Machine, Data and Learning

MDPs and Linear Programming

Linear Programming

- Mathematical programming is used to find the **best or optimal solution** to a problem that requires a decision or set of decisions about how best to use a set of limited resources to achieve a state goal of objectives.
- **Steps involved in mathematical programming**
  - Conversion of stated problem into a mathematical model that abstracts all the essential elements of the problem.
  - Exploration of different solutions of the problem.
  - Finding out the most suitable or optimum solution.
- **Linear programming** requires that all the mathematical functions in the model be **linear functions**.

The Linear Programming Model (1)

Let: $X_1, X_2, X_3, \ldots\ldots\ldots, X_n$ = decision variables

Z = Objective function or linear function

Requirement: Maximization of the linear function Z.

$Z = c_1X_1 + c_2X_2 + c_3X_3 + \ldots\ldots\ldots + c_nX_n$ .....Eq (1)

subject to the following constraints:

$$\begin{array}{c} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \le b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \le b_2 \\ \vdots \qquad\qquad\qquad \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \le b_n \\ \text{all } x_j \ge 0 \end{array}$$

The Linear Programming Model (2)

- The linear programming model can be written in more efficient notation as:

Maximize

$$Z = \sum_{j=1}^{n} c_j x_j$$

subject to:

$$\sum_{j=1}^{n} a_{ij} x_j \leq b_i$$

where

$$i = 1, 2, \ldots, m$$

and

$$x_j \geq 0$$

where

$$j = 1, 2, \ldots, n$$

The decision variables, x_l, x_2, ..., x_n, represent levels of n competing activities.

Examples of LP Problems (1)

1. A Product Mix Problem

- A manufacturer has fixed amounts of different resources such as raw material, labor, and equipment.
- These resources can be combined to produce any one of several different products.
- The quantity of the i^{th} resource required to produce one unit of the j^{th} product is known.
- The decision maker wishes to produce the combination of products that will **maximize total income.**

Examples of LP Problems (2)

2. A Transportation Problem

- A product is to be shipped in the amounts $a_1, a_2, \ldots, a_m$ from m shipping origins and received in amounts $b_1, b_2, \ldots, b_n$ at each of n shipping destinations.
- The cost of shipping a unit from the ith origin to the jth destination is known for all combinations of origins and destinations.
- The problem is to determine the amount to be shipped from each origin to each destination such that the total **cost of transportation is a minimum.**

Examples of LP Problems (3)

3. A Flow Capacity Problem

- One or more commodities (e.g., traffic, water, information, cash, etc.) are flowing from one point to another through a network whose branches have various constraints and flow capacities.
- The direction of flow in each branch and the capacity of each branch are known.
- The problem is to determine the **maximum flow**, or capacity of the network.

Developing LP Model

- Consider the product mix problem

- **Steps Involved:**
  - Determine the objective of the problem and describe it by a criterion function in terms of the decision variables.
  - Find out the constraints.
  - Do the analysis which should lead to the selection of values for the decision variables that optimize the criterion function while satisfying all the constraints imposed on the problem.

Developing LP Model

XYZ Company produces two products: I and II. The raw material requirements, space needed for storage, production rates, and selling prices for these products are given in Table 1.

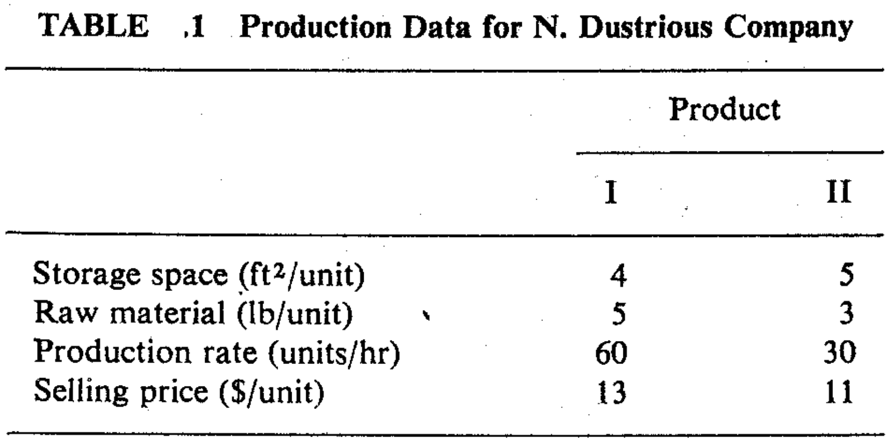


TABLE .1 Production Data for N. Dustrious Company

	Product	
	I	II
Storage space (ft²/unit)	4	5
Raw material (lb/unit)	5	3
Production rate (units/hr)	60	30
Selling price ($/unit)	13	11

The total amount of raw material available per day for both products is 15751b. The total storage space for all products is 1500 ft^2, and a maximum of 7 hours per day can be used for production.

Developing LP Model

All products manufactured are shipped out of the storage area at the end of the day. Therefore, the two products must share the total raw material, storage space, and production time. **The company wants to determine how many units of each product to produce per day to maximize its total income.**

Solution

- The company has decided that it wants to maximize its sale income, which depends on the number of units of product I and II that it produces.
- Therefore, the decision variables, x_1 and x_2 can be the number of units of products I and II, respectively, produced per day.

