Workshop 7

Recap: Classical Planning Problem

Not every problem belongs to classical planning problem

Deterministic action: S – a -> S’

- Every action only has a certain outcome, and you know what that outcome will be
- Counterexample: coin toss -> probabilistic actions
- Single-agent
- Static environment
- ……

Other action types

- **Probabilistic:** We could possibly end up in more than one state, and we know the probability distribution of these states (Example: Toss a fair coin)
- **Non-deterministic:** We know all possible outcome, but not the probability distribution
- **Stochastic:** limited info about possible outcomes

MDP problem

- Still use model-based approach to solve it

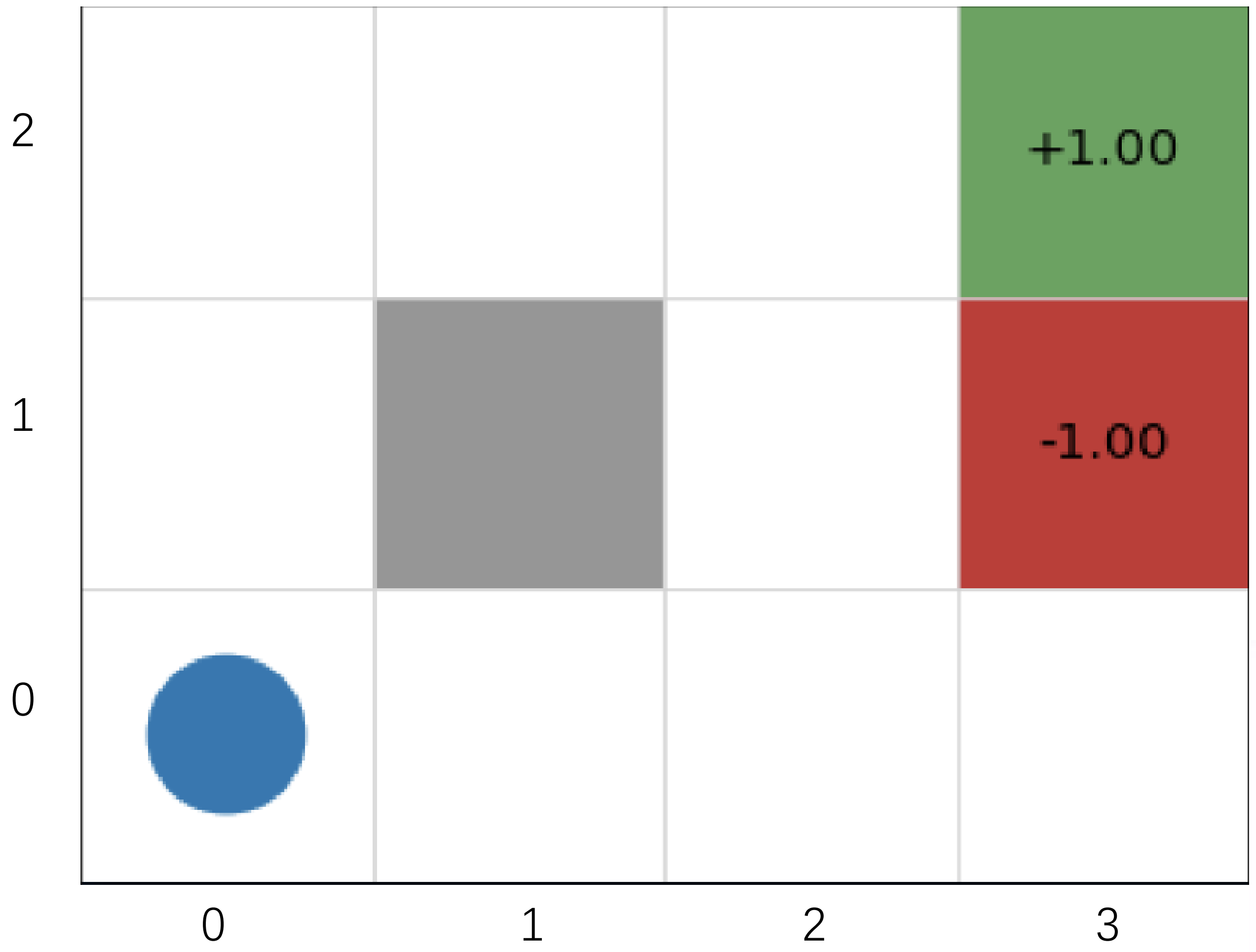
2 Models:

