COMPSCI 389
Introduction to Machine Learning

Basics and Reward Design

Prof. Philip S. Thomas (pthomas@cs.umass.edu)

Review

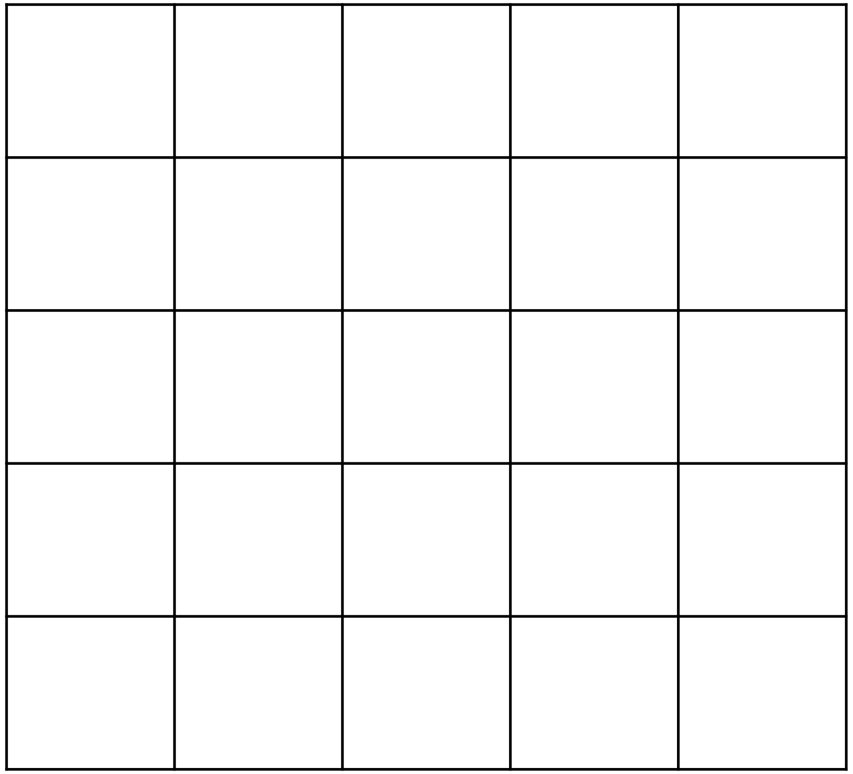
- An RL agent's goal is to find a policy that maximizes the expected return (the expected sum of rewards it receives).

Gridworlds

- Gridworlds are common examples used when learning about RL algorithms.
- They are not important problems, but rather tools for understanding RL and RL agent behavior.
- Gridworlds range in difficulty from trivial to nearly impossible.

Gridworlds: States

- Each cell in the grid is a state.



Gridworlds: States

- Each cell in the grid is a state.
  - They could be numbered, 1, 2, 3, …

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

Gridworlds: States

- Each cell in the grid is a state.
  - They could be numbered, 1, 2, 3, …
  - They could be represented as (x, y) coordinates
- RL includes problems with **continuous states** (e.g., joint angles, blood glucose, etc.).
- This problem has **discrete states**.

(1,1)	(2,1)	(3,1)	(4,1)	(5,1)
(1,2)	(2,2)	(3,2)	(4,2)	(5,2)
(1,3)	(2,3)	(3,3)	(4,3)	(5,3)
(1,4)	(2,4)	(3,4)	(4,4)	(5,4)
(1,5)	(2,5)	(3,5)	(4,5)	(5,5)

Gridworlds: States

- Each cell in the grid is a state.
  - They could be numbered, 1, 2, 3, …
  - They could be represented as (x, y) coordinates
- RL includes problems with **continuous states** (e.g., joint angles, blood glucose, etc.).
- This problem has **discrete states**.
- For simplicity, at first I recommend thinking of discrete states as being integers, $1, 2, \ldots$