Developing LP Model

- The object is to maximize the equation:

$$Z = 13x_1 + 11x_2$$

subject to the constraints on storage space, raw materials, and production time.

- Each unit of product I requires 4 ft^2 of storage space and each unit of product II requires 5 ft^2. Thus a total of $4x_1 + 5x_2$ ft^2 of storage space is needed each day. This space must be less than or equal to the available storage space, which is 1500 ft^2. Therefore,

$$4X_1 + 5X_2 \leq 1500$$

- Similarly, each unit of product I and II produced requires 5 and 3 lbs, respectively, of raw material. Hence a total of $5x_l + 3x_2$ Ib of raw material is used.

Developing LP Model

- This must be less than or equal to the total amount of raw material available, which is 1575 Ib. Therefore,

$$5x_1 + 3x_2 \leq 1575$$

- Product I can be produced at the rate of 60 units per hour. Therefore, it must take 1/60 of an hour to produce 1 unit. Similarly, it requires 1/30 of an hour to produce 1 unit of product II. Hence a total of $x_1/60 + x_2/30$ hours is required for the daily production. This quantity must be less than or equal to the total production time available each day. Therefore,

$$x_1 / 60 + x_2 / 30 \leq 7 \text{ or } x_1 + 2x_2 \leq 420$$

- Finally, the company cannot produce a negative quantity of any product, therefore x_1 and x_2 must each be greater than or equal to zero.

Developing LP Model

- **The linear programming model for this example can be summarized as:**

Maximize

$$Z = 13x_1 + 11x_2$$

subject to:

$$\begin{aligned} 4x_1 + 5x_2 &\leq 1500 \\ 5x_1 + 3x_2 &\leq 1575 \\ x_1 + 2x_2 &\leq 420 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned}$$

Linear Programming for MDPs

- Maximize

$\sum_{i \in S} V_i$ (S is the set of states, Vi is value of state)

Such that:

$$V_i <= [\ R(I,A) + \gamma \sum\ P(J|I,A)^* V_j]$$

Linear Programming

- Linear programming polynomial time (Karmarkar, 84)
  - Popular method Dantzig's simplex (1963)
  - Simplex performs well in practice, but could be exponential

- Although polynomial time typically slower than MDP specific algorithms such as value iteration

- Determine policy from V_i using method from previous slides
  - Yields a deterministic policy for MDPs: why?

- Key advantage: Many times it is amenable to modeling specific constraints

A more popular formulation

$$\max \sum_i \sum_a x_{ia} r_{ia}$$

$$\sum_a x_{ja} - \sum_i \sum_a x_{ia} p_{ij}^a =$$

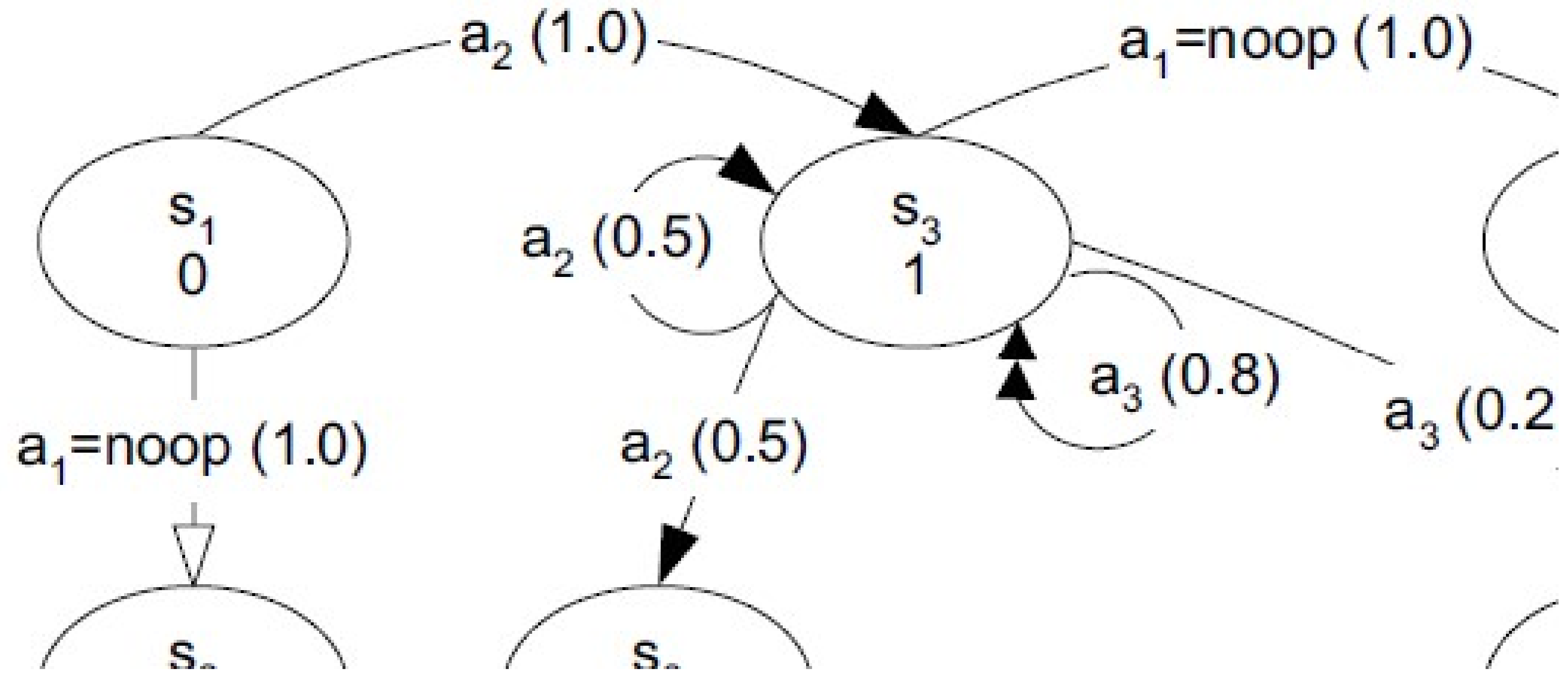
x_{ia}: Expected number of times action a is taken in state i
r_{ia}: Reward for taking action a in state i
p^a_{ij}: Probability of reaching state j when action a is taken in state I
α_{j}: Initial probability of being in state j

