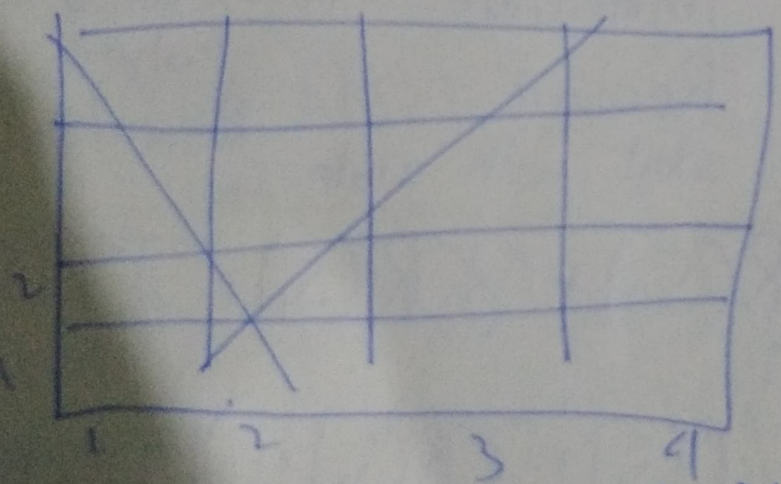
Define V^{π}, V^{*}, π^{*}

V^{π} For a policy π, $V^{\pi}: S \longrightarrow R$

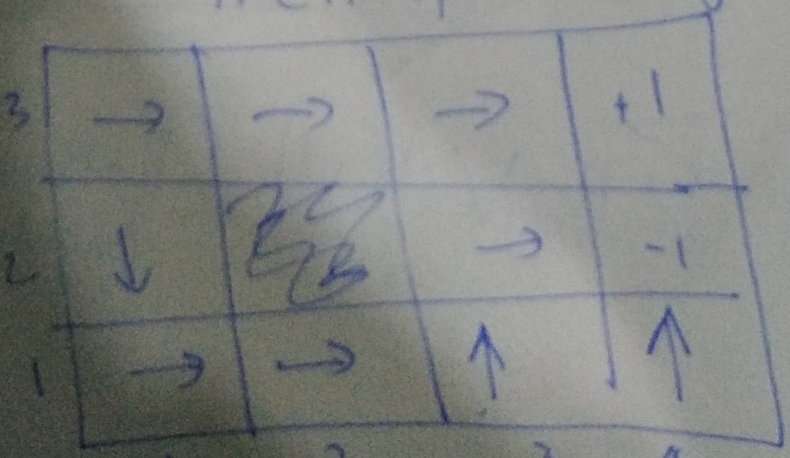
s.t $V^{\pi}(s)$ is the expected total payoff for starting in state s and executing π.

$V^{\pi} = E[R(s_0) + \gamma R(s_1) + \cdots | \pi, s_0 = s]$.

↳ Value function of a policy π.



π (A specific Policy for our MDP).



↳ Not a great Policy.

V^{π} (for the above policy π):-

.52	.73	.77	+1
-.90		-.82	-1
-0.88	-0.87	-0.85	-1.06

Bellman's equation:-

$$V^{\pi}(s) = R(s_0) + \gamma \sum_{s'} P_{s\pi(s)}(s') V^{\pi}(s')$$

$R(s_0)$ ↓ Immediate Reward

$\sum_{s'} P_{s\pi(s)}(s') V^{\pi}(s')$ ↓ Expected Future Reward.

