9. Complex decision making agents

By: Alex S.

Apr, 2024

1. Making complex decisions

- Complex decisions represent a sequence of decisions where the result is not known by doing one action.

Let's imagine a new world in Figure 1.

- Board is 4 x 3
- Square (4, 2) has a reward of –1
- Square (4, 3) has a reward of 1
- Square (2, 2) has an obstacle
- Square (1, 1) is a starting position

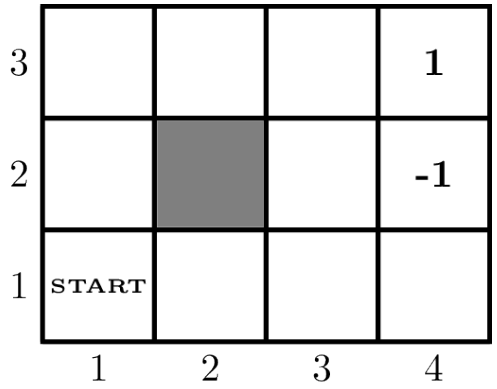


Figure 1: Environment representation

1.1. Deterministic environment

Let's imagine that the agent has 2 actions to plan ahead. The agent has complete information about the environment. For being in every state, the agent receives -0.04 points or receives points from (4, 2) and (4, 3) squares.

Let's imagine that the agent is in the square (3, 2). Then the decision tree will look like this(Figure 2).

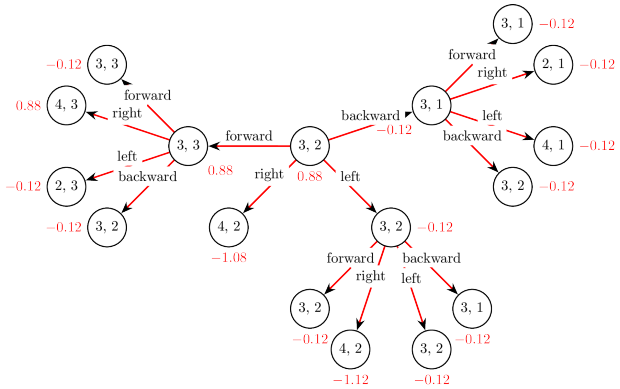


Figure 2: Deterministic environment representation

As can be seen in Figure 2, the agent can easily obtain the optimal path $(3,2) \rightarrow (3,3) \rightarrow (4,3)$. However, in real life the tree is much bigger than presented here.

1.2. Stochastic environment

Stochastic environment is more complex and requires a model for the agent to obtain. For the example, let's obtain that the environment's model is the following:

1. The agent might go to the planned location with the probability of 80%
2. The agent might go to the perpendicular directions with the probability 10% each

This model can be represented as in Figure 3.

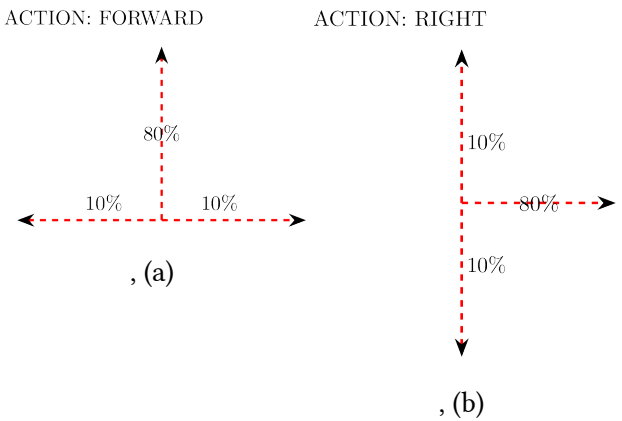


Figure 3: Move models. (a) for the forward action, (b) for the right action

If we consider that the agent can do only a pair of actions sequentially, for example, (FORWARD, RIGHT), then we can construct a tree with all consequences. There are going to be $2^4 = 16$ trees overall. After that we can maximize our utility by choosing the tree with maximum value.

In Figure 4 is shown a tree for the action pair (FORWARD, RIGHT). Note that now we have only 3 actions due to the model. We cannot move backwards.