Refactoring

$$\max \sum_{i} \sum_{a} x_{ia} r_{ia} \left| \sum_{i} \sum_{a} (\delta_{ij} - p_{ij}^{a}) x_{i} \right.$$

where δ_{ij} is the Kronecker delta, de£ned as $\delta_{ij} = 1$

MDP Example



A simple MDP: $S = \{s_1, \ldots, s_6\}, A = \{a_1 = noop, a_2, a_3\}$

MDP Example

$$\max(\mathbf{rx}) \,\big|\, \mathbf{Ax} = \boldsymbol{\alpha},\ \mathbf{x}$$

$$\mathbf{x} = [(x_{11}, x_{12}), x_{21}, (x_{31}, x_{32}, x_{33}), x_{41}, x_{51}, x_{61}]^T,$$

$$\mathbf{r} = [(0,0), 5, (1,1,1), -10, 50, 60], \quad \boldsymbol{\alpha} = [0.1, 0.1, 0.1, 0.$$

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0.5 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.2 & 0 & 1 & 0 \end{pmatrix}$$

Solution to the LP

$$\mathbf{x} = [(0, 0.1), 0.1, (0, 0.4, 0), 0.1,$$

$$\mathbf{d} = [a_2, a_1, a_2, a_1, a_1, a_1]$$

Solving the LP we get x, which maps to a deterministic uniformly-optimal policy

We get the following output if we change alpha to the following

$$\boldsymbol{\alpha} = [1, 0, 0, 0, 0, 0]$$

$$\mathbf{x} = [(0, 1), 0, (0, 2, 0), 0, 0, 1]$$

$$d_1 = a_2, d_3 = a_2, d_6$$

arbitrary actions for s_2, s_4, and s_5

CMDP

- A key advantage of the formulation is the ability to model constraints
- If we want to say state 1 must have all actions take with equal probability
  – x11 = x12
- Addition of such constraints results in a CMDP (Constrained MDP)
- How do you model action 1 in state 1 must be taken 30% of time ?
  – X11/(x11+x12) = 1/3
  – How can you do this with value or policy iteration ?

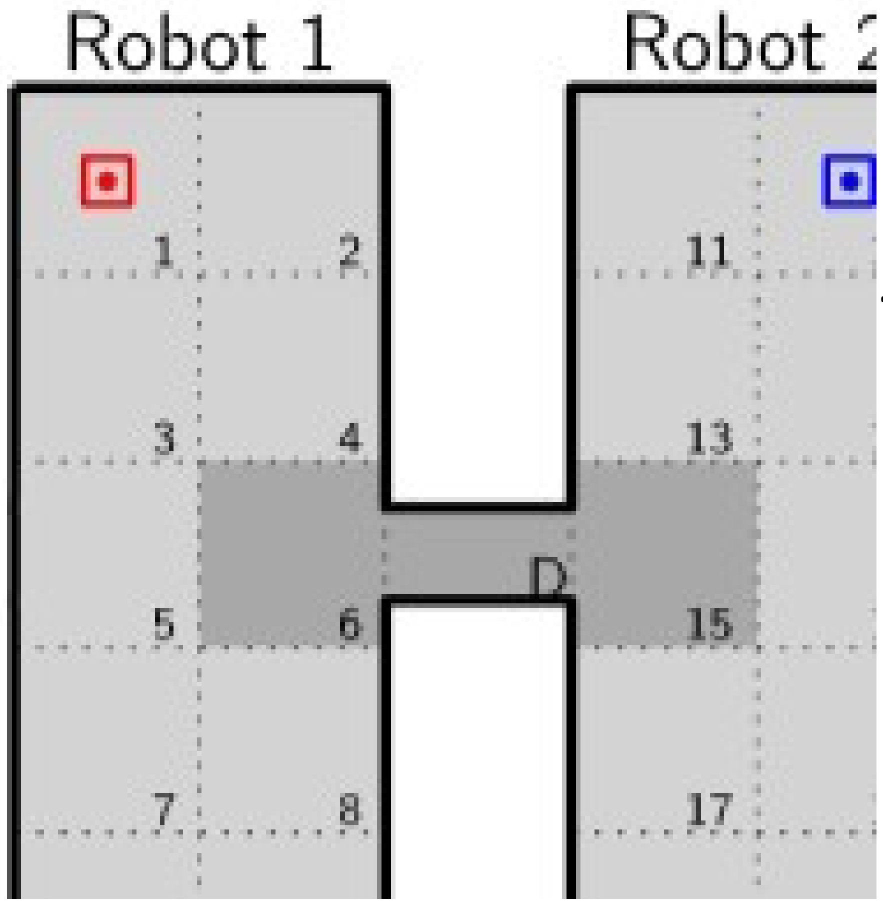
Multi-Agent MDP (MMDP)