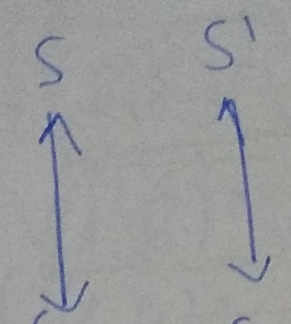
Let say you start with state s_0:-

$$V^{\pi}(s) = E\left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots \mid \pi, s_0\right]$$

$$V^{\pi}(s) = E\left[R(s_0) + \gamma \left(R(s_1) + \gamma R(s_2) + \cdots\right) \mid \pi, s_0 = s_1\right]$$

$$V^{\pi}(s) = R(s_0) + \gamma \sum_{s'} P_{s\pi(s)} V^{\pi}(s') \quad \text{(Bellmans equation)}.$$

(x)



Above equation mapping to our example.

(1) $V^{\pi}(s) = E[R(s) + \gamma V^{\pi}(s')]$.

$s' \sim P_{s\pi(s)}$ (s' distribution).

In state s, take action $a = \pi(s)$.

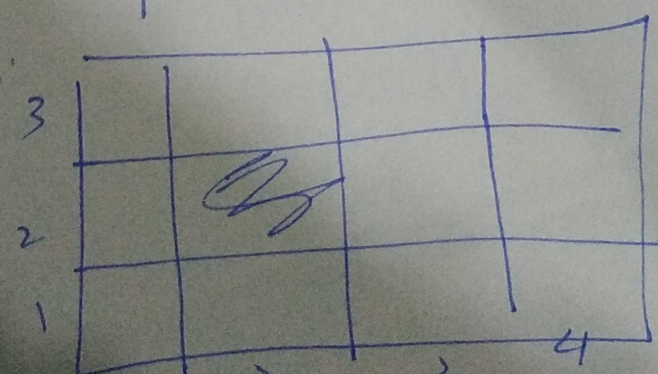
→ Now again taking (1) and taking into s' distribution we have Ballmans equation

$$V^{\pi}(s) = E(R(s)) + E(\gamma V^{\pi}(s'))$$

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} P_{s\pi(s)} V^{\pi}(s')$$

→ Given π, get a linear system of equation in terms of $V^{\pi}(s)$ (By Ballmans equation).

↳ Example:



let say you are in the position (3,1).

$\odot \pi(3,1) = \uparrow$

$$V^{\pi}((3,1)) = R(3,1) + \gamma\left(0.8\,V^{\pi}(3,2) + 0.1\,V^{\pi}(2,1) + 0.1\,V^{\pi}(4,1)\right)$$

→ □ Are unknown variables and if you have 11 states then you 11 equation (Value function for each state) and 11 unknows which can solved through system of 11 linear equation.

→ Given any policy π, you can have 11 unknow variable vector and solve it.

$$\begin{bmatrix} V^{\pi}((1,2)) \\ V^{\pi}((1,2)) \\ \vdots \\ V^{\pi}((4,3)) \end{bmatrix} \rightarrow 11 \text{ unknown} \in R^{11}$$

→ Now lets define V^{*}

V^* is the optimal value function:

$$V^*(s) = \max_{\pi} V^{\pi}(s)$$

↳ Over all the combinatorially large number of possible policies for this MDP, lets take the value of the best possible value for that state.

→ Bellman's equations for $V^*(s)$ (Optimal value):

$$V^*(s) = R(s) + \max_a \gamma \sum_{s'} P_{s'a}(s') V^*(s')$$

↳ Written in neat form:

$$V^*(s) = R(s) + \max_a \gamma \sum_{s'} P_{s'a}(s') V(s')$$

↓ Immediate Reward

↓ Action which maximizes the future expected reward.

→ Now based on $V^{\pi}(s)$ we will Compute:

$$\pi^*(s) = \arg\max_a \sum_{s'} P_{sa} V(s')$$

$\pi^*(s)$ → Optimal Policy

↳ What is the policy that maximizes the future expected value/reward.

→ Practice with confusion notation:

$$V^*(s) = V^{\pi^*}(s) \geq V^{\pi}(s)$$

for every π, s.

→ Strategy to find optimal Policy

1) Find V^*
2) Use argmax equation to find π^*

First Strategy: Value iteration:

→ Algorithm:-

Initialize $V(s) := 0$ for every state s.

