5. ALGORITHMS FOR MDPs

THINK OF AN ALGORITHM HAVING TWO STEPS

1. A POLICY IMPROVEMENT STEP: eg.

$$\hat{\pi}(x) \in \underset{a}{\text{ARGMAX}} \quad r(x,a) + \beta \mathbb{E}_{x,a}\left[R(\hat{x}_1, \pi)\right]$$

2. A POLICY EVALUATION STEP: eg.

FIND

$$R(x,\pi) := \mathbb{E}_x\left[\sum_{t=0}^{\infty} \beta^t r(X_t, \pi(X_t))\right]$$

WE COVER TWO ALGORITHMS

1. VALUE ITERATION

2. POLICY ITERATION

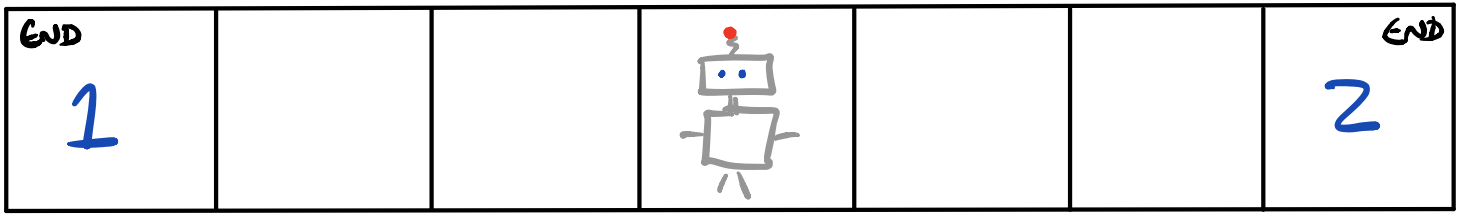
VALUE ITERATION

TAKE $V_0(x) = 0 \quad \forall x$

UNTIL CONVERGENCE DO:

$$V_{t+1}(x) = \max_a \; r(x,a) + \beta \, \mathbb{E}_{x,a}\left[V_t(\hat{x}) \right]$$

VALUE ITERATION EXAMPLE



t=0	0	0	0	0	0
t=1	1	0	0	0	2
t=2	1	1	0	2	2
t=3	1	1	2	2	2
t=4	1	2	2	2	2
t=5	2	2	2	2	2
t=6	2	2	2	2	2

SOME CODE:

```
def Value_Iteration(V,P,r,discount):
    ''' Value Iteration - a numerical solution to a MDP

    # Arguments:
        P - P[a][x][y] gives probablity of x -> y for action a
        r - r[a][x][y] gives reward for x -> y for action a
        V - V[x] gives value for state x
        discount - a float. discount fac tor

    # Returns:
        Value function and policy from **one** value iteration
    '''
    number_of_actions = len(P)
    number_of_states = len(P[0])

    Q = np.zeros((number_of_actions,number_of_states))

    for _ in range(time):
        for a in range(number_of_actions):
            for x in range(number_of_states):
                Q[a][x] = np.dot(P[a][x],r[a][x]+discount*V)
        V_new = np.amax(Q, axis=0)

    pi = np.argmax(Q, axis=0)

    return V_new, pi
```

A RESULT:

Thrm 59. *For positive programming, i.e. where all rewards are positive and the discount factor β belongs to the interval $(0,1]$, then*

$$0 \leq V_s(x) \leq V_{s+1}(x) \nearrow V(x), \quad \text{as} \quad s \to \infty .$$

Here $V(x)$ is the optimal value function.

PROOF IS SAME AS — PROGRAMMING.

