5. ALGORITHMS FOR MDPs

THINK OF AN ALGORITHM HAVING TWO STEPS

1. A POLICY IMPROVEMENT STEP: eg.

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

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:

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

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*

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

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

PROOF IS SAME AS -ve 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*

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

UNTIL CONVERGENCE DO

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

FINDING $R(x,\pi)$:

WE KNOW FROM MARKOV CHAINS (LECTURE 2) THAT:

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

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

INTEPRET AS VECTORS & MATRICES.

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

IS JUST MATRIX ALGEBRA

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
```

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*

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

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

CURRENT

NEXT

NOTICE π FOR $\hat{\pi}$

BY DEF

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

$$\therefore \quad R(x,\pi) \leq R(x,\hat{\pi})$$

✓ IS INCREASING.

TO PROVE CONVERGENCE TO OPTIMAL POLICY

LET

$$M_t = \underbrace{\sum_{s=0}^{t-1} \beta^s r(x, \pi^*(x))}_{\text{DO OPTIMAL POLICY}} + \beta^t \underbrace{R(x, \pi_{T-t})}_{\text{THEN DO } \pi_{T-t}}$$

EXPECTATION UNDER OPTIMAL POLICY

$$\mathbb{E}^*[M_{t+1} - M_t \mid \mathcal{F}_t] = \beta^t \mathbb{E}^*\Big[\underbrace{\beta R(x_{t+1}, \pi_{T-t-1}) + r(x_t, \pi^*) - R(x, \pi_{T-t})}_{\leq 0} \Big| \mathcal{F}_t\Big]$$

BECAUSE POLICY π_{T-t} IS BETTER $\forall$ ACTIONS WRT $R(\cdot, \pi_{T-t-1})$

$\therefore$ M_t IS SUPER M_g.

$\therefore$

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

$$\geq R_T(x, \pi^*) \longrightarrow R(x, \pi^*) \quad \square$$