For every s update:

$$V(s) := R(s) + \max_a \gamma \sum_{s'} P_{sa}(s') V(s')$$

→ For example in our ~~maldp~~, you will create a 11-dimensional vector ~~ea~~ for in order to represent value function for each state:-

$$\begin{bmatrix} V((1,1)) \\ V((1,2)) \\ \vdots \\ V((9,3)) \end{bmatrix} \rightarrow \in \mathbb{R}^{11}$$

(Initialize to zero).

→ Two ways:-

a) Sychronous update:-

→ Compute $\max_a \gamma \sum_{s'} P_{sa}(s') V(s')$ Simultenously for 11 states and update all 11 states

b): Asychronous update:-

→ Over writing each value one at a time.

* In Sychronous update we make use of Bellman's backup operator:-

$V := BV(V)$

this is the old value vector

Equation previous

$V := B(V)$

New Value Vector for 11 states

Old Vector (Value Vector) for 11 states

* Value iteration causes V to converge to V^* (there is a formal proof) after a specific number of iteration. (100 or few iteration).

* If you run Value iteration on our MDP you end up with:-

3	0.86	0.90	0.93	+1
2	0.81		0.69	-1
1	0.78	0.75	0.71	0.99
	1	2	3	4

→ optimal policy

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

Let say you are in state (3,1) how do you compute optimal policy (using π^* equation) (Future Reward for each action a in state (3,1))

① $W = \sum_{s'} P_{sa} V(s') = 0.8 \times 0.75 + 0.1 \times 0.69 + 0.1 \times 0.71 = 0.740$

② $N = 0.8 \times 0.69 + 0.1 \times 0.75 + 0.1 \times 0.71 = 0.676$

③ $E = 0.8 \times 0.99 + 0.1 \times 0.69 + 0.1 \times 0.75 = 0.53$

↳ Based on this optimal policy it is to go W (Left) (Higher Value)

Second Strategy: Policy iteration:

→ Algorithm:

Initialize π randomly

Repeat {

Set $V := V^{\pi}$ (i.e solve Bellmon equation to get V^{π})

Set $\pi(s) = \arg\max_{a} \sum_{s'} P_{sa}(s') V(s')$

}

As perivously shown that for fixed π Solve for V^{π} through system of linear equations

→ In value iteration our focus is on V^* (coming up with policy at end)

→ In policy iteration our focus is on policy. (coming up with new policy at each iteration)

→ Pros & Cons: (VI & PI)

→ Policy iteration requires solving system of linear equation which become quite expensive in large state space so in those cases Value iteration is more feasible.

→ Value iteration only come closes to V^* (but is never exactly whereas policy iteration solves for exact V^*

→ What if don't know P_{sa}?

→ In practical cases you don't know the state transition probabilities

→ In many MDP you need to estimate $P_{sa}(s')$ from data

→ $P_{sa}(s') = \frac{\text{\# of times agent lands in state } s' \text{ given state } s \text{ \& action } a}{\text{\# of time agent is in state } s \text{ and execute action } a.}$

$= \frac{\text{\# of time took action "a" in state } s \text{ and got to } s'}{\text{\# of times ~~at~~ took action } a \text{ in state } s.}$

→ or $\frac{1}{|S|}$ if above ~~ab~~ is $\frac{0}{0}$

↳ total number of states.

→ Putting it together:

(See on next page) !!!