- **MMDP:** A direct extension over MDPs for multiple agents
- M = $<S, \{A_i\}_{i \in m}, T, R>$
  - **S** is set of possible world states
  - $\mathbf{\{A_i\}_{i \in m}}$ is set of joint actions, $<a_1, \ldots, a_m>$ where $a_i \in A_i$
  - **T** defines transition probabilities over joint actions
  - **R** is team reward function
- State is fully observable by each agent
- In absence of communication, random policies can lead to mis-coordination
- Ex: In state s, policy a1b2 = .7 and a2b1 = .3 would lead to a1b1 with .21, a1b2 with .49, a2b1 with .09 and a2b2 with .21

Dec-MDP

- In MMDP model agent observes the joint state
- Many times in a joint problem, an agent may observe only his/her local state
- Separating out the policy computation for each agent is not an option since they have joint transitions and rewards
- Dec-MDP – A formal framework to address these issues

Robots coordinating in a hallway with a narrow passage



Can be modeled using Dec-MDP framework

MDP vs. POMDPs

- **MDP:** Agent's percept in any given state identify the state that it is in, e.g., state (4,3) vs (3,3)
  - Given observations, uniquely determine the state
  - Hence, we will not explicitly consider observations, only states

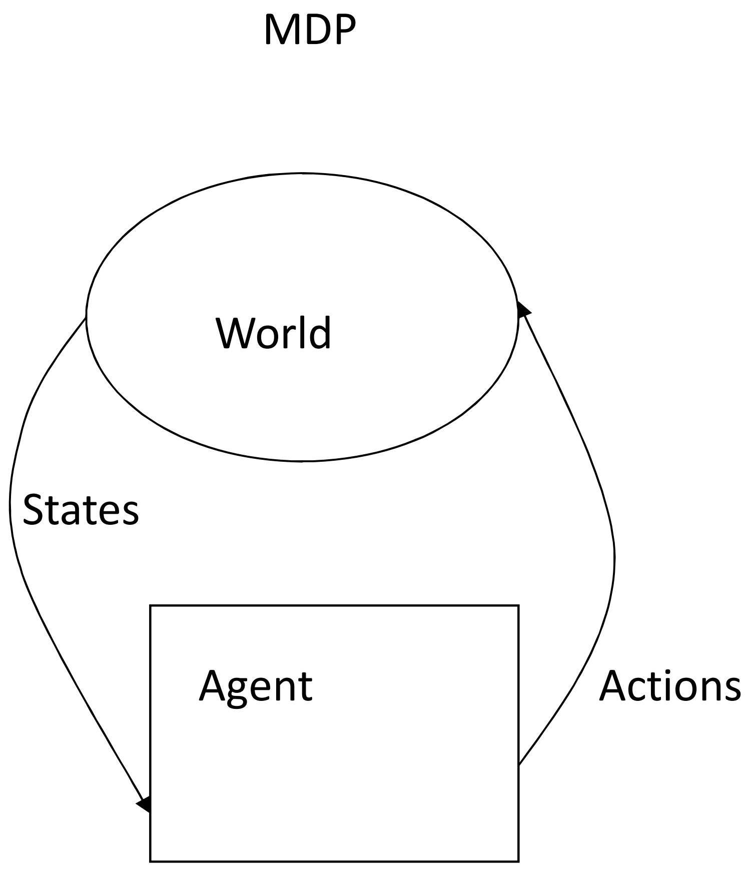
- **POMDP:** Agent's percepts in any given state **DO NOT** identify the state that it is in, e.g., may be (4,3) or (3,3)
  - Given observations, not uniquely determine the state
  - POMDP: Partially observable MDP for inaccessible environments

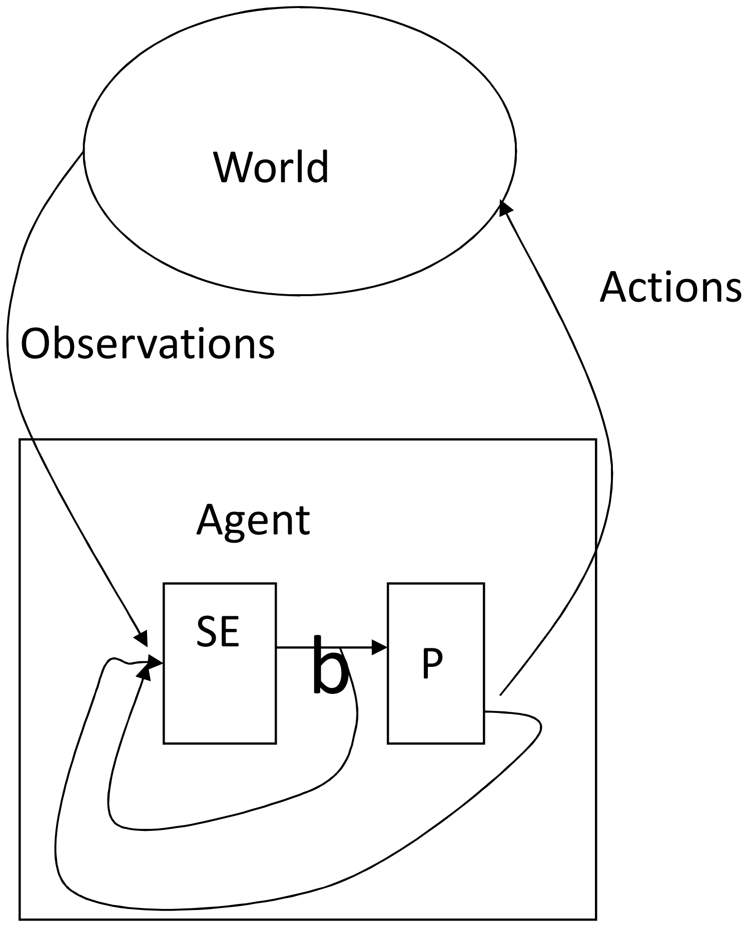
POMDP: Partially Observable Markov Decision Process