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

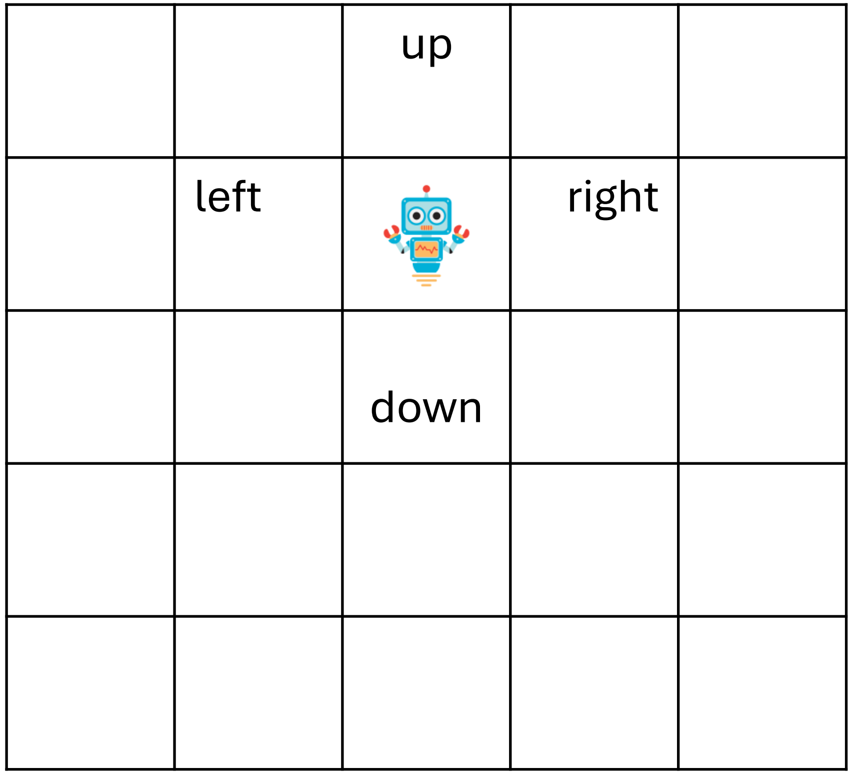
Gridworlds: States

- The set of all possible states in an RL problem is called the **state set,** $\mathcal{S}$.
- Here, $\mathcal{S} = \{1,2,\dots,25\}$
- The state at time t is S_t

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

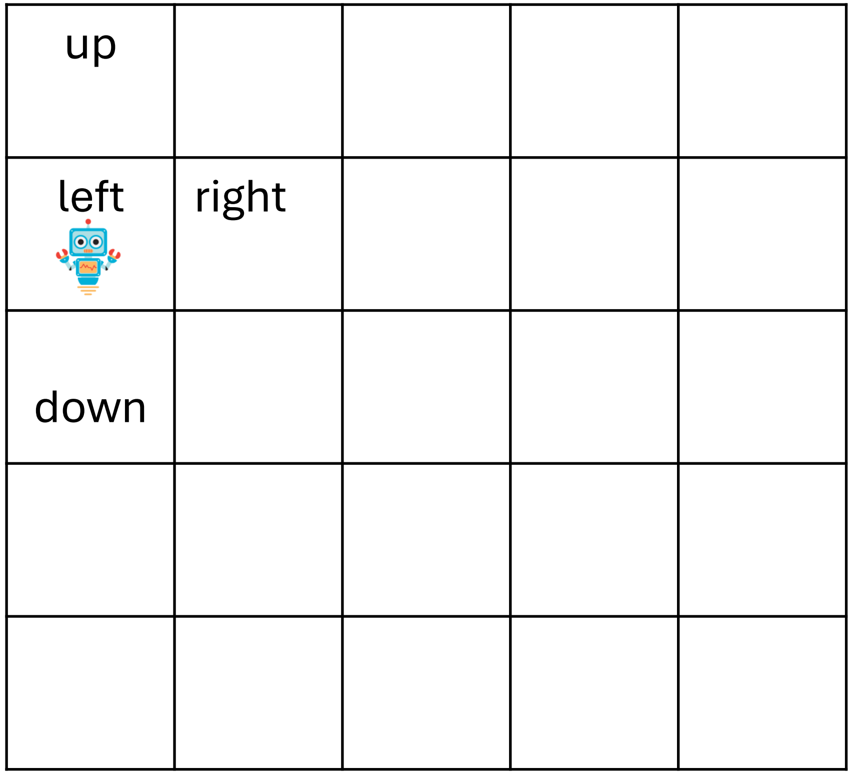
Gridworlds: Actions

- There are typically four actions, up, down, left, and right.
- The set of possible actions is called the **action set** and is denoted by $\mathcal{A}$.
- Here $\mathcal{A} = \{\text{up}, \text{down}, \text{left}, \text{right}\}$
- The action at time t is A_t