Repeat {

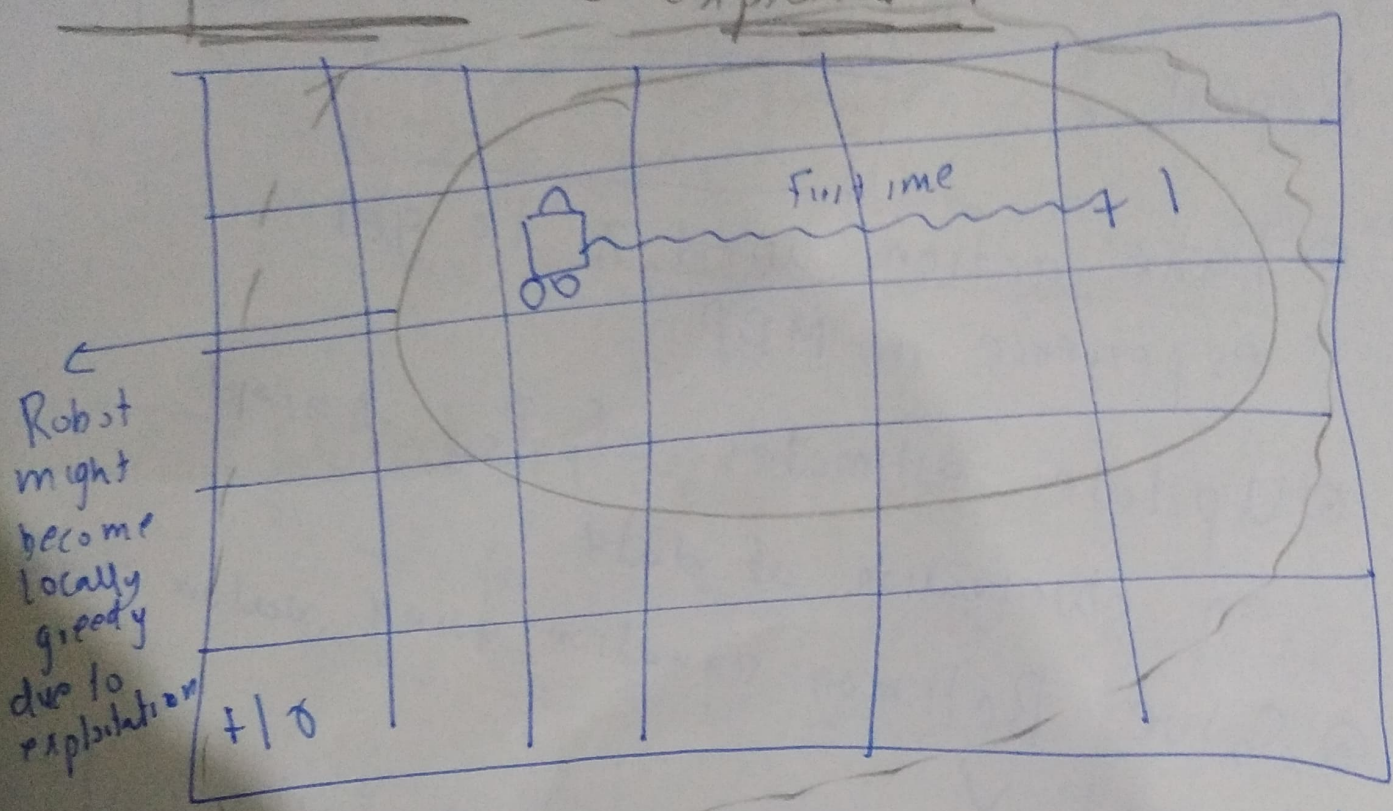
ε-greedy
0.9 chance w.r.t π
0.1 chance randomly

- Take action w.r.t π to get experience in MDP
- Update estimates of P_{sa} based on observation of data
- Solve Bellman equation using value iter to get V
- Update $\pi(s) = \arg\max_a \sum_{s'} P_{sa}(s') V(s')$

}

→ Sometimes Reward function maybe unknown (May need to design it)
  - ↳ Depends on action sometime
  - ↳ Sometime random function of environment).

⊛ Exploration vs Exploitation



↳ Exploration would enable robot to find +10.

↳ How greedy should be the robot to maximize reward (should it explore MDP or only exploit the reward).

→