- **Goal-cost MDP model**: with a set of specific goal state, intend to achieve some goals, objective: minimize our cost to the goal
- **Discounted reward MDP model**: don’t have goal state, have terminal state instead, objective: maximize the reward

2 Solvers:

- Value Iteration
- Policy Iteration

Lecture Example



0.8 succ
0.1 slip left
0.1 slip right

Representations

S = {<x,y> | x belongs to (0,3), y belong to {0,2}} U {s_t} \ (1,1)}

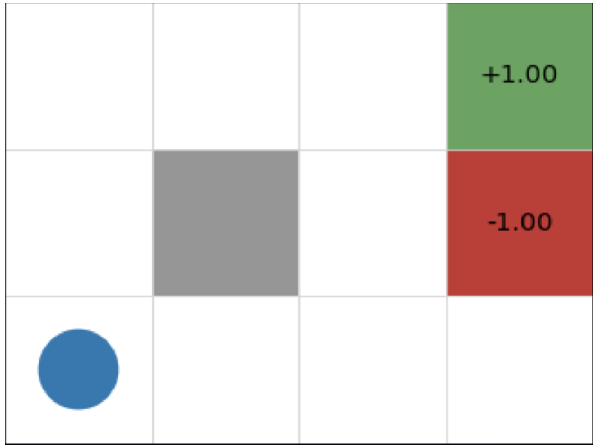
s0 = <0, 0>   S_T = {s_t}

Action function:

A(s_t) = {}

A(s) = {N,W,E,S}

except A((3,2)) = A((3,1)) = {exit}



Reward function:

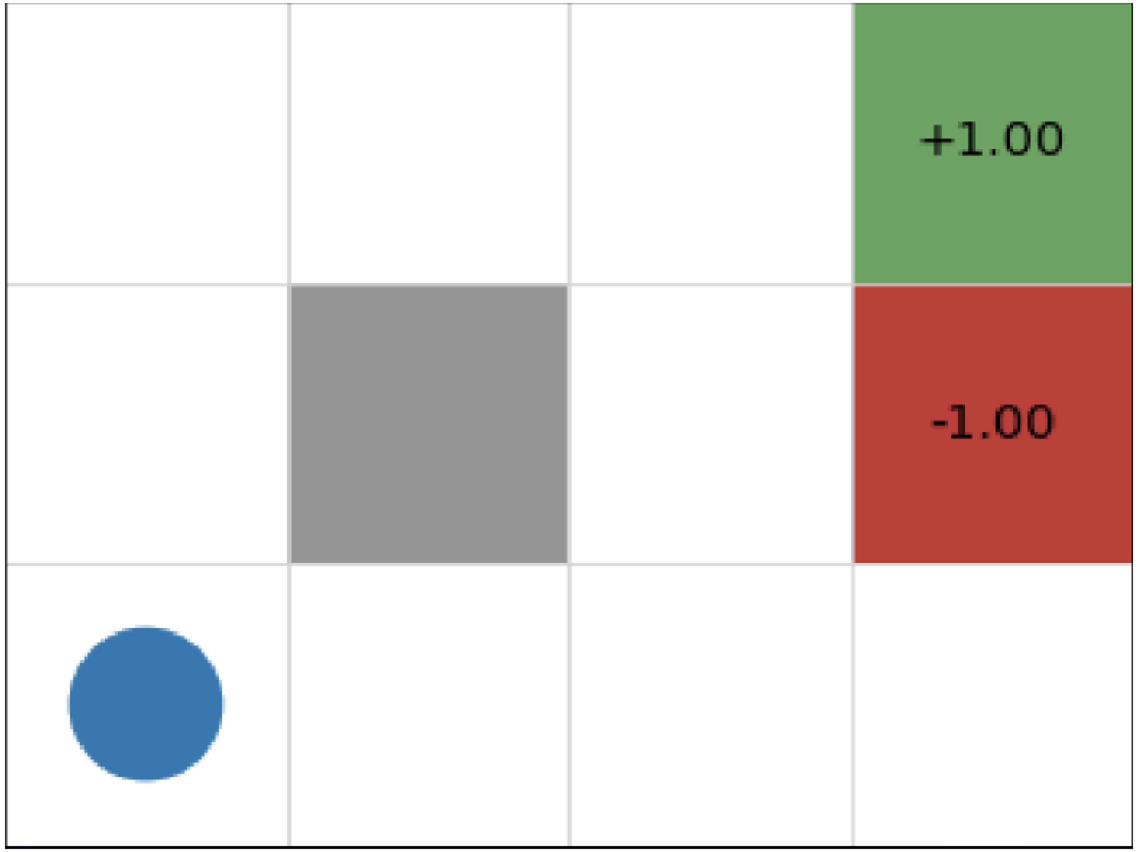
r(s, a, s’) = 0 for any s, s’ belong to S, a belongs to A