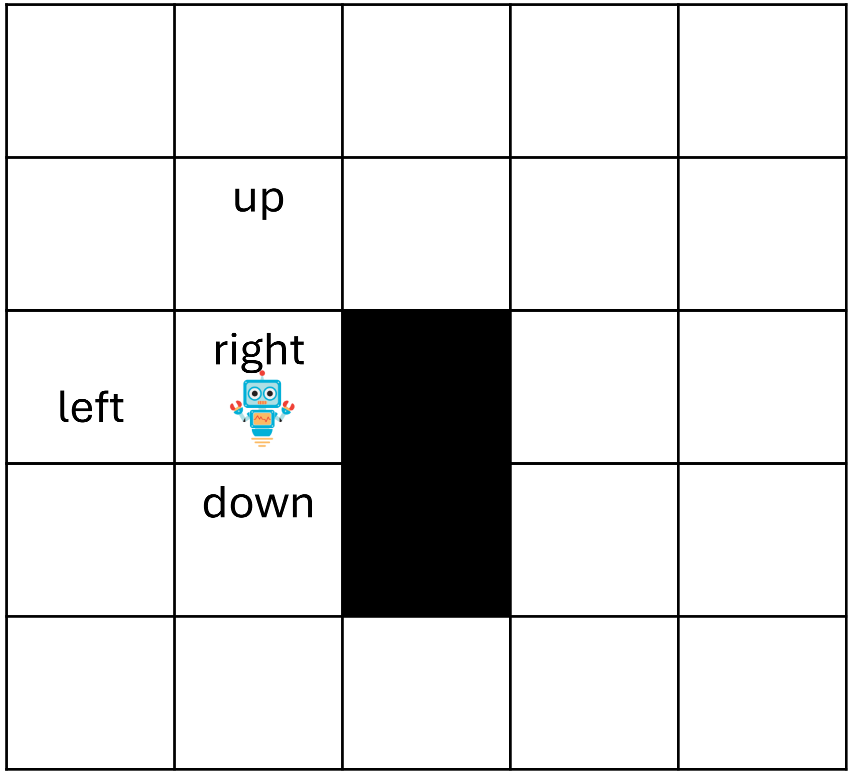
Gridworlds: Transition Dynamics

- Taking an action that would cause the agent to leave the grid usually results in the agent not moving.



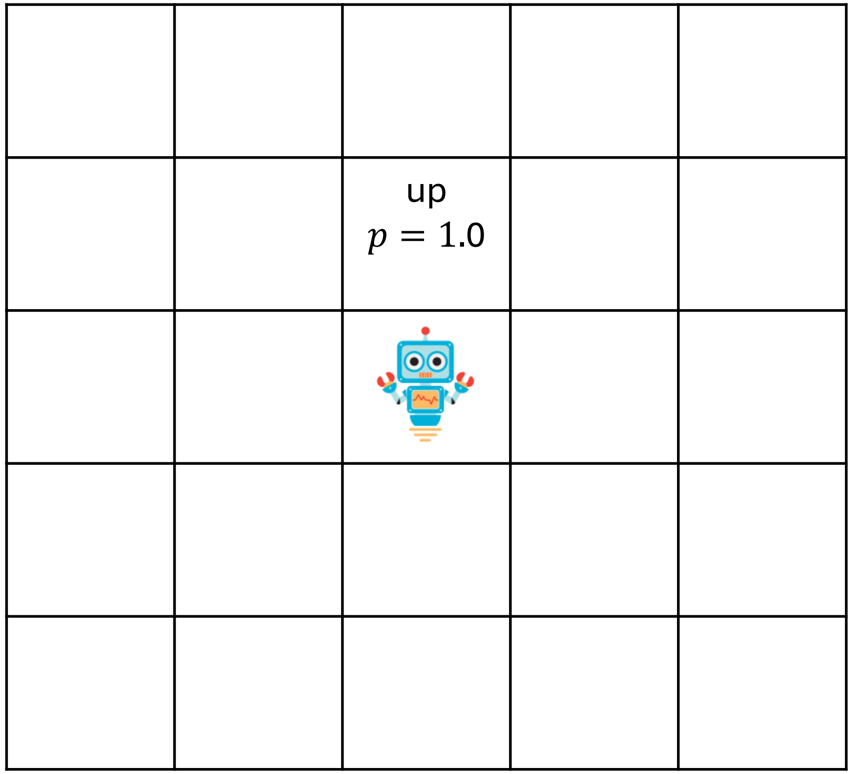
Gridworlds: Transition Dynamics

- Taking an action that would cause the agent to leave the grid usually results in the agent not moving.
- Sometimes gridworlds contain "obstacles", which are cells in the grid that cannot be entered.



