filldefaultparams(algorithm_params,default_params);
```

이제 algorithm_params 는 비어있는 field 를 default value 로 채워넣음.

Standardmdpsolve.m

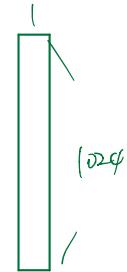
17. 4. 24 오후 11:31 C:\Users\JEON\Desktop\Research\standardmdpsolve.m 1 / 1

```
1 % Solve a standard MDP and return value function, Q function, and policy.
2 function mdp_solution = standardmdpsolve(mdp_data,r)
3 %  
4 % Run value iteration to compute the value function.
5 v = stdvalueiteration(mdp_data,r);
6
7 % Compute Q function and policy.
8 [q,p] = stdpolicy(mdp_data,r,v);
9
10 % Return solution.
11 mdp_solution = struct('v',v,'q',q,'p',p);
12
```

StdValueIteration.m

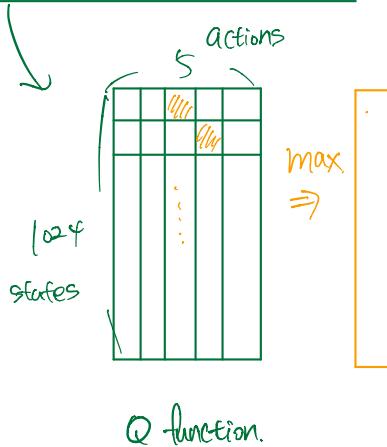
17. 4. 24 오후 11:31 C:\Users\JEON\Desktop\Rese...\\stdvalueiteration.m 1 / 1

```
1 % Run value iteration to solve a standard MDP.  
2 function v = stdvalueiteration(mdp_data,r,vinit)  
3  
4 % Allocate initial value function & variables.  
5 diff = 1.0;  
6 if (nargin == 3)  
7     vn = vinit;  
8 else  
9     vn = zeros(mdp_data.states,1);  
10 end;          (024)  
11  
12 % Perform value iteration.  
13 while diff >= 0.0001,  
14     vp = vn; (Rt+1) + (P(s'|s,a) * V(s')) * γ  
15     vn = max(r + sum(mdp_data.sa_p.*vp(mdp_data.sa_s),3)*mdp_data.discount,[],2); →  
16     diff = max(abs(vn-vp)); (024x5x5)    (024x5x5)  
17 end;  
18  
19 % Return value function.  
20 v = vn;  
21
```



각 state의 value 값이
가장 큰 action의 value를
value function으로 채워진다.

```
1 % Given reward and value functions, solve for q function and policy.  
2 function [q,p] = stdpolicy(mdp_data,r,v)  
3  
4 % Compute Q function.  
5 q = r + sum(mdp_data.sa_p.*v(mdp_data.sa_s),3)*mdp_data.discount;  
6  
7 % Compute policy.  
8 [~,p] = max(q,[],2);  
9
```



Standardmdpfrequency.m

17. 4. 25 오전 12:02 C:\Users\JEON\Desktop\...\standardmdpfrequency.m 1 / 1

```
1 % Compute the occupancy measure of the MDP given a policy.
2 function x = standardmdpfrequency(mdp_data, mdp_solution)
3
4 % Build flow constraint matrix.
5 A = sparse([],[],[], mdp_data.states, mdp_data.states,...)    1024x1024 sparse matrix
6 mdp_data.states * size(mdp_data.sa_p,3);                         s
7 for s=1:mdp_data.states,                                         1024x5
8     A(s,s) = A(s,s)+1;                                           s
9     a = mdp_solution.p(s); → state s에 대해 각 action a의 policy π(s,a)
10    for k=1:size(mdp_data.sa_p,3),                                1024x5
11        sp = mdp_data.sa_s(s,a,k);
12        A(sp,s) = A(sp,s) - mdp_data.discount * mdp_data.sa_p(s,a,k);
13    end;
14 end;
15
16 % Solve linear system to get occupancy measure.
17 x = A \ ((1/mdp_data.states) * ones(mdp_data.states, 1));
18
```

$$A \mathbf{x} = \frac{1}{1024} \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}, \quad 1024 \times 1024$$

Occupancy Frequencies

The occupancy frequency $f(s,a)$ expresses the total discounted probability of being in state s and taking action a .

Given $f(s,a)$ we can recover the policy π

$$\pi(s,a) = \frac{f^\pi(s,a)}{\sum_a f^\pi(s,a)}$$

The value of optimal π^* corresponds to the value of the optimal f^*

$$V_r^{\pi^*} = r \cdot f^*$$

```
% Evaluate result returned by IRL algorithm on given MDP.
function test_result = evaluateirl(irl_result,true_r,example_samples,%
    mdp_data,mdp_params,mdp_solution,mdp,~,test_models,test_metrics,%
    feature_data,true_feature_map)

% test_result - data structure encapsulating generic test results:
% metric_scores - array of results from each metric, for each MDP type
% irl_result - copy of result from IRL algorithm
% true_r - true reward function
% example_samples - copy of example samples
% mdp_data - copy of MDP data
% mdp_params - copy of MDP parameters
% mdp_solution - copy of true MDP solution
% feature_data - copy of feature data
% mdp - type of MDP used

metric_scores = cell(length(test_models),length(test_metrics));
for m=1:length(test_models),      standardmdp.linearmdp8
    % Evaluate for each test model.
    cur_model = test_models{m};
    mdp_solve = str2func(strcat(cur_model,'solve'));
    irl_soln = mdp_solve(mdp_data,irl_result.r);
    mdp_soln = mdp_solve(mdp_data,true_r);
    irl_r = irl_result.r;

    % Evaluate each metric.
    for k=1:length(test_metrics),
        cur_metric = test_metrics{k};
        metric_scores{m,k} = feval(strcat(cur_metric,'score'),mdp_soln,%
            true_r,irl_soln,irl_r,feature_data,true_feature_map,%
            mdp_data,mdp_params,cur_model);
    end;
end;

% Build results structure.
test_result = struct(...%
    'metric_scores',{metric_scores},...%
    'irl_result',irl_result,...%
    'true_r',true_r,...%
    'example_samples',{example_samples},...%
    'test_models',{test_models},...%
    'test_metrics',{test_metrics},...%
    'mdp_data',mdp_data,...%
    'mdp_params',feval(strcat(mdp,'defaultparams'),mdp_params),...%
    'mdp_solution',mdp_solution,...%
    'feature_data',feature_data,...%
    'mdp',mdp);
```