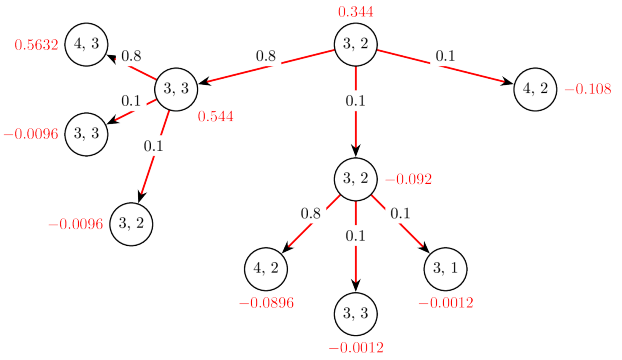


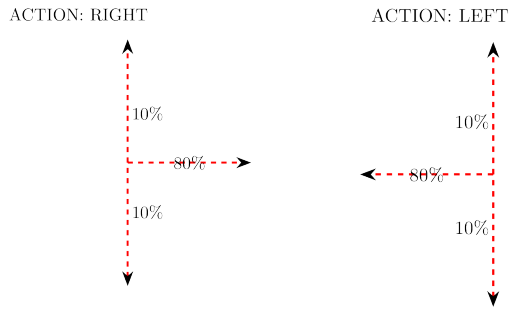
Figure 4: Stochastic environment representation

To calculate the final value of leaf nodes(v_{leaf}) we have to multiply a prior value(v) with the parents' probabilities(p):

$$v_{\text{leaf}} = \Pi_{i \in \text{ parents(leaf)}} p_i \cdot v$$

For example, the path $(3,2) \to (3,3) \to (4,3)$ value is $0.8 \cdot 0.8 \cdot (1 - 0.12) = 0.5632$. After the value for each node is calculated, it is summed to get the overall value of the root. In this case $v_{\text{root}} = 0.344$.

Other example is the model of the actions (RIGHT, LEFT) in Figure 5.



(a) (b)

Figure 5: Move models. (a) for the right action, (b) for the left action

The decision tree of this model looks like in Figure 6. In this case the sequence is not optimal. It produces negative value at the root($v_{\text{root}} = -0.888$).

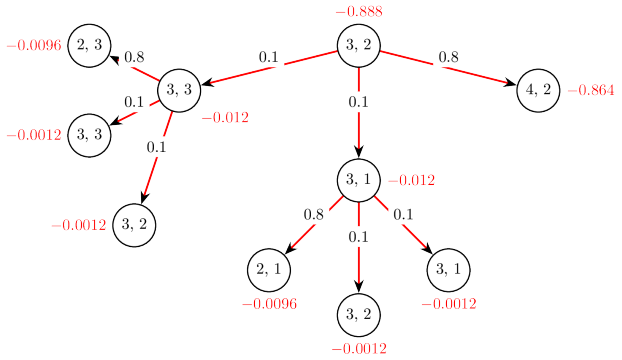


Figure 6: Second sequence stochastic environment representation

After all trees are constructed, we can make a decision on where to go(Figure 7). It turns out that the first sequence is the best one to choose.

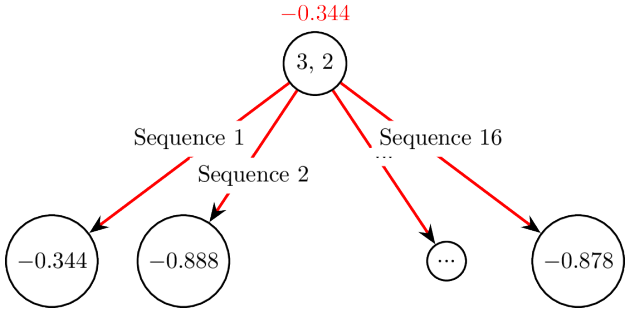


Figure 7: Final decision tree of stochastic environment where the best sequence is chosen.

2. Markovian decision process

Transition model (or just "model") describes the outcome of each action in each state. The outcome is stochastic, so we write $P(s' \mid s, a)$ to denote the probability of reaching state s' if action a is done in state s. We will assume that transitions are **Markovian**.

Because the decision problem is sequential, the utility function will depend on a sequence of states — an **environment history** — rather than on a single state.