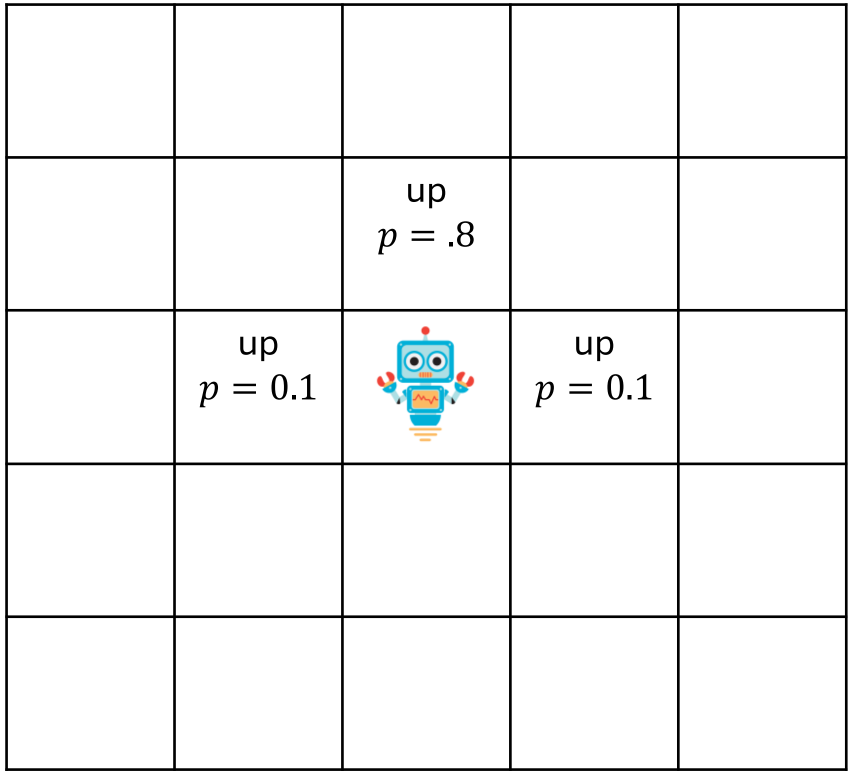
Gridworld: State Transition Dynamics

- Often the action the agent selects always succeeds (assuming the agent doesn't leave the grid).