- Set of states, **S**
- Set of actions, **A**
- **P** is the table of transition probabilities
- **R(s,a)** reward received for taking action “a” in state “s”
- **Policy** π maps a **state “s”** to an **action “a”**

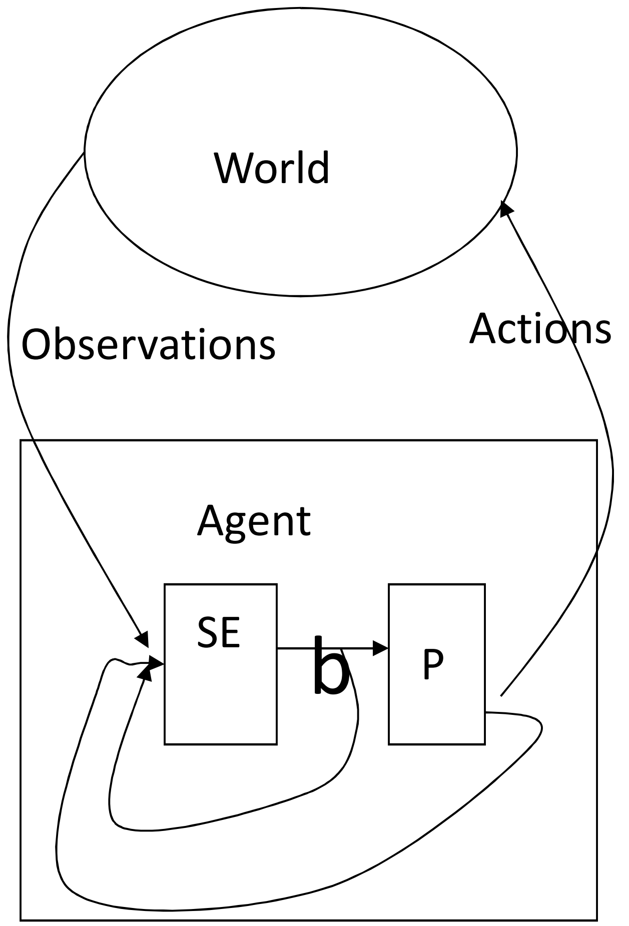
- **PLUS**
  - Finite set Ω of observations
  - Table **O** of observation probabilities where **O(o|a,s’)** is the probability that “o” is observed given that action “a” taken leads to state s’
  - **Policy** maps **histories of observations** to **actions**

MDP vs POMDP


MDP
World
States
Agent
Actions


World
Actions
Observations
Agent
SE
b
P

POMDP



SE: State estimator

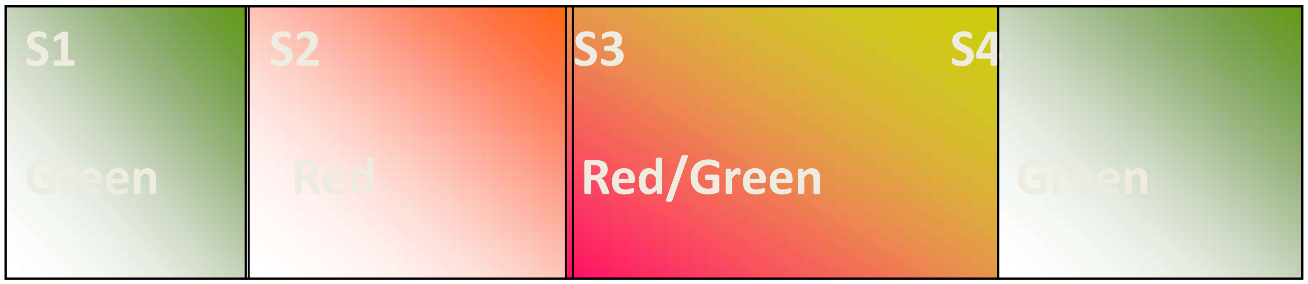
b: Belief state

SE updates the beliefs
based on last observation,
previous belief state
and previous action

