Optimal Policies Value Iteration

MICHAEL WOLLOWSKI

Grid World

We already introduced the simple world that our agent is to explore.

Let's add a kink into our simple world.

Suppose actions do not always go as planned.

In technical terms, we move to a stochastic transition model.

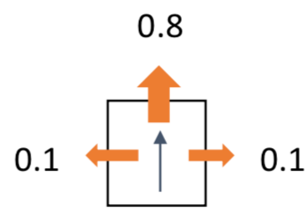
3				+1
2		W		-1
1	*Start*			
	1	2	3	4

Stochastic Transition Function

In particular, a planned action has an 80% probability of succeeding.

In 10% of the cases, rather than moving straight ahead, the agent ends up moving to its right and

In 10% of the cases, the agent moves to its left.

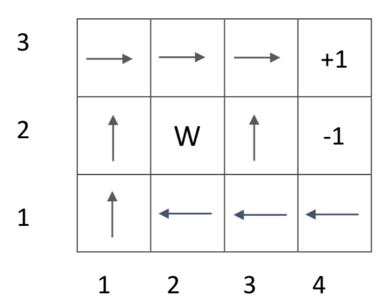


Optimal Policies

Optimal policy: For every state, there is no other action that gets a higher sum of discounted future rewards.

An optimal policy for the stochastic environment with:

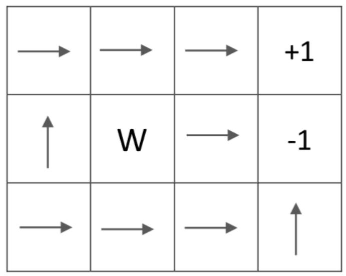
- R(s) = -0.04



Optimal Policies

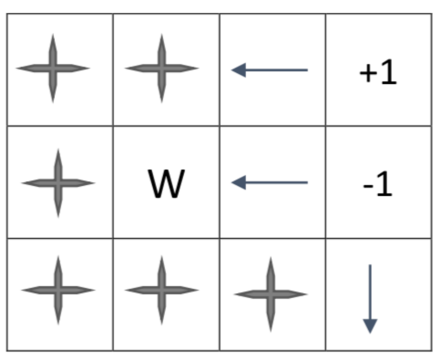
To understand the effect of the utilities on policies, let's have a look at some odd policies.

What do you think the utilities are for this policy?



Optimal Policies

How about for this policy?



From Values To Policies

One way to determine an optimal policy is to:

1. determine the values/utilities of each state,
2. followed by determining for each state the neighboring state with the highest value.

3	0.812	0.868	0.918	+1
2	0.762	W	0.660	-1
1	0.705	0.655	0.611	0.388
	1	2	3	4

⇨

3	→	→	→	+1
2	↑	W	↑	-1
1	↑	←	←	←
	1	2	3	4

From Values To Policies

For each state, calculate:

$$\underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s' \mid s, a)\, U(s')$$

Consider state <1,1>

There are four possible actions:
- up
- down
- left
- right

3	0.812	0.868	0.918	+1
2	0.762	W	0.660	-1
1	0.705	0.655	0.611	0.388
	1	2	3	4

From Values To Policies

For the action “up,” according to our transition function, we have three cases to consider:

- We succeeded in moving up:

  P(<1,2> | <1,1>, up) U(<1,2>)

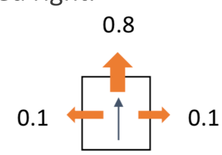
- We attempted to move up but instead headed left, but that means falling off the edge of our world, so we stay put:

  P(<1,1> | <1,1>, up) U(<1,1>)

- We attempted to move up but instead headed right.

  P(<1,2> | <1,1>, up) U(<1,2>)

$$\underset{a \in A(s)}{\text{argmax}} \sum_{s'} P(s' | s, a)\, U(s')$$



From Values To Policies

We calculate the values of those three terms:

P(<1,2> | <1,1>, up) U(<1,2>) = 0.8 * 0.762

P(<1,1> | <1,1>, up) U(<1,1>) = 0.1 * 0.705

P(<1,2> | <1,1>, up) U(<1,2>) = 0.1 * 0.655

And sum them, giving:

0.7456

We now need to calculate the values for the remaining three actions: down, left and right.

Please use the worksheet to do so.

3	0.812	0.868	0.918	+1
2	0.762	W	0.660	-1
1	0.705	0.655	0.611	0.388
	1	2	3	4

From Values To Policies

Based on the value we calculate and the data from your worksheet, we should have:

- up: 0.7456
- down: 0.697
- left: 0.7107
- right: 0.6707

Based on the formula, we select the action up, since it leads us in the direction of the highest reward in the end.

$$\underset{a \in A(s)}{\text{argmax}} \sum_{s'} P(s' \mid s, a)\, U(s')$$

Values/Utilities of a State

The utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, assuming that the agent chooses the optimal action.

The utility of a state is given by the Bellman equation:

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U(s')$$

Values/Utilities of a State

Notice the similarities to the function with which we calculate a policy.

Here, we do not take the action that lead to the max, but instead the value of the max.

We multiply the value with a tuning factor g that determines the degree to which we favor the immediate reward over later rewards.

We will explore the tuning factor later.

$$U(s) = R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' | s, a) U(s')$$

Calculating Utility

Consider the following world.

Using the worksheet, calculate the utility of state <3, 3>, using 0.9 for γ

3	-0.04	-0.04	-0.04	+1
2	-0.04	W	-0.04	-1
1	*Start*	-0.04	-0.04	-0.04
	1	2	3	4

Calculating Utility

You should have calculated the utility as follows:

U(3,3) = -0.04 + 0.9 max[0.8U(3,4) + 0.1U(3,3) + 0.1U(2,3), // right

0.8U(2,3) + 0.1U(3,2) + 0.1U(3,4), // down

0.8U(3,2) + 0.1U(3,3) + 0.1U(2,3), // left

0.8U(3,3) + 0.1U(3,2) + 0.1U(3,4)] // up

which gives: 0.6728

3	-0.04	-0.04	-0.04	+1
2	-0.04	W	-0.04	-1
1	*Start*	-0.04	-0.04	-0.04
	1	2	3	4

Value Iteration

The value iteration algorithm is a way to calculate utilities.

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s' \mid s, a)$,
rewards $R(s)$, discount γ
ϵ, the maximum error allowed in the utility of any state
local variables: U, U', vectors of utilities for states in S, initially zero
δ, the maximum change in the utility of any state in an iteration

repeat
$U \leftarrow U'$; $\delta \leftarrow 0$
for each state s **in** S **do**
$U'[s] \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a)\, U[s']$
if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
until $\delta < \epsilon(1-\gamma)/\gamma$
return U

Algorithm source: Russell and Norvig: AIMA 2nd Ed.