Except r((3,2), exit, s_t) = +1

And r((3,1), exit, s_t) = -1

Discount factor $0 < \gamma < 1$

Probability Distribution

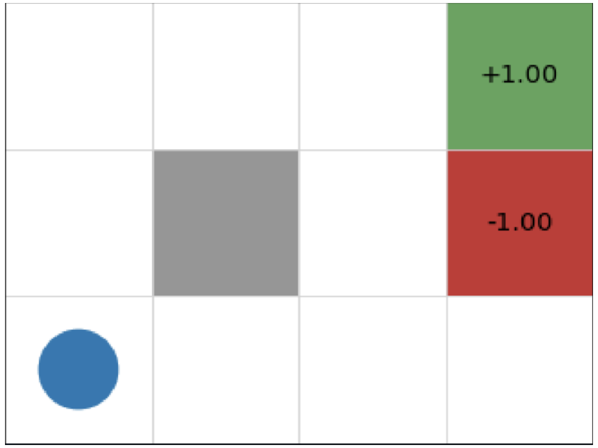


Probability distribution for exit action

- P_exit(s_t | (3, 2)) = 1
- P_exit(s_t | (3, 1)) = 1
- P_exit(s' | any s except above 2 state) = 0

Probability Distribution for North action

P_N((x', y') | (x, y)) =



Common case

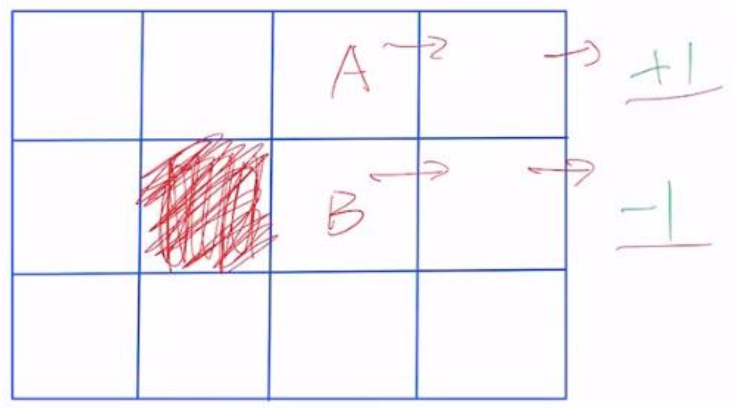
- Successful: If x',y' == x, min(2, y+1) then p = 0.8
- Slip Right: If x',y' == min(3, x+1), y then p = 0.1
- Slip Left: If x',y' == max(0, x-1), y then p = 0.1