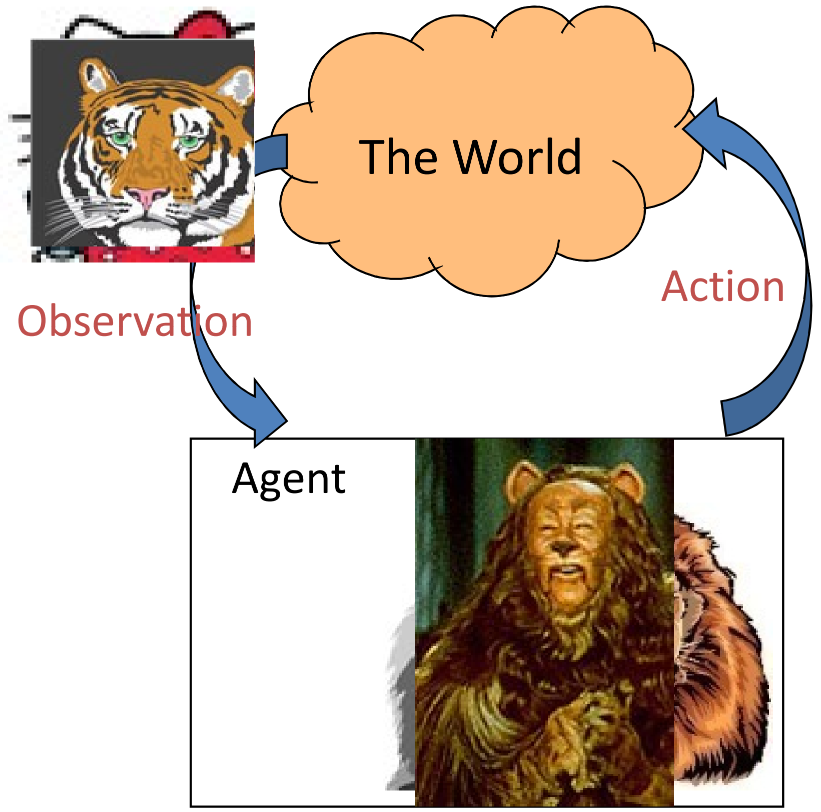
P: Policy is no longer a function of the state,
But of the agent's belief state

POMDP:**<S, A, P, R, Ω, O>**

- **S,** Set of states
- **A,** finite set of actions
- **P** is the table of transition probabilities
- **R(s,a)** reward received for taking action "a" in state "s"
- Finite set Ω of observations, e.g., {*red, green} in example below*
  - Observations hint at state, e.g., *Observe Red room, but not S3*
- Table **O** of observation probabilities
  - ***O(o|a,s')*** *prob "o" observed given action "a" leads to state s'*
  - *P(red | LEFT, S3) = 0.4*



POMDP: Partially Observable Markov Decision Process



- Agent has initial beliefs
- Agent takes an action
- Gets an observation
- Interprets the observation
- Updates beliefs
- Decides on an action
- Repeats

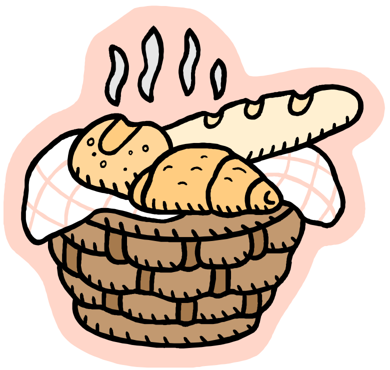
Agent takes optimal action considering world/other agents

Elements: {States, Actions, Transitions, Rewards, Observations }

POMDP: Partially Observable Markov Decision Process

- Underlying dynamics are still **Markovian**: World has **NOT** changed its characteristics, agents sensors have changed
- Observations only hint at what state we are in, but not exactly identify state
- So, somehow agent may need to remember what it observed in the past and what action it took:
  - *If I observed feature "green" in the past, then took action "left" and then observed "red", it must mean that I am either in state S3 (probability of 0.9) or S2 (Prob 0.1) now*
- *Need to maintain beliefs*

Tiger Problem



- Standing in front of two closed doors
- World is in one of two states: tiger is behind left door or right door
- Three actions: Open left door, open right door, listen
    - *Listening is not free, and not accurate (may get wrong info)*

- Reward: Open the wrong door and get eaten by the tiger (large –ve)
  Open the right door and get a prize (small +ve)

Tiger Problem: POMDP Formulation

- Two states: SL and SR
- Three actions: LEFT, RIGHT, LISTEN
- Transition probabilities:

Listen	SL	SR
SL	1.0	0.0
SR	0.0	1.0

Left	SL	SR
SL	0.5	0.5
SR	0.5	0.5

Right	SL	SR
SL	0.5	0.5
SR	0.5	0.5

Tiger Problem: POMDP formulation

- Observations: TL (tiger left) or TR (tiger right)
- Observation probabilities:

