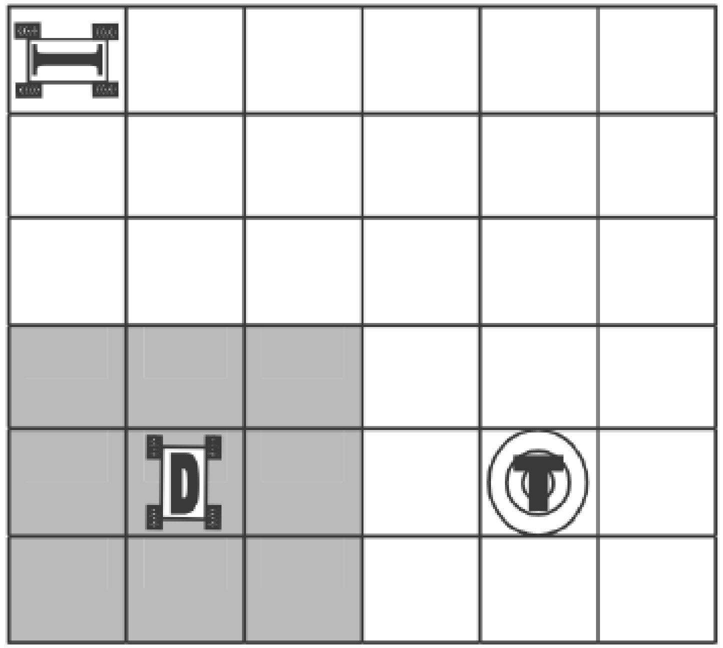
Mechanical & Industrial Engineering
UNIVERSITY OF TORONTO

ML Bootcamp
Minimax Q-learning

Koorosh Moslemi
koorosh.moslemi@mail.utoronto.ca
(slides adapted from MIE567 TA slides by David Molina)

Project 4 Invader-Defender



(a) One possible initial positions of the players when the game starts

Invader-Defender Game: Invader agent (I), Defender agent (D), Goal Territory (T). 6x6 grid.

Main modelling tasks: Markov Game definition.

Main programming tasks:

- Code Invader-Defender Game.
- Implement multi-agent methods:
  - Shapley Value Iteration.
  - Distributed Q-learning and MiniMax Q-learning.
- Compare/visualize the performances and value functions

Main writing tasks:

- Analysis based on the experiments.