POLICY ITERATION:

Def 60 (Policy Iteration). *Given the stationary policy* Π, *we may define a new (improved) stationary policy,* $\mathcal{I}\Pi$, *by choosing for each* x *the action* $\mathcal{I}\Pi(x)$ *that solves the following maximization*

$$\mathcal{I}\Pi(x) \in \underset{a \in \mathcal{A}}{\operatorname{argmax}} \; r(x,a) + \beta \mathbb{E}_{x,a}\left[R(\hat{X}, \Pi)\right]$$

UNTIL CONVERGENCE DO

$$\Pi_{n+1} = \mathcal{I}\Pi_n$$

FINDING $R(x,\pi)$:

WE KNOW FROM MARKOV CHAINS (LECTURE 2) THAT:

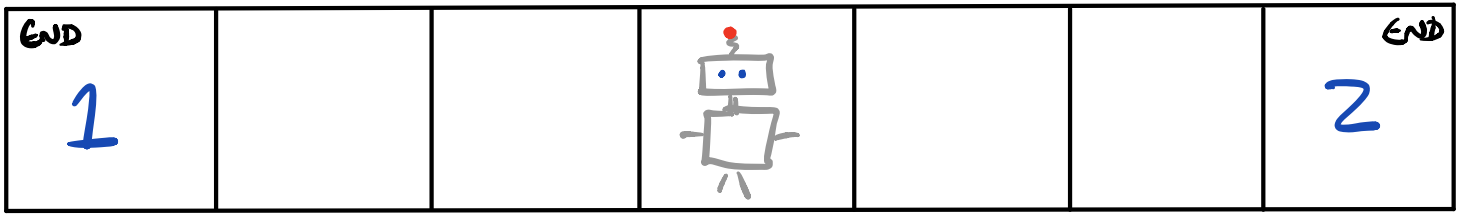
$$R(x) = r(x) + \beta \mathbb{E}_{x\hat{x}} R(\hat{x})$$

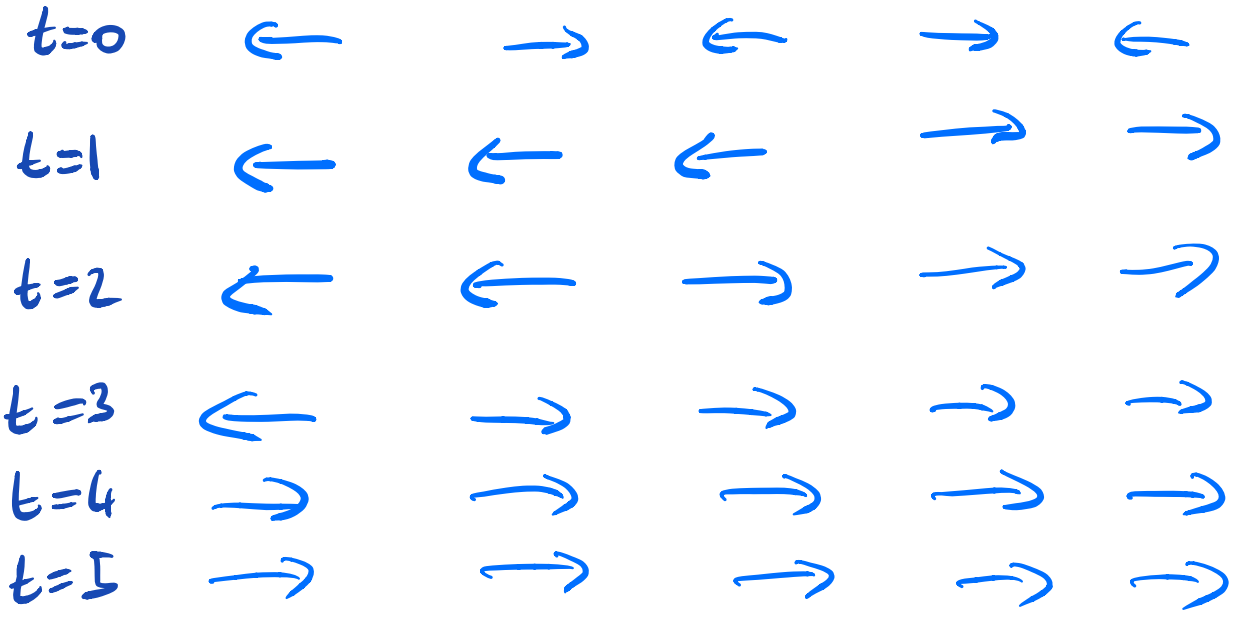
HERE $R(x) = R(x,\pi)$, $r(x) = r(x,\pi)$, $P_{xy} = P_{xy}^{\pi(x)}$.

