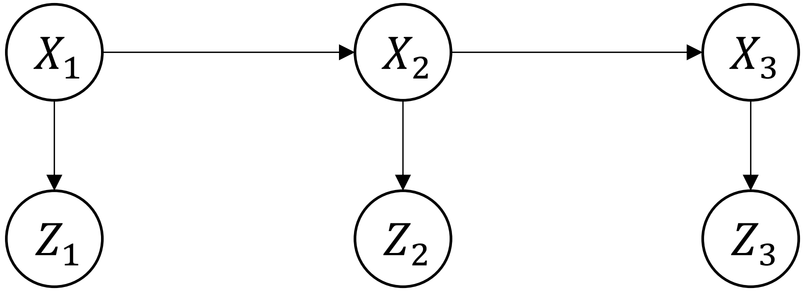
CS 3630!

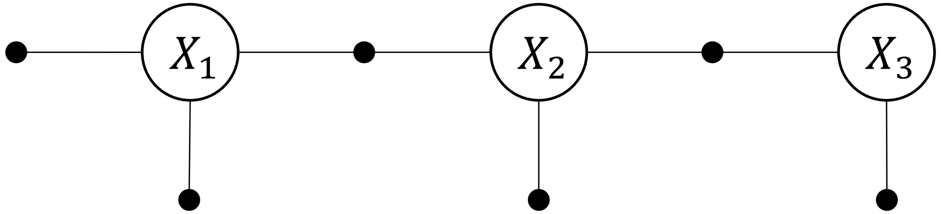
Lecture 9:

A Vacuum Cleaning Robot: MDPs, Sensing, and Perception

Lecture 9 Recap

Factor Graphs





- Measurements are given – get rid of them!

$$P(\mathcal{X}|\mathcal{Z}) \propto P(X_1)L(X_1;z_1)P(X_2|X_1)L(X_2;z_2)P(X_3|X_2)L(X_3;z_3)$$

- This becomes:

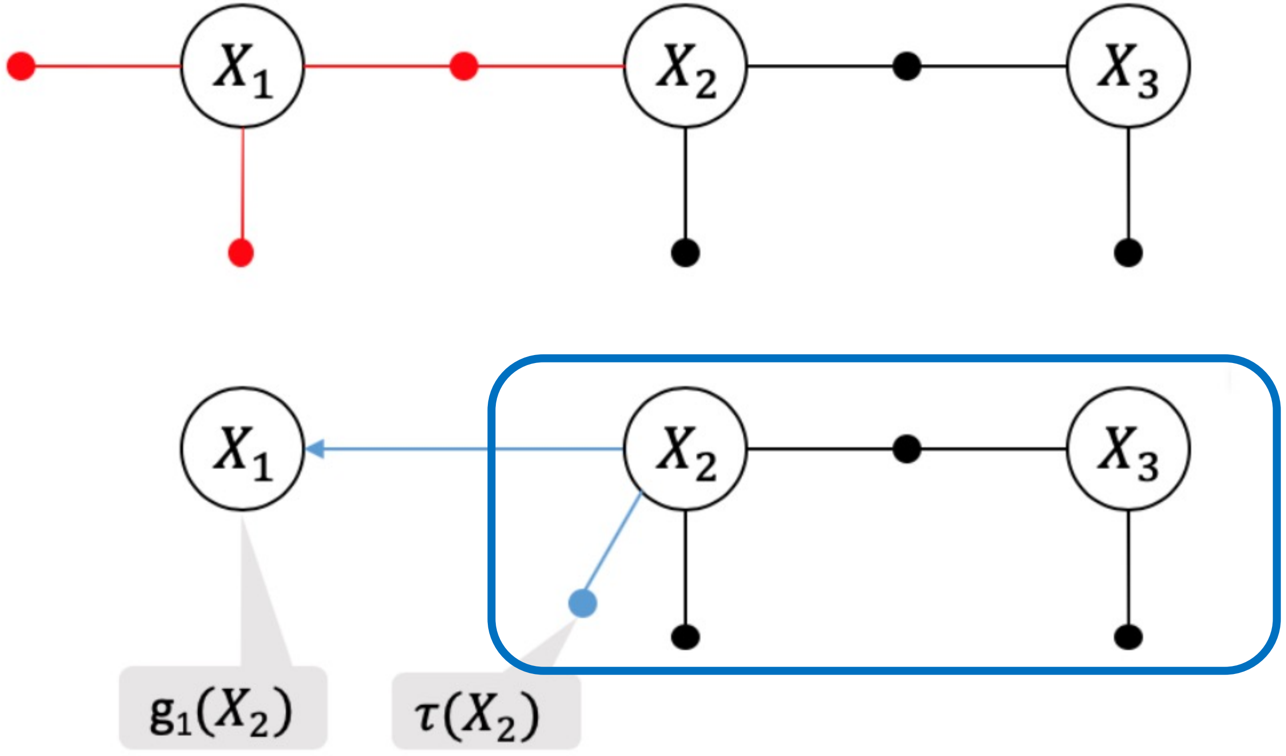
$$\phi(\mathcal{X}) = \phi_1(X_1)\phi_2(X_1)\phi_3(X_1,X_2)\phi_4(X_2)\phi_5(X_2,X_3)\phi_6(X_3)$$