A sequential decision problem for a fully observable, stochastic environment with a **Markovian** transition model and additive rewards is called a **Markov decision process(MDP)**. It is defined by:

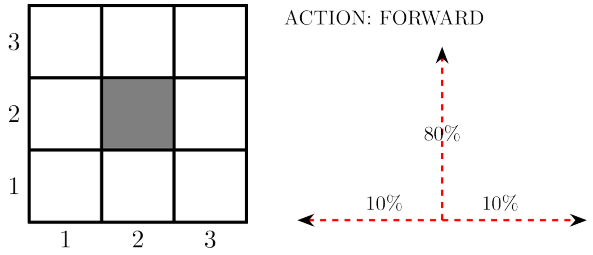
1. Initial state s_0, e.g. $s_0 = (1,1)$
2. A set of actions in each state $A(s)$
3. a reward function $R(s)$, e.g. in the example:

$$\begin{cases} \pm 1 \text{ if terminal state} \\ -0.04 \text{ else} \end{cases}$$

4. A transition model $P(s' \mid s, a)$ or M_{ij}

3. A transition model

Let's consider a simpler example of the game in the first chapter. We need to create a transition model M^{forward} that defines all probabilities transitioning from one state to another.



,

,

Figure 8: A simpler environment and a move model

Then $M^{\text{forward}} =$

	(1, 1)	(1, 2)	(1, 3)	(2, 1)	(2, 3)	(3, 1)
(1, 1)	0.1	0.8	0	0.1	0	0
(1, 2)	0	0.2	0.8	0	0	0
(1, 3)	0	0	0.9	0	0.1	0
(2, 1)	0.1	0	0	0.8	0	0.1
(2, 3)	0	0	0.1	0	0.8	0.1

Note: 2 columns were cut to save space. The notation of coordinates: (column, row).

4. Policy

A solution must specify what the agent should do for any state that the agent might reach. A solution of this kind is called a **policy**. In other words a state-action mapping.

Policy is denoted by π and $\pi(s)$ is the action recommended by the policy π for state s.

Optimal policy(denoted by π^*) is a policy that yields the highest expected utility(EU):

$$\text{EU}(a|e) = \sum_i P(\text{result}(a_i) = s' \mid a_i, e) \cdot U(s')$$

Policy provides a balance between a risk and a reward.

5. Finding optimal policy

Our analysis draws on **multiattribute utility theory** and we use an environment history to measure a sum of rewards: $U_h(s_0, s_1, ..., s_n)$

There could be:

1. A **finite horizon** for decision making. It means that there is a fixed time N after which nothing matters - the game is over. Thus

   $$U_h(s_0, s_1, ..., s_{N+k}) = U_h(s_0, s_1, ..., s_N)$$

   for all $k > 0$.

   The optimal policy with the finite horizon is **non-stationary**, i.e. N value changes the behavior of the policy.

2. An **infinite horizon** for decision making - no time or step limit.

   The optimal policy with **infinite horizon** is **stationary**.

Stationarity for preferences means the following: if two state sequences $(s_0, s_1, s_2, ...)$ and $(s'_0, s'_1, s'_2,)$ begin with the same state (i.e., $s_0 = s'_0$).

Stationarity is a fairly innocuous-looking assumption with very strong consequences: it turns out that under stationarity there are just two coherent ways to assign utilities to sequences:

1. **Additive rewards**: The utility of a state sequence is

   $$U_h(s_0, s_1, s_2, ...) = R(s_0) + R(s_1) + R(s_2) + ...$$

2. **Discounted rewards**:

   $$U_h(s_0, s_1, s_2, ...) = R(s_0) + \alpha R(s_1) + \alpha^2 R(s_2) + ...$$

   where $0 < \alpha < 1$ is a discount factor. It describes the preference of an agent for current rewards over future rewards. It is equivalent to an interest rate of $\frac{1}{\alpha} - 1$

With discounted rewards, the utility of an infinite sequence is finite. In fact, if $\alpha < 1$ and rewards are bounded by $\pm R_{\max}$, we have:

$$U_h(s_0, s_1, s_2, ...) = \sum_{t=0}^{\infty} \alpha^t R(s_t) \leq \sum_{t=0}^{\infty} \alpha^t R_{\max} = \frac{R_{\max}}{1 - \alpha}$$

Using the standard formula for the sum of an infinite geometric series.

If the environment contains terminal states and if the agent is guaranteed to get to one eventually, then we will never need to compare infinite sequences.