INTERPRET AS VECTORS & MATRICES.

$$\underline{R} = \underline{r} + \beta P R \iff \underline{R} = (I - \beta R)^{-1} \underline{r}$$

POLICY ITERATION EXAMPLE





SOME CODE:

```
def Policy_Iteration(pi,P,r,discount):
    ''' Policy Iteration - a numerical solution to a MDP
    # Arguments:
        P - P[a][x][y] gives probablity of x -> y for action a
        r - r[a][x][y] gives reward for x -> y for action a
        pi - pi[x] gives action for state x
        discount - disount factor

    # Returns:
        policy from **one** policy iteration
        value function of input policy
    '''

    # Collate array of states and actions
    number_of_actions, number_of_states = len(P), len(P[0])
    Actions, States = np.arange(number_of_actions), np.arange(
    number_of_states)

    # Get transitions and rewards of policy pi
    P_pi = np.array([P[pi[x]][x] for x in States ])
    r_pi = np.array([r[pi[x]][x] for x in States])
    Er_pi = [ np.dot(P_pi[x],r_pi[x]) for x in States]

    # Calculate Value of pi
    I = np.identity(number_of_states)
    A = I - discount * P_pi
    R_pi = np.linalg.solve(A, Er_pi)

    # Calculate Q_factors of pi
    Q = np.zeros((number_of_actions,number_of_states))
    for a in range(number_of_actions):
        for x in range(number_of_states):
            Q[a][x] = np.dot(P[a][x],r[a][x]+discount*R_pi)

    # policy iteration update
    pi_new = np.argmax(Q, axis=0)

    return pi_new, R_pi
```

IS JUST MATRIX ALGEBRA

A RESULT:

Thrm 61. *Under Policy Iteration*

$$R(x, \Pi_{n+1}) \geq R(x, \Pi_n)$$

and, for bounded programming,

$$R(x, \Pi_n) \nearrow V(x) \qquad \text{as} \quad n \to \infty$$

PROOF:

RECALL: FROM MARKOV CHAINS

$$R(x) = \beta(PR)(x) + r(x), \qquad x \in \mathcal{X}.$$

Moreover, if function $\tilde{R} : \mathcal{X} \to \mathbb{R}_+$ *satisfies*

$$\tilde{R}(x) \geq \beta(P\tilde{R})(x) + r(x), \qquad x \in \mathcal{X}.$$

then $\tilde{R}(x) \geq R(x)$, $x \in \mathcal{X}$.

NOTICE π (CURRENT) FOR $\hat{\pi}$ (NEXT)

$$R(x,\pi) = r(x,\pi(x)) + \beta \mathbb{E}_{x,\pi}[R(\hat{x},\pi)] \overset{\text{BY DEF}}{\leq} r(x,\hat{\pi}) + \beta \mathbb{E}_{x,\hat{\pi}}[R(\hat{x},\hat{\pi})]$$

$\therefore R(x,\pi) \leq R(x,\hat{\pi})$ (+) ✓ IS INCREASING.

PROOF (CONT)

FURTHER NOTICE $\forall a$

$$r(x,a) + \beta \mathbb{E}_{x,a}[R(\hat{x},\pi)]$$

$$\leq r(x,\mathcal{Q}_\pi(x)) + \beta \mathbb{E}_{x,\mathcal{Q}_\pi(x)}[R(\hat{x},\pi)]$$

$$\leq r(x,\mathcal{Q}_\pi(x)) + \beta \mathbb{E}_{x,\mathcal{Q}_\pi(x)}[R(\hat{x},\mathcal{Q}_\pi)]$$