Formal Definition of Stochastic Games

SG is defined by a tuple $(n, S, A_1 \dots A_n, T, R_1 \dots R_n)$

n : the number of players,

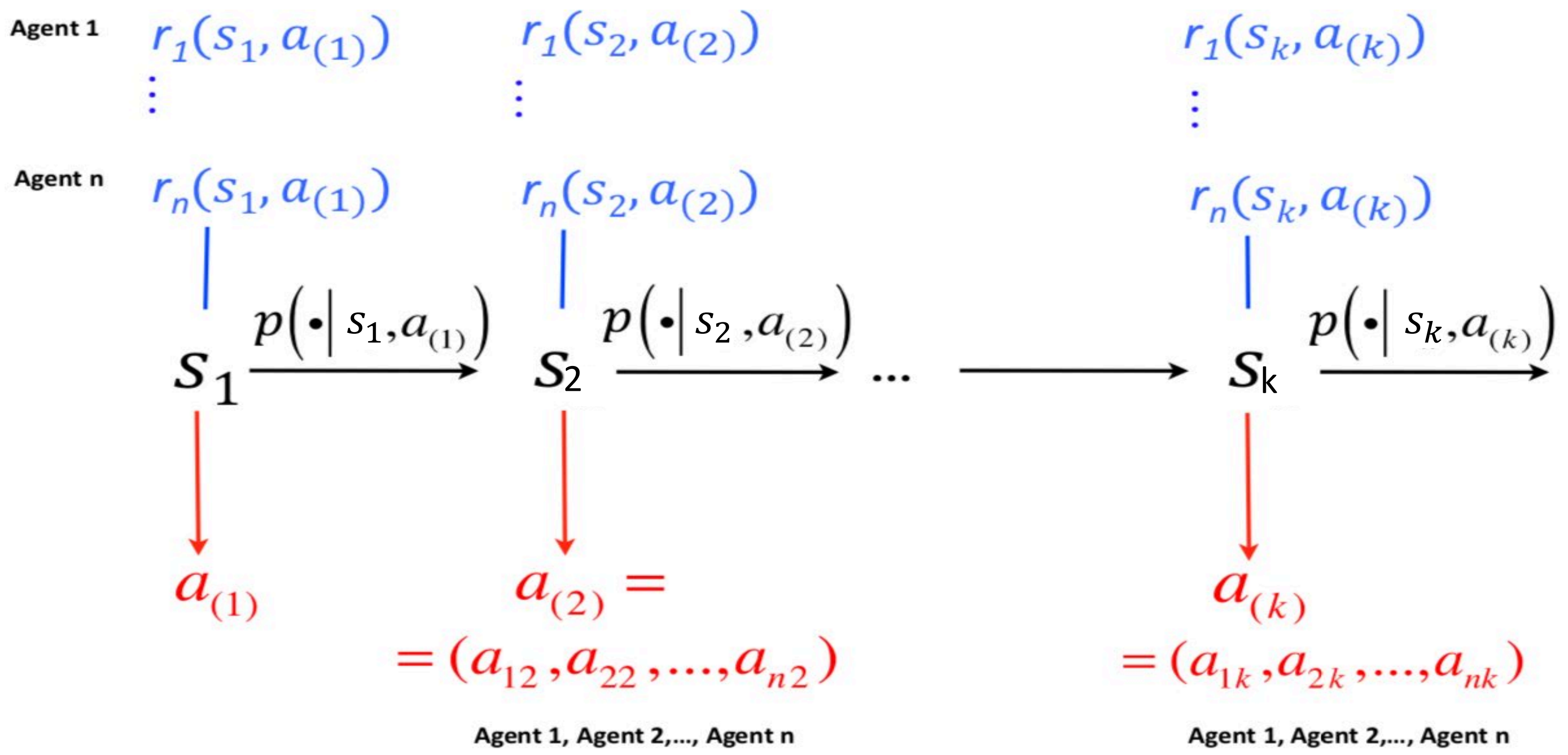
S : the set of states,

A_i : the set of actions available to player i (and A is the joint action space $A_1 \times \cdots \times A_n$),

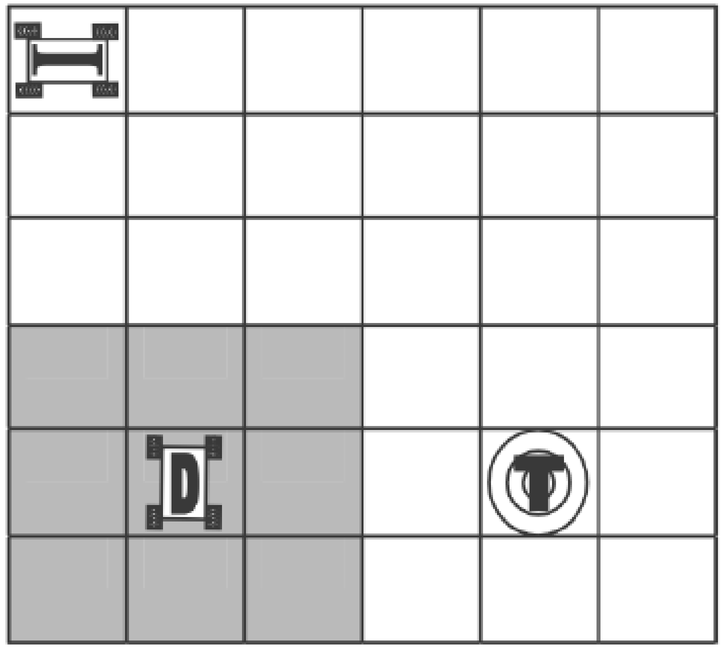
T : the transition function $S \times A \times S \to [0,1]$, and

R_i : the reward function of the i-th agent $S \times A \to R$.