The utility of a given state sequence is the sum of discounted rewards obtained during the sequence, we can compare policies by comparing the expected utilities obtained when executing them. The expected utility is:

$$U^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \alpha^t \cdot R(S_t)\right]$$

where $U^\pi(s)$ is the utility on the state s using a specified policy π, α - discount factor, S - state sequence, e.g. $S_0 = s$ - the current state.

Note: $R(s)$ is the "short term" reward for being in s, whereas $U(s)$ is the "long term" total reward from s onward.

Out of all policies the agent could choose one best policy:

$$\pi^*(s) = \underset{\pi}{\text{argmax}}\ U^\pi(s)$$

Remember that π^*_s is a policy, so it recommends an action for every state; its connection with s in particular is that it's an optimal policy when s is the starting state.

The utility function $U(s)$ allows the agent to select actions by using the principle of maximum expected utility:

$$\pi_s^* = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s' \mid s, a) \cdot U(s')$$

Note: From the transitional model M_{ij} we can imagine that $i = s$ and $j = s'$, therefore there is a transition $M_{i=s, j=s'}$

6. Finding optimal policies: Bellman equation

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:

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

Let us look at one of the Bellman equations for 4x3 world for the state (1, 1):

$$\begin{aligned} U(1,1) = -0.04 + \\ +\alpha \cdot \max[&0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), \text{(Up)} \\ &0.9U(1,1) + 0.1U(1,2), \text{(Left)} \\ &0.9U(1,1) + 0.1U(2,1), \text{(Down)} \\ &0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)] \text{ (Right)} \end{aligned}$$

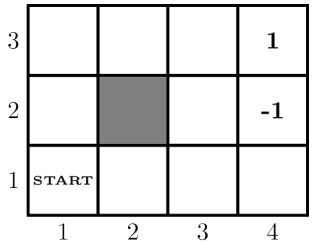


Figure 9: A reference to the board

7. The value iteration algorithm

NEED TO COME BACK TO THIS PLACE AND ADD ADDITIONAL COMMENTS FROM THE BOOK!!! page 652

$$U_{i+1}(s) = R(s) + \alpha \max_{a \in A(s)} \sum_{s'} P(s' \mid s, a) \cdot U_i(s')$$

where U_{i+1} is the utility on the next iteration

$$\pi^*(s) = \underset{a \in A(s)}{\operatorname{argmax}} \sum_{s'} P(s' \mid s, a) \cdot U(s')$$

Algorithm(formal description from the book):

7.1. Example:

- $\alpha = 1$
- Every state has $\forall s \; U(s) = 0$

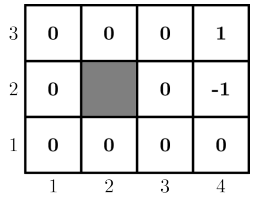


Figure 10: Starting state

FIRST ITERATION:

1. $U(3,3)$:

(Up): $-0.04 + 0.8 \cdot 0 + 0.1 \cdot 0 + 0.1 \cdot 1 = 0.06$

(Right): $-0.04 + 0.8 \cdot 1 + 0.1 \cdot 0 + 0.1 \cdot 0 = 0.76$

(Left): $-0.04 + 0.8 \cdot 0 + 0.1 \cdot 0 + 0.1 \cdot 0 = -0.04$

(Down): $-0.04 + 0.8 \cdot 0 + 0.1 \cdot 0 + 0.1 \cdot 1 = 0.06$

$\Rightarrow \max U(3,3) = 0.76$

$\Rightarrow \pi^* = \text{Right}$

2. $U(3,2)$:

(Up): $-0.04 + 0.8 \cdot 0 + 0.1 \cdot 0 + 0.1 \cdot (-1) = -0.14$

(Right): $-0.04 + 0.8 \cdot (-1) + 0.1 \cdot 0 + 0.1 \cdot 0 = -0.84$

(Left): $-0.04 + 0.8 \cdot 0 + 0.1 \cdot 0 + 0.1 \cdot 0 = -0.04$

(Down): $-0.04 + 0.8 \cdot 0 + 0.1 \cdot (-1) + 0.1 \cdot 0 = -0.14$

$\Rightarrow \max U(3,2) = -0.04$

$\Rightarrow \pi^* = \text{Left}$

3. $U(4,1)$:

(Up): $-0.04 + 0.8 \cdot (-1) + 0.1 \cdot 0 + 0.1 \cdot 0 = -0.84$

(Right): $-0.04 + 0.8 \cdot 0 + 0.1 \cdot (-1) + 0.1 \cdot 0 = -0.14$

(Left): $-0.04 + 0.8 \cdot 0 + 0.1 \cdot (-1) + 0.1 \cdot 0 = -0.14$

(Down): $-0.04 + 0.8 \cdot 0 + 0.1 \cdot 0 + 0.1 \cdot 0 = -0.04$

$\Rightarrow \max U(4,1) = -0.04$

$\Rightarrow \pi^* = \text{Down}$

Other positions produce the value of $U(s) = -0.04$

The final results after the first iteration in Figure 11.