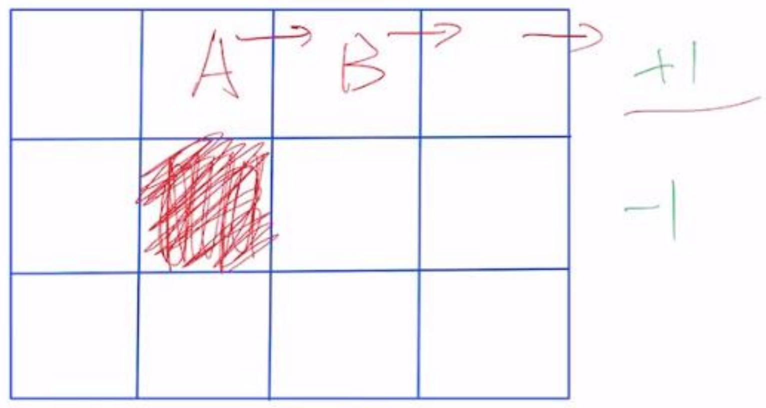
Special Case: Wall

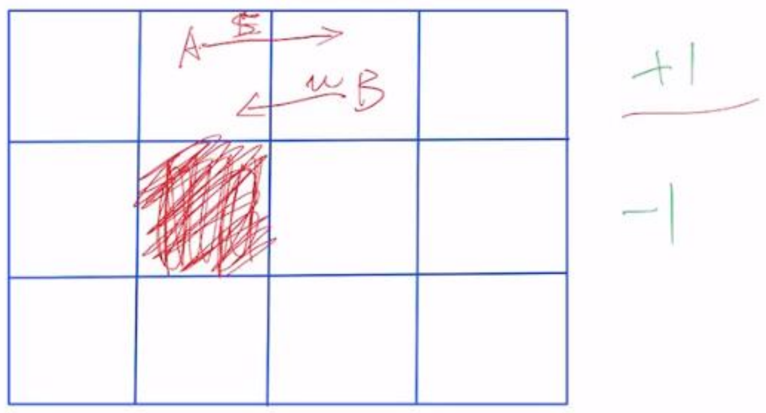
- Do North and Successful: If x, y == x', y' == (1,0) then p = 0.8
- Do North but Slip Left: If x, y == x', y' == (2,1) then p = 0.1
- Do North but Slip Right: If x, y == x', y' == (0,1) then p = 0.1

0.8 succ
0.1 slip left
0.1 slip right

$$V(s) = r + \gamma * V(s')$$

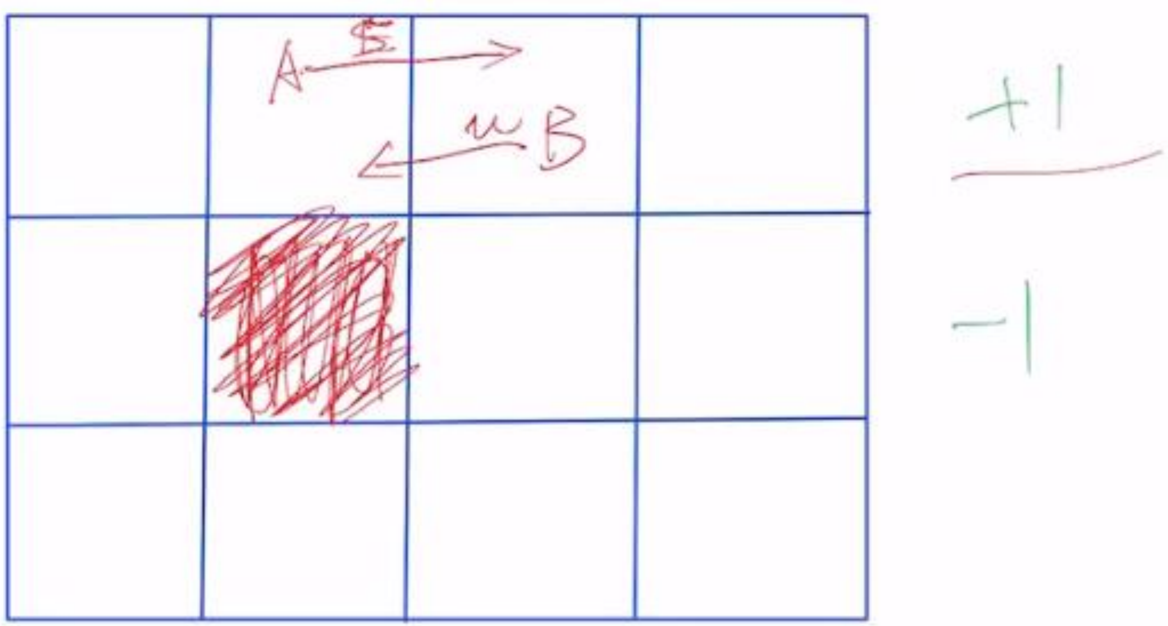






$$Q(s, a) = r(s, a, s') + \gamma * V(s')$$

$$V(s) = \max(Q(s, a)), \text{ where } a \text{ belongs to } A(s)$$



Formula for V(s)

$$V(s) = \max_{a \in A(s)} \sum_{s' \in S} P_a(s'|s)\ [r(s, a, s') + \gamma\ V(s')]$$