Formal Definition of Stochastic Games


Agent 1
$r_1(s_1, a_{(1)})$
$r_1(s_2, a_{(2)})$
$r_1(s_k, a_{(k)})$
Agent n
$r_n(s_1, a_{(1)})$
$r_n(s_2, a_{(2)})$
$r_n(s_k, a_{(k)})$
s_1
$p(\cdot | s_1, a_{(1)})$
s_2
$p(\cdot | s_2, a_{(2)})$
…
s_k
$p(\cdot | s_k, a_{(k)})$
$a_{(1)}$
$a_{(2)} = (a_{12}, a_{22}, \ldots, a_{n2})$
$a_{(k)} = (a_{1k}, a_{2k}, \ldots, a_{nk})$
Agent 1, Agent 2,…, Agent n
Agent 1, Agent 2,…, Agent n

Preliminary codes - env



(a) One possible initial positions of the players when the game starts

- Time is broken up into discrete epochs.
- The world is a square area that is broken up into a 6-by-6 grid of cells.
- There are two players or agents; both of them can move up, down, left, or right. At each time step, both players take their action simultaneously. Each cannot see the others' action at that time step, but the state of the system and previous actions are fully observable to each. If the chosen action takes an agent outside of the grid, then the agent remains in the current position.
- The invader, marked I, starts in the upper-left corner of the world. The defender, marked D, starts in the bottom-left corner of the world.
- The 9 cells centered around the defender's current position (see Figure 1), is the region where the invader has a 90% chance of beign captured. The game ends whenever the invader is captured by the defender, or the invader enters the fixed territory cell marked T, whichever comes first. The discount factor for future rewards is $\gamma = 0.95$.
- The goal of the defender is to capture the invader as far away as possible from the territory T. The goal of the invader is to move as close as possible to the territory T before being captured.

Preliminary codes - env

1.1 Explain how you would model this problem as a stochastic game: [10 Marks]

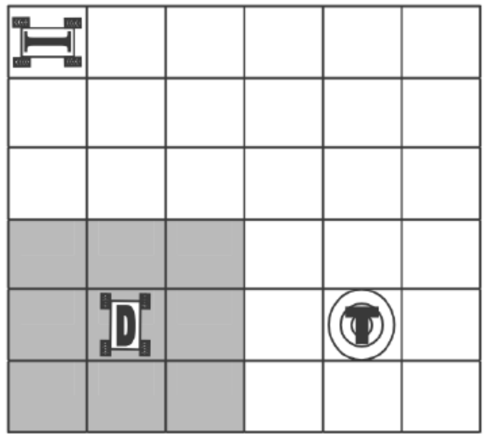
(a) What is the state space, and what are the action spaces for each agent?

(b) Describe the transition probability matrix for this stochastic game.

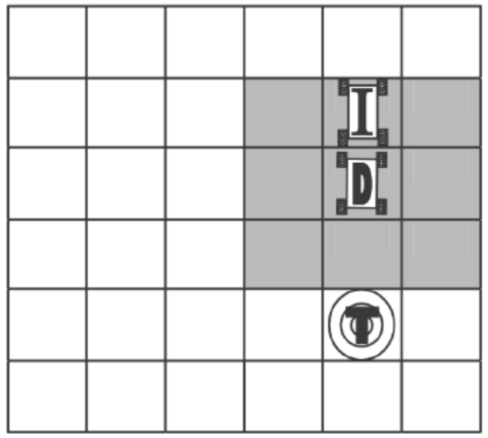
(c) Which states, if any, are terminal states?

(d) Derive one suitable reward function that allows the defender to achieve the goal for this task. Derive one suitable reward function that allows the invader to achieve the goal for this task. Are these the only possible suitable reward functions? Why or why not? What is the relationship between the defender's and invader's reward functions? Choosing a good reward function is critical to get good results in RL, so please choose carefully.