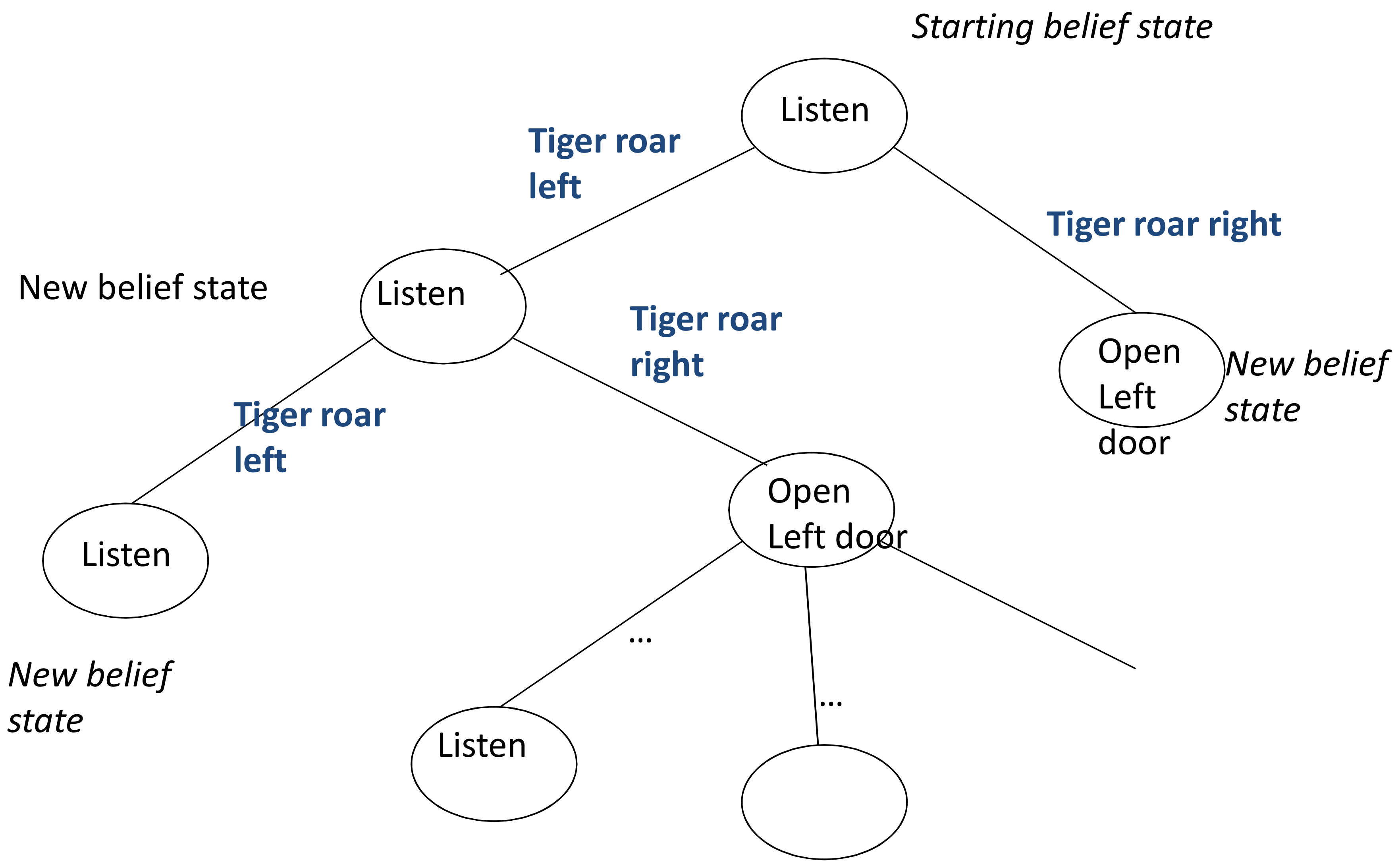
Listen	TL	TR
SL	0.85	0.15
SR	0.15	0.85

Left	TL	TR
SL	0.5	0.5
SR	0.5	0.5

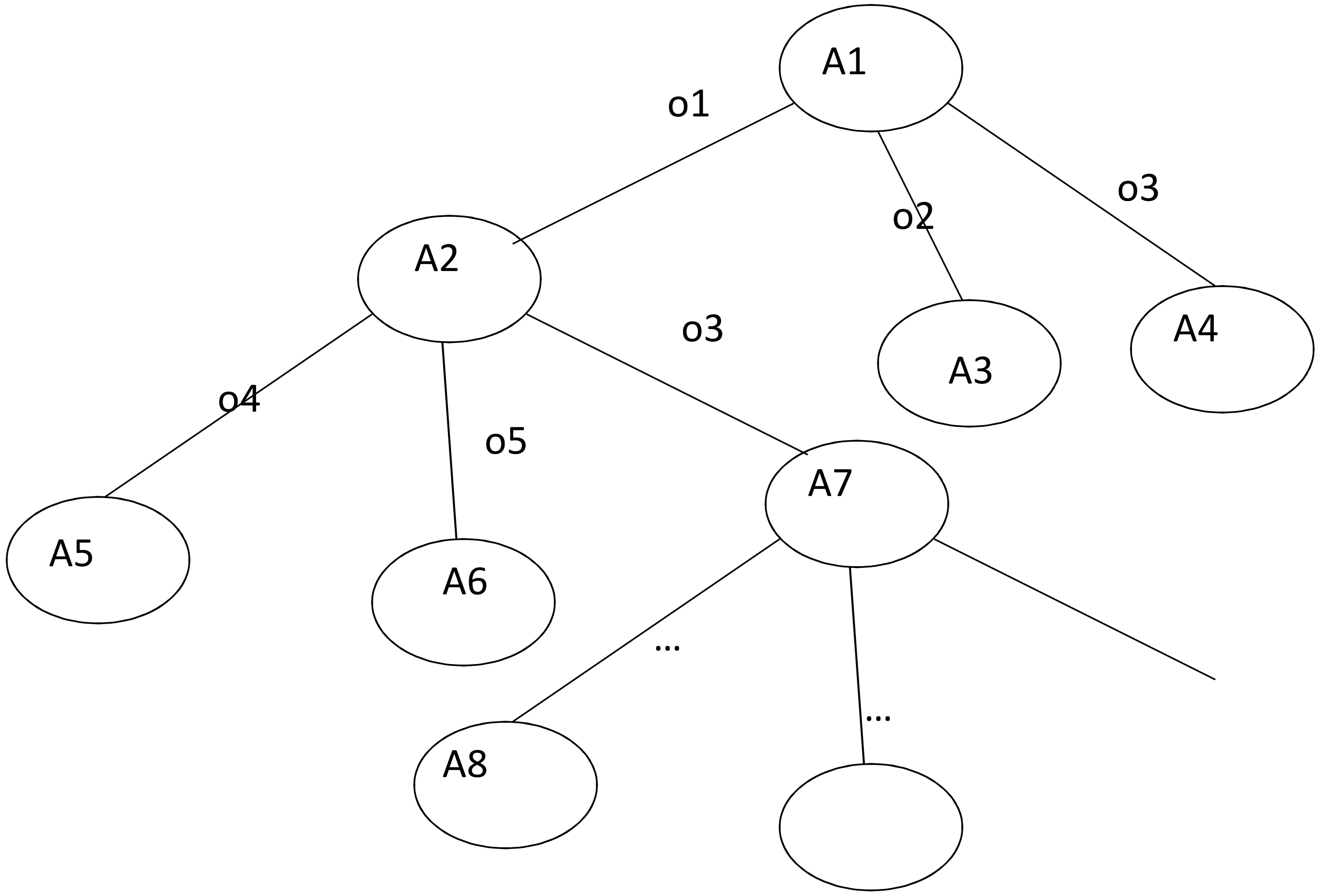
Right	TL	TR
SL	0.5	0.5
SR	0.5	0.5

- Rewards:
  - *R(SL, Listen) = R(SR, Listen) = -1*
  - *R(SL, Left) = R(SR, Right) = -100*
  - *R(SL, Right) = R(SR, Left) = +10*