Each factor defines a function ϕ which is a function only of its (non-factor node) neighbors.

MPE via max-product

- Eliminate one variable at a time by forming product, then max:

$$\phi(X_1, X_2) = \phi_1(X_1)\phi_2(X_1)\phi_3(X_1, X_2).$$

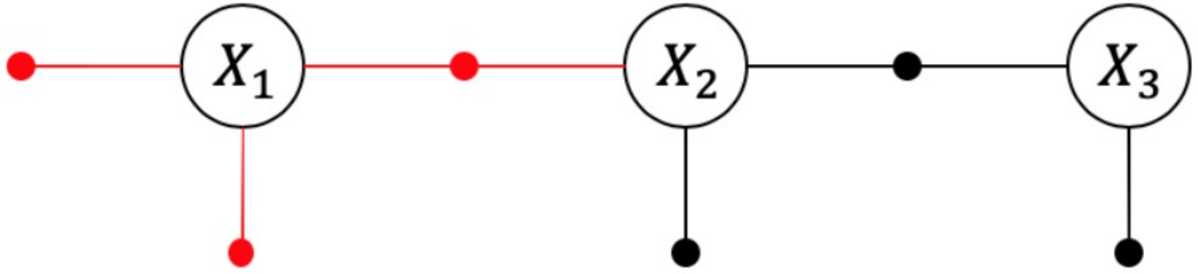


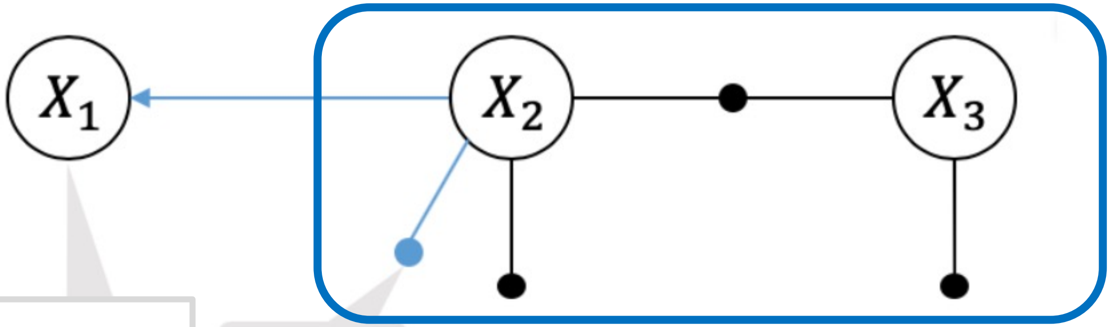
$$g_1(X_2) = \arg\max_{x_1} \phi(x_1, X_2) \qquad \tau(X_2) = \max_{x_1} \phi(x_1, X_2)$$

Posterior via sum-product:

- Eliminate one variable at a time by forming product, then sum:

$$\phi(X_1, X_2) = \phi_1(X_1)\phi_2(X_1)\phi_3(X_1, X_2).$$





$$P(X_1|X_2) = \frac{\phi_1(X_1)\phi_2(X_1)\phi_3(X_1, X_2)}{\tau(X_2)}.$$

$\tau(X_2)$

$$\tau(X_2) \doteq \sum_{X_1} \phi_1(X_1)\phi_2(X_1)\phi_3(X_1, X_2)$$

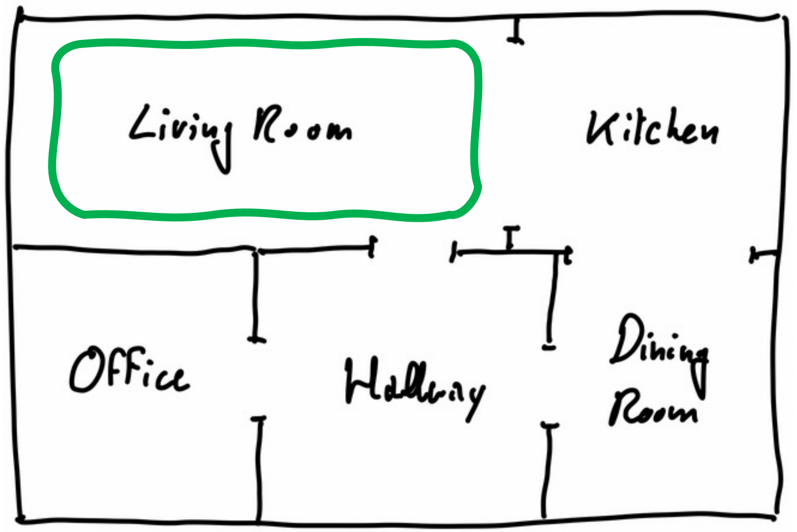
Markov Decision Processes

- Planning is the process of choosing which actions to perform.
- In order to plan effectively, we need quantitative criteria to evaluate actions and their effects.
- MDPs include a reward function that characterizes the immediate benefit of applying an action.
- Policies describe how to act in a given state.
- The value function characterizes the long-term benefits of a policy.
- We assume that the robot is able to ***know*** its current state with certainty.

➢ ***We will see how to define reward functions and use these to compute optimal policies for MDPs.***

Reward Functions

- Most general form depends on current state, action, and next state: $R: \mathcal{X} \times \mathcal{A} \times \mathcal{X} \rightarrow \mathbb{R}$
- In our example, we just care about where we end up after taking an action:




```
def reward_function(state:int, action:int, next_state:int):
    """Reward that returns 10 upon entering the living room."""
    return 10.0 if next_state == "Living Room" else 0.0

print(reward_function("Kitchen", "L", "Living Room"))
print(reward_function("Kitchen", "L", "Kitchen"))
```

```
10.0
0.0
```

Expected Reward

- A greedy way to act would be to calculate the immediate expected reward for every possible action:

$$\boldsymbol{\bar{R}(x,a) = E[R(x,a,X')]}$$

- Since we know the transition probabilities, we can easily compute this:

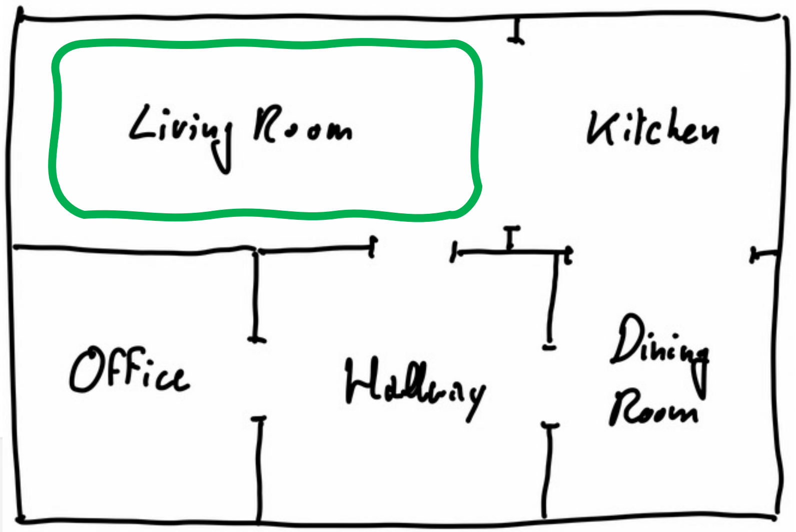
$$\bar{R}(x,a) \doteq E[R(x,a,X')] = \sum_{x'} P(x'|x,a)R(x,a,x')$$

- We then have a simple **greedy planning** algorithm:

$$a^* = \arg\max_{a \in \mathcal{A}} E[R(X_t, a, X_{t+t})]$$

Example

- The expected immediate reward for all four actions in the Kitchen:




```python
x = vacuum.rooms.index("Kitchen")
for a in range(4):
    print(f"Expected reward ({vacuum.rooms[x]}, {vacuum.action_space[a]}) = {T[x,a] @ R[x,a]}")
```

✓ 0.9s

```
Expected reward (Kitchen, L) = 8.0
Expected reward (Kitchen, R) = 0.0
Expected reward (Kitchen, U) = 0.0
Expected reward (Kitchen, D) = 0.0
```

- Hence, when in the kitchen, always do L !
- This is a greedy policy

When we continue…

- Rollouts and Utility
- Policies
- The value of a policy
- Policy and value iteration