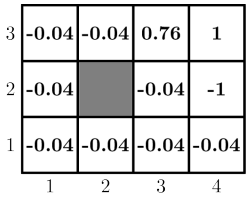


Figure 11: The values after the first iteration

SECOND ITERATION:

1. $U(3,1)$: **NEED TO CHECK RESULTS HERE!**

(Up): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot (-0.04) + 0.1 \cdot (-0.04) = -0.08$

(Right): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot (-0.04) + 0.1 \cdot (-0.04) = -0.08$

(Left): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot (-0.04) + 0.1 \cdot (-0.04) = -0.08$

(Down): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot (-0.04) + 0.1 \cdot (-0.04) = -0.08$

$\Rightarrow \max U(4,1) = -0.08$

$\Rightarrow \pi^* = ?$

2. $U(3,3)$:

(Up): $-0.04 + 0.8 \cdot 0.76 + 0.1 \cdot (-0.04) + 0.1 \cdot 1 = 0.644$

(Right): $-0.04 + 0.8 \cdot 1 + 0.1 \cdot 0.76 + 0.1 \cdot (-0.04) = 0.832$

(Left): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot 0.76 + 0.1 \cdot (-0.04) = 0$

(Down): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot 1 + 0.1 \cdot (-0.04) = 0.024$

$\Rightarrow \max U(3,3) = 0.832$

$\Rightarrow \pi^* =$ Right

3. $U(3,2)$:

(Up): $-0.04 + 0.8 \cdot 0.76 + 0.1 \cdot (-0.04) + 0.1 \cdot (-1) = 0.464$

(Right): $-0.04 + 0.8 \cdot (-1) + 0.1 \cdot 0.76 + 0.1 \cdot (-0.04) = 0.768$

(Left): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot 0.76 + 0.1 \cdot (-0.04) = 0$

(Down): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot (-0.04) + 0.1 \cdot (-1) = 0.176$

$\Rightarrow \max U(3,2) = 0.464$

$\Rightarrow \pi^* =$ Up

Other positions produce the value of $U(s) = -0.08$

The final results after the first iteration in Figure 12.

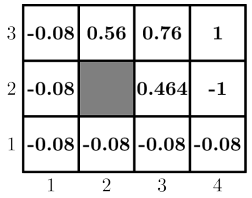
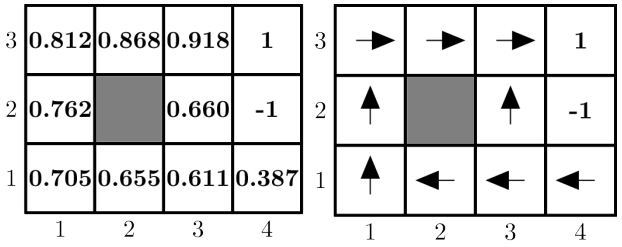


Figure 12: The values after the second iteration

The same actions are done for all states. The convergence can be received after 18 iterations in Figure 13



, ,

Figure 13: The values after 18 iterations. The convergence is received. On the right image we see the optimal policy π^*

7.2. Rewards

Depending on the constant reward value, the agent will behave differently.