Algorithm – Value iteration

Input: MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s,a,s') \rangle$

Output: Value function V

Set V to arbitrary value function; e.g., $V(s) = 0$ for all s

Repeat

$\Delta \leftarrow 0$

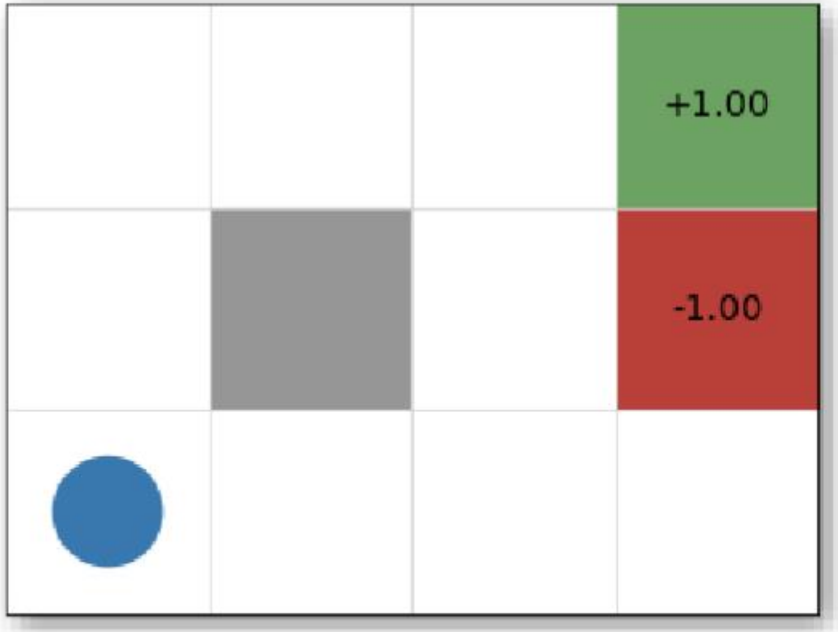
For each $s \in S$

$$\underbrace{V'(s) \leftarrow \max_{a \in A(s)} \sum_{s' \in S} P_a(s' \mid s)\, [r(s,a,s') + \gamma\, V(s')]}_{\text{Bellman equation}}$$

$\Delta \leftarrow \max(\Delta, |V'(s) - V(s)|)$

$V \leftarrow V'$

Until $\Delta \leq \theta$



Formula for V(s)

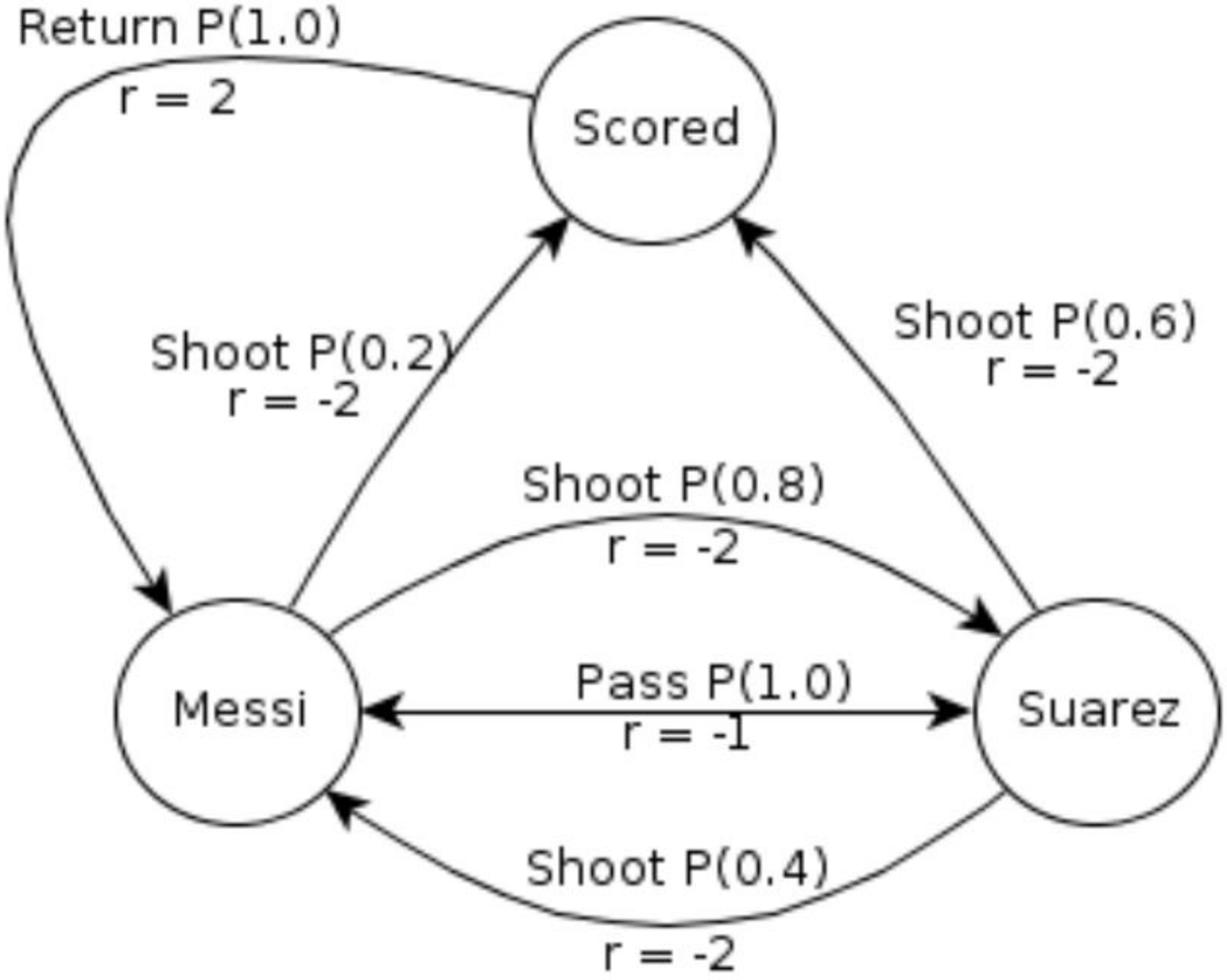
$$V(s) = \max_{a \in A(s)} \sum_{s' \in S} P_a(s'|s)\ [r(s, a, s') + \gamma\ V(s')]$$

