Bellman and Value Iteration

Frederik Mallmann-Trenn
6CCS3AIN

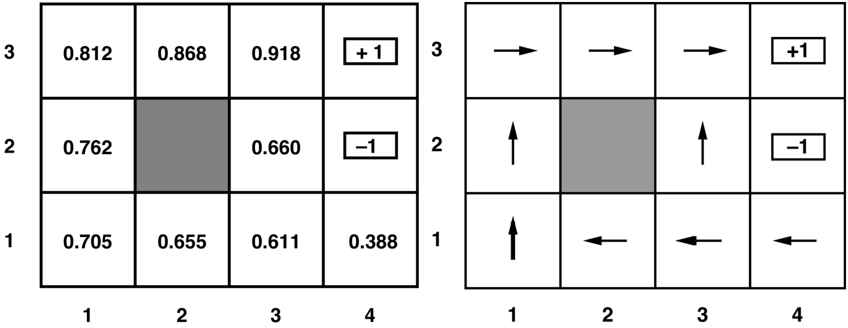
Optimal policies

- If we have the correct utility values, the agent has a simple decision process
- It just picks the action a that maximises the expected utility of the next state:

$$\pi^*(s) = \arg\max_{a \in A(s)} \sum_{s'} P(s'|s,a) U^{\pi^*}(s')$$

- Only have to consider the next step.
- The big question is how to compute $U^{\pi^*}(s)$.

Optimal policies



- Note that this is specific to the value of the reward $R(s)$ for non-terminal states — different rewards will give different values and policies.

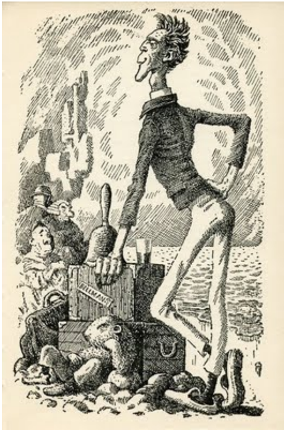
Bellman equation

- How do we find the best policy (for a given set of rewards)?
- Turns out that there is a neat way to do this, by first computing the utility of each state.
- We compute this using the Bellman equation

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

- γ is a discount factor.

Not this Bellman



”Just the place for a Snark!” the Bellman cried,
As he landed his crew with care;
Supporting each man on the top of the tide
By a finger entwined in his hair.

”Just the place for a Snark! I have said it twice:
That alone should encourage the crew.
Just the place for a Snark! I have said it thrice:
What I tell you three times is true.”

Lewis Carroll

(Mervyn Peake’s illustrations to “The Hunting of the Snark”).

Bellman equation

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

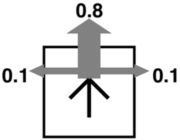
- Apply:

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

and we get:

Bellman equation

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



$$U(1,1) = -0.04 + \gamma\, max \left\{ \begin{array}{ll} 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), & (Up) \\ 0.9U(1,1) + 0.1U(1,2), & (Left) \\ 0.9U(1,1) + 0.1U(2,1), & (Down) \\ 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1) & (Right) \end{array} \right\}$$

Value iteration

- In an MDP wth n states, we will have n Bellman equations.

(Pendleton Ward/Cartoon Network)

- Hard to solve these simultaneously because of the max operation
  - Makes them non-linear

Value iteration

- Luckily an iterative approach works.
- Start with arbitrary values for states.
- Then apply the Bellman update:

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_{a \in A(s)} \sum_{s'} P(s'|s,a) U_i(s')$$

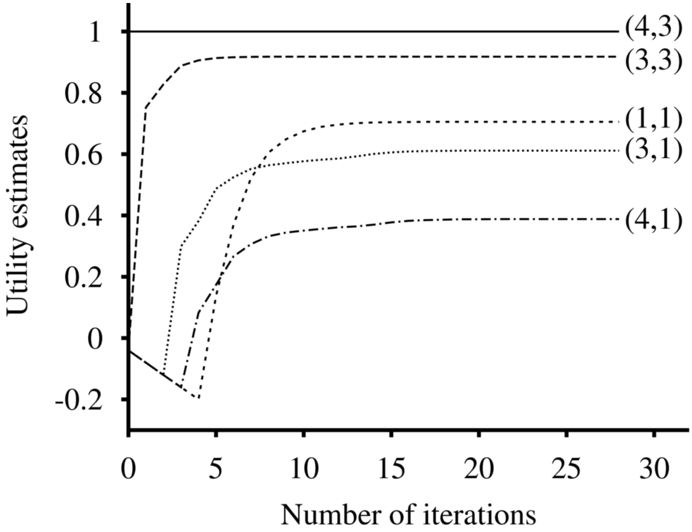
  simultaneously to all the states.
- Continue until the values of states do not change.
- The values are guaranteed to converge on the optimal values (but might take some time).

Value iteration

procedure VALUE ITERATION
  for s in S **do**
    $U(s) \leftarrow 0$
  end for
  repeat
    $U_{copy} \leftarrow U$
    for s in S **do**
      $U(s) \leftarrow R(s) + \gamma \max_{a \in A(S)} \sum_{s'} P(s'|s,a) U_{copy}(s')$
    end for
  until $U == U_{copy}$
end procedure

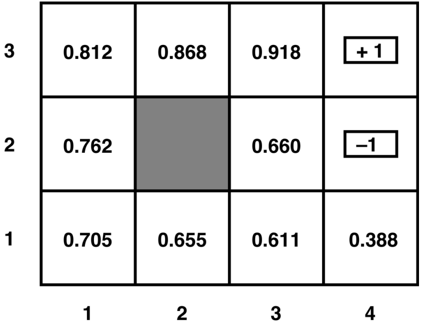
- States, S, reward, $R(s)$, set of actions, $A(s)$ and transition model, $P(s'|s,a)$, are exactly as defined earlier.
- γ is the discount factor.

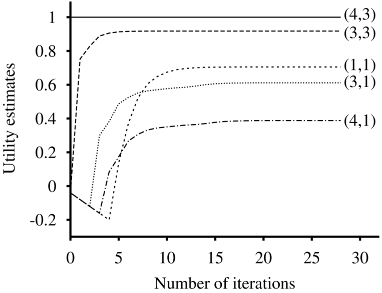
Value iteration



- How the values of states change as updates occur.

Value iteration





- $U(4,3)$ is pinned to 1.
- $U(3,3)$ quickly settles to a value close to 1
- $U(1,1)$ becomes negative, and then grows as positive utility from the goal feeds back to it.

Rewards

- The example so far has a negative reward $R(s)$ for each state.
- Encouragement for an agent not to stick around.
- Can also think of $R(s)$ is being the cost of moving to the next state (where we obatin the utility):

$$R(s) = -c(s, a)$$

  where s is the action used.
- Bellman becomes:

$$U_{i+1}(s) \leftarrow \gamma \max_{a \in A(s)} (\sum_{s'} P(s'|s, a) U_i(s')) - c(s, a)$$

- Note that the action can be dependent on the state.