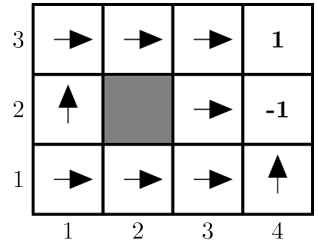


Figure 14: $R(s) < -1.684$ - everything is so bad, so the agent decides to suicide right away

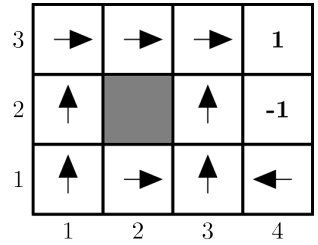


Figure 15: $-0.427 < R(s) < -0.08$ - The agent becomes quite risky

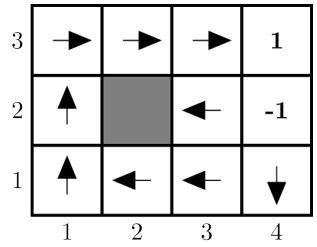


Figure 16: $-0.0221 < R(s) < 0$ - The agent becomes to play too safely

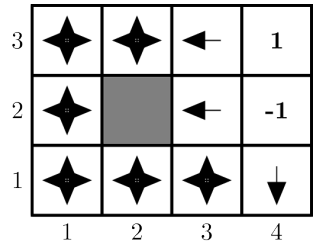


Figure 17: $R(s) > 0$ - The agent does not want to die, and it wants to leave in this world for so long it is possible.

8. Policy iteration algorithm

It consists of 2 steps:

1. Policy evaluation During the policy evaluation, we don't need max of arguments. That way we get a system of linear equations, instead of non-linear. It should be faster to calculate.

$$U_i(s) = R(s) + \alpha \sum_{s'} P(s' \mid s, \pi_i(s)) \cdot U(s')$$

2. Policy improvement(Finding better π^*)

   If this inequality holds:

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

then:

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

8.1. Example

Let's imagine that we have a random policy like in Figure 18.

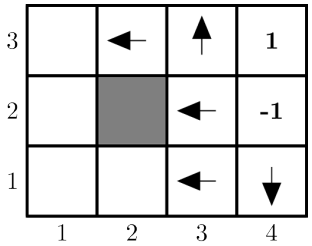


Figure 18: A part of randomly created policy

1. Let's calculate the policy evaluation

$U(3,2) = -0.04 + 0.8 \cdot 0 + 0.1 \cdot 0 + 0.1 \cdot 0 = -0.04$

$U(3,3) = -0.04 + 0.8 \cdot 0 + 0.1 \cdot 0 + 0.1 \cdot 1 = 0.06$

$U(3,1) = -0.04 + 0.8 \cdot 0 + 0.1 \cdot 0 + 0.1 \cdot 0 = -0.04$

$U(4,1) = -0.04$

$U(2,3) = -0.04$

We get the following distribution in

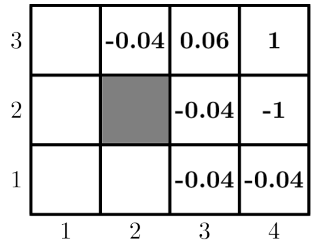


Figure 19: A random policy value distribution

2. Can we improve it?

- *Note*: We use updated values in in this iteration

$U(3,2)$:

(Up): $-0.04 + 0.8 \cdot 0.06 + 0.1 \cdot (-0.04) + 0.1 \cdot (-0.04) = -0.096$

(Right): $-0.04 + 0.8 \cdot (-1) + 0.1 \cdot 0.06 + 0.1 \cdot (-0.04) = -0.838$

(Left): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot 0.06 + 0.1 \cdot (-0.04) = -0.07$

(Down): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot (-1) + 0.1 \cdot (-0.04) = -0.176$

In this case the equation does not hold, because $U(3,2) = -0.04$:

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

Therefore the policy does not change:

$\pi^*(3,2) = \leftarrow$

$U(3,3)$:

(Up): $-0.04 + 0.8 \cdot 0.06 + 0.1 \cdot 1 + 0.1 \cdot (-0.04) = 0.104$

(Right): $-0.04 + 0.8 \cdot 1 + 0.1 \cdot 0.06 + 0.1 \cdot (-0.04) = 0.762$

(Left): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot 0.06 + 0.1 \cdot (-0.04) = -0.07$

(Down): $-0.04 + 0.8 \cdot (-0.04) + 0.1 \cdot 1 + 0.1 \cdot (-0.04) = 0.024$

In this case the equation does hold, because $U(3,3) = 0.06 < 0.762$.

Therefore the policy does change:

$\pi(3,3) = \uparrow (U(3,3) = 0.06)$

$\pi^*(3,3) = \rightarrow (U(3,3) = 0.762)$

For policy iteration algorithm, it is required to have only 5 iterations for the algorithm to converge to stable values. In comparison to value iteration process, where 18 were needed.