Input: MDP $M = \langle S, s_0, A, P_a(s' \mid s), r(s, a, s') \rangle$

Workshop Problems

The football game can be modelled as a discounted-reward MDP with three states: $Messi$, $Suarez$ (denoting who has the ball), and $Scored$ (denoting that a goal has been scored); and the following action descriptions:

- If Messi shoots, he has 0.2 chance of scoring a goal and a 0.8 chance of the ball going to Suarez. Shooting towards the goal incurs a cost of 2 (or a reward of -2).
- If Suarez shoots, he has 0.6 chance of scoring a goal and a 0.4 chance of the ball going to Messi. Shooting towards the goal incurs a cost of 2 (or a reward of -2).
- If either player passes, the ball will reach its intended target with a probability of 1.0. Passing the ball incurs a cost 1 (or a reward of -1).
- If a goal is scored, the only action is to return the ball to Messi, which has a probability of 1.0 and has a reward of 2. Thus the reward for scoring is modelled by giving a reward of 2 when *leaving* the goal state.



Return P(1.0)
r = 2
Scored
Shoot P(0.2)
r = -2
Shoot P(0.6)
r = -2
Shoot P(0.8)
r = -2
Pass P(1.0)
r = -1
Messi
Suarez
Shoot P(0.4)
r = -2

Workshop Problems

Assume that we have calculated the following *non-optimal* value function V for this problem using value iteration with $\gamma = 1.0$, after iteration 2 we arrive at the following:

Iteration		0	1	2	3
V(Messi)	=	0.0	-1.0	-2.0	
V(Suarez)	=	0.0	-1.0	-1.2	
V(Scored)	=	0.0	2.0	1.0	

If Messi has the ball (the system is in the Messi state), what action should we choose to maximise our reward in the next state: pass or shoot? Assume we are using the values for V after three iterations.

Complete the values of these states for iteration 3 using value iteration. Show your working.

Workshop Problems

Assume that we have calculated the following *non-optimal* value function V for this problem using value iteration with $\gamma = 1.0$, after iteration 2 we arrive at the following:

Iteration		0	1	2	3
V(Messi)	=	0.0	-1.0	-2.0	
V(Suarez)	=	0.0	-1.0	-1.2	
V(Scored)	=	0.0	2.0	1.0	

If Messi has the ball (the system is in the Messi state), what action should we choose to maximise our reward in the next state: pass or shoot? Assume we are using the values for V after three iterations.

Complete the values of these states for iteration 3 using value iteration. Show your working.