(e) What kind of stochastic game is this (e.g. what special properties does it possess based on what you have learned about stochastic games)?



(a) One possible initial positions of the players when the game starts



(b) One possible terminal positions of the players when the game ends

Equilibrium-based Q-Learning

- Learning Equation

$$Q_i(s,\mathbf{a}) \longleftarrow Q_i(s,\mathbf{a}) + \alpha\big(R_i(s,\mathbf{a}) + \gamma Value_i(s') - Q_i(s,\mathbf{a})\big),$$

  where

  a is a joint action used by agents from state s,
  $Value_i\ (s)$ is player i's value in the equilibrium of the game in state s
  Such a game is composed of the Q_i-values of agents in state s

- Assume that agent knows actions & rewards of all the others (observation, communication)

- Depending on what type of equilibrium we compute:

Equilibrium-based Q-Learning

```
for all $s$, $\mathbf{a}$ and $j \in Ag$ do Initialize $Q_j(s, \mathbf{a})$ with arbitrary values.
Set $s$ to the current state.
Build the game $G(s)$ based on the values $Q_j$ in $s$, $\forall j \in Ag$.
Choose a policy $\pi_i(s) = Equilibrium_i(G(s))$.
repeat
    Play policy $\pi_i(s)$ with some exploration.
    Observe joint action and put it in $\mathbf{a}$.
    Observe each agent's reward and put them in $\mathbf{r}$.
    Observe new state and put it in $s'$.
    Build the game $G(s')$ based on the values $Q_j$ in $s'$, $\forall j \in Ag$.
    Choose a policy $\pi_i(s') = Equilibrium_i(G(s'))$.
    for each player $j$ do Update value $Q_j(s, \mathbf{a})$ using equation (8.2).
    $s \leftarrow s'$, $t \leftarrow t + 1$.
until $t > T$
for $s \in S$ do
    Build matrix $G(s)$ as previously.
    $\pi_i(s) = Equilibrium_i(G(s))$.
return $\pi_i(s)$ $\forall s$.
```

- In the Minimax-Q algorithm above, Equilibrium (G(s)) is the **minimax** equilibrium
- By replacing the minimax equilibrium by **Nash equilibrium**, the algorithm becomes Nash-Q

2-Persons 0-Sum Game with Multiple Actions

Linear Programming for 2-person 0-sum stage game

Two-person zero-sum game: X, Y, A

A: $m \times n$ matrix.

X = $\{1,2,\ldots,m\}$: strategy set for max player (Player 1)

Y = $\{1,2, \ldots,n\}$: strategy set for min player (Player 2)

A mixed strategy for Player 1 is an m-tuple $p = (p_1, p_2, \ldots, p_m)^T$ of probabilities that add up to 1.

A mixed strategy for Player 2 is an n-tuple $q = (q_1, q_2, \ldots, q_n)^T$ of probabilities that add up to 1.

2-Persons 0-Sum Game with Multiple Actions

Linear Programming for 2-person 0-sum stage game

The sets of mixed strategies of players 1 and 2 will be denoted respectively by X^* and Y^* ,

$$X^* = \left\{p = (p_1, \dots, p_m)^T : p_i \geq 0, \text{for } i = 1, \dots, m \text{ and } \sum p_i = 1\right\}$$

$$Y^* = \left\{q = (q_1, \dots, q_n)^T : q_j \geq 0 \text{, for } j = 1, \dots, n \text{ and } \sum q_j = 1\right\}$$

Note: pure strategies are a special case and all included in X^* and Y^*

$$\sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} q_j \right) p_i = \sum_{i=1}^{m} \sum_{j=1}^{n} p_i a_{ij} q_j = \boldsymbol{p}^T \boldsymbol{A} \boldsymbol{q}.$$

Expected Payoff:

If Player 2 chooses the q-mix strategy $(q_1, \ldots, q_n)^T$, Player 1's expected payoff from action i is

$$\sum_{j=1}^{n} a_{ij} q_j = (\boldsymbol{A}\boldsymbol{q})_i,$$

Likewise, player 2's expected payoff from action j given p-mix $(p_1, \ldots, p_m)^T$ used by Payer 1:

$$\sum_{i=1}^{n} p_i a_{ij} = (\boldsymbol{p}^T\boldsymbol{A})_j,$$

More generally, if Player 1 uses $p \in X^*$ and Player 2 uses $q \in Y^*$, the average payoff to Player 1 becomes

$$\sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} q_j \right) p_i = \sum_{i=1}^{m} \sum_{j=1}^{n} p_i a_{ij} q_j = \boldsymbol{p}^T\boldsymbol{A}\boldsymbol{q}.$$

Best Responses

Suppose that Player 2 uses strategy $q \in Y^*$. Then Player 1 will use $p \in X^*$ that maximizes $p^T Aq$.

That is,

$$\max_{1\le i\le m} \sum_{j=1}^{n} a_{ij} q_j \dashrightarrow \max_{\boldsymbol{p}\in X^*} \boldsymbol{p}^T \boldsymbol{Aq}$$

Any $p \in X^*$ that achieves the maximum of $p^T Aq$ is called a best response against q.

Similarly, if Player I is going to use strategy $p \in X^*$, then Player II will use $q \in Y^*$ that minimizes $p^T Aq$.

Her average payoff would be

$$\min_{1\le j\le n} \sum_{i=1}^{m} p_i a_{ij} \dashrightarrow \min_{\boldsymbol{q}\in Y^*} \boldsymbol{p}^T \boldsymbol{Aq}.$$

Any $q \in Y^*$ that achieves the minimum is called a best response Player 2 against p.

Mathematical Programming Formulation

What to optimize?

- Remember that Max player should maximize the **weakest spot**!
- Player 1 wants to choose $p_1, \dots, p_m$ to **maximize** $\min_{1 \le j \le n} \boldsymbol{p^T A q}$ subject to $p \in X^*$.

The maximization problem of P1 is to choose $p_1, \dots, p_m$ as follows:

$$\text{maximize} \quad \min_{1 \le j \le n} \sum_{i=1}^{m} p_i a_{ij}$$

subject to the constraints

$$p_1 + \cdots + p_m = 1$$

and

$$p_i \ge 0 \quad \text{for } i = 1, \dots, m.$$

Linearization

- P1: maximize v

$$v \le \sum_{i=1}^{m} p_i a_{i1}$$

$$\vdots$$

$$v \le \sum_{i=1}^{m} p_i a_{in}$$

$$p_1 + \cdots + p_m = 1$$

$$p_i \ge 0 \quad \text{for } i = 1, \ldots, m.$$

- P2 minimize w

$$w \ge \sum_{j=1}^{n} a_{1j} q_j$$

$$\vdots$$

$$w \ge \sum_{j=1}^{n} a_{mj} q_j$$

$$q_1 + \cdots + q_n = 1$$

$$q_j \ge 0 \quad \text{for } j = 1, \ldots, n.$$

Scipy linprog

- SciPy linprog function requires the previous linear programming formulation to be written in the following format:

$$
\begin{aligned}
&\min_{x} c^T x \\
\text{s.t.}\,& A_{ub} x \le b_{ub} \\
& A_{eq} x = b_{eq} \\
& x \in [x_{lb}, x_{ub}]
\end{aligned}
$$

$$\Longrightarrow$$

$$
\min -v
$$

$$
\text{s.t.} \quad
\begin{aligned}
& v - \textstyle\sum_i p_i * A_{i1} \le 0 \\
& v - \textstyle\sum_i p_i * A_{i2} \le 0 \\
& v - \textstyle\sum_i p_i * A_{i3} \le 0 \\
& v - \textstyle\sum_i p_i * A_{i4} \le 0 \\
& \textstyle\sum_i p_i = 1 \\
& p_i \ge 0
\end{aligned}
$$