$$= R(x,\mathcal{Q}_\pi) \qquad (\dagger)$$

← BY DEF OF $\mathcal{Q}_\pi$

↖ BY $(\dagger)$

To prove convergence to optimal policy

Let

$$M_t = \sum_{s=0}^{t-1} \beta^s r(x, \pi^*(x)) + \beta^t R(x, \pi_{T-t})$$

- π^*: expectation under optimal policy
- $\sum_{s=0}^{t-1} \beta^s r(x, \pi^*(x))$: do optimal policy
- $\beta^t R(x, \pi_{T-t})$: then do π_{T-t}

$$\mathbb{E}^*[M_{t+1} - M_t \mid \mathcal{F}_t] = \beta^t \mathbb{E}^x[\beta R(x_{t+1}, \pi_{T-t-1}) + r(x_t, \pi^*) - R(x, \pi_{T-t})]$$

(By (H))

≤ 0

Because policy π_{T-t} is better $\forall$ actions wrt $R(\cdot, \pi_{T-t-1})$

$\therefore$ M_t is super M_0.

$\therefore$

$$R(x, \pi_t) = \mathbb{E}[M_0] \geq \mathbb{E}[M_T]$$

$$\geq R_T(x, \pi^*) \to R(x, \pi^*)$$

□

LINEAR PROGRAMMING APPROACH:

WE CAN RECAST AS A LINEAR OPTIMIZATION HERE THERE ARE LOTS OF ALGORITHMS:

- SIMPLEX ALGORITHM
- PROJECTED GRADIENT DESCENT
- INTERIOR POINT METHODS
- ...

- WE WANT TO FIND

$$V(x) = \max_{a \in A_x} \; r(x,a) + \beta \mathbb{E}_{\hat{x},\mu}\left[V(\hat{x})\right]$$

- i.e. $(V(x) : x \in \mathcal{X})$ IS THE SMALLEST FUNCTION/VECTOR SUCH THAT

$$V(x) \geq r(x,a) + \beta \sum_{\hat{x}} P(\hat{x}|x,a) V(\hat{x})$$

- SO SOLVE

$\alpha(x)$ — ANY POSITIVE NUMBER.

$$\min_{V \in \mathbb{R}^{\mathcal{X}}} \; \sum_x \alpha(x) V(x)$$

$$\text{s.t.} \quad V(x) \geq r(x,a) + \beta \sum_{\hat{x}} P(\hat{x}|x,a) V(\hat{x}) \quad \forall x, a \qquad \text{(PDP)}$$

LINEAR PROGRAMMING (DUAL)

THE DUAL OF THE LINEAR PROGRAM (PDP) IS

$$\max_{\eta \geq 0} \sum_{x} \eta(x,a)\, r(x)$$

$$\text{s.t.} \sum_{\hat{a}} \eta(\hat{x},\hat{a}) = \zeta(\hat{x}) + \beta \sum_{x,a} \eta(x,a)\, P(\hat{x}|x,a) \quad \forall \hat{x}$$

HERE $\eta(\hat{x},\hat{a})$ CAN BE INTERPRETED

AS THE OCCUPANCY MEASURE OF THE DYNAMIC PROGRAM

PROOF: THE LAGRANGIAN IS

$$L(x;\eta) = \sum_x \xi(x) v(x) - \sum_{x,a} \eta(x,a)\left(v(x) - r(x,a) - \beta \sum_{\hat{x}} P(\hat{x}|x,a) v(\hat{x})\right)$$

≥ 0 FOR COMPLEMENTARY SLACKNESS

$$= \sum_x v(x)\left[\xi(x) - \sum_a \eta(x,a) + \beta \sum_{\hat{x},a} \eta(\hat{x},a) P(x|\hat{x},a)\right] + \sum_{x,a} \eta(x,a) r(x,a)$$

THIS $=0$ FOR A WELL DEFINED LAGRANGIAN

$\therefore$ THE DUAL IS

$$\text{MAX} \sum_{x,a} \eta(x,a) r(x,a)$$

$$\text{s.t.} \quad \xi(x) + \beta \sum_{\hat{x},\hat{a}} \eta(\hat{x},a) P(x|\hat{x},a) = \sum_a \eta(x,a)$$

∎