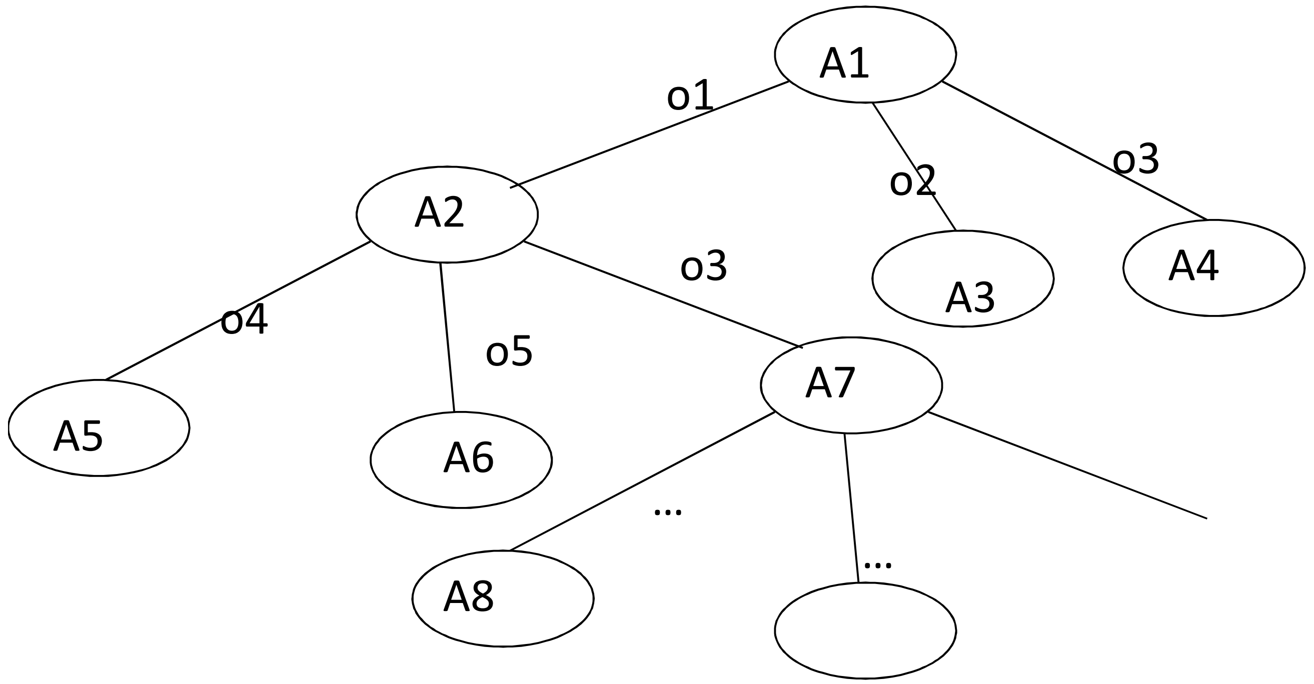
Sample POMDP Policy Tree



POMDP Policy Tree



How Many POMDP policies possible



1

|O|

|O|^2

...

How many policy trees, if |A| actions, |O| observations, T horizon:

- How many nodes in a tree:

$$N = \sum_{i=0}^{T-1} |O|^{i} = (|O|^{T} - 1) / (|O| - 1)$$

How many trees:

$$|A|^{N}$$