$$
\min w
$$

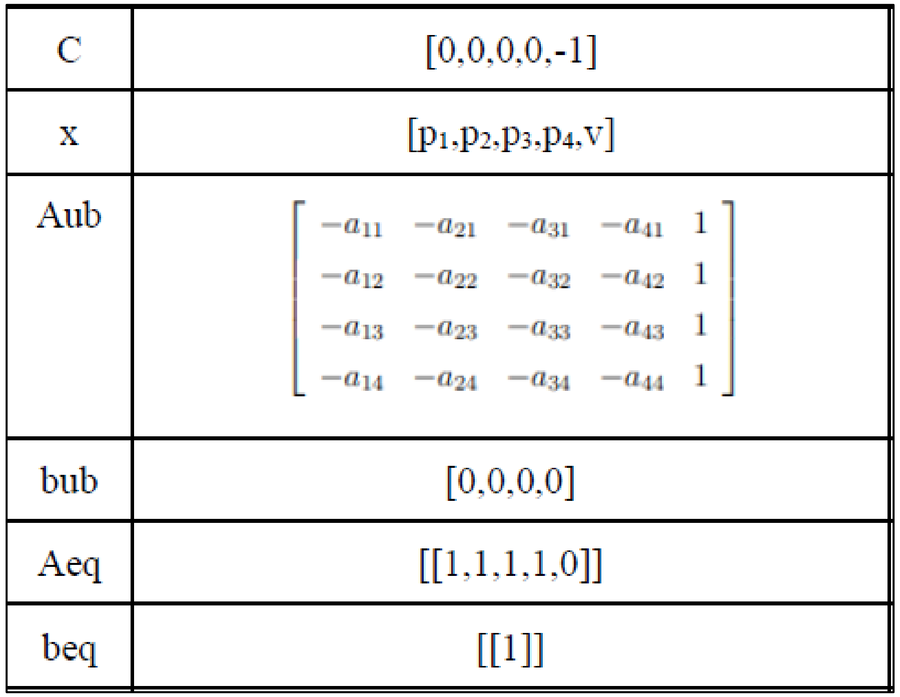
$$
\text{s.t.} \quad
\begin{aligned}
& -w + \textstyle\sum_j q_j * A_{1j} \le 0 \\
& -w + \textstyle\sum_j q_j * A_{2j} \le 0 \\
& -w + \textstyle\sum_j q_j * A_{3j} \le 0 \\
& -w + \textstyle\sum_j q_j * A_{4j} \le 0 \\
& \textstyle\sum_j q_j = 1 \\
& q_j \ge 0
\end{aligned}
$$

Scipy linprog

$$\begin{aligned} &\min_x c^T x \\ \text{s.t.}\ &A_{ub}x \le b_{ub} \\ &A_{eq}x = b_{eq} \\ &x \in [x_{lb}, x_{ub}] \end{aligned}$$

$$\min -v$$

$$\text{s.t.} \quad \begin{aligned} &v - \textstyle\sum_i p_i * A_{i1} \le 0 \\ &v - \textstyle\sum_i p_i * A_{i2} \le 0 \\ &v - \textstyle\sum_i p_i * A_{i3} \le 0 \\ &v - \textstyle\sum_i p_i * A_{i4} \le 0 \\ &\textstyle\sum_i p_i = 1 \\ &p_i \ge 0 \end{aligned}$$



C	[0,0,0,0,-1]
x	$[p_1,p_2,p_3,p_4,v]$
Aub	$\begin{bmatrix} -a_{11} & -a_{21} & -a_{31} & -a_{41} & 1 \\ -a_{12} & -a_{22} & -a_{32} & -a_{42} & 1 \\ -a_{13} & -a_{23} & -a_{33} & -a_{43} & 1 \\ -a_{14} & -a_{24} & -a_{34} & -a_{44} & 1 \end{bmatrix}$
bub	[0,0,0,0]
Aeq	[[1,1,1,1,0]]
beq	[[1]]

https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.linprog.html