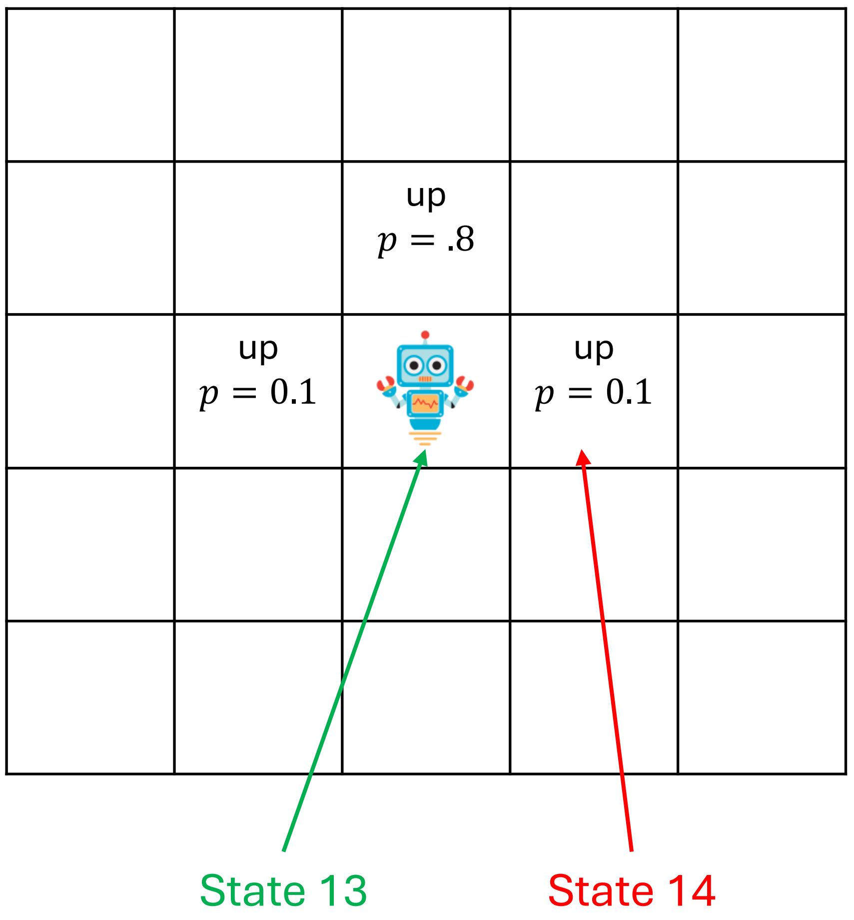
Gridworld: State Transition Dynamics

- Often the action the agent selects always succeeds (assuming the agent doesn't leave the grid).
- Sometimes actions have a probability of failing or sending the agent in the wrong direction.



Gridworld: State Transition Dynamics

- The function describing how states transition given actions is called the **transition function**, p
- For all states s and s' and actions a:
  $p(s, a, s') = \Pr(S_{t+1} = s' | S_t = s, A_t = a)$
- Here, $p(13, \text{up}, 14) = 0.1$



Gridworld: Rewards

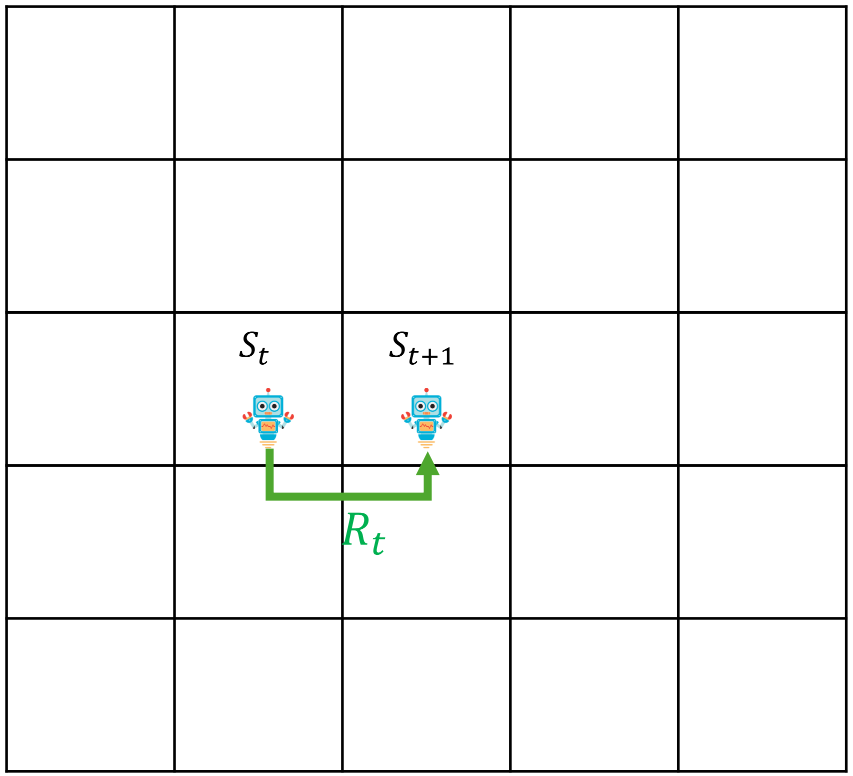
- Whenever:
  - The state is S_t
  - The agent selects action A_t
  - The state transitions to S_{t+1}
- The environment also emits a reward, R_t.
- The **reward function** R gives the expected reward given a state and action:

$$R(s, a) = \mathbf{E}[R_t | S_t = s, A_t = a].$$

- If rewards are deterministic given s and a, then the reward function specifies the reward:

$$R_t = R(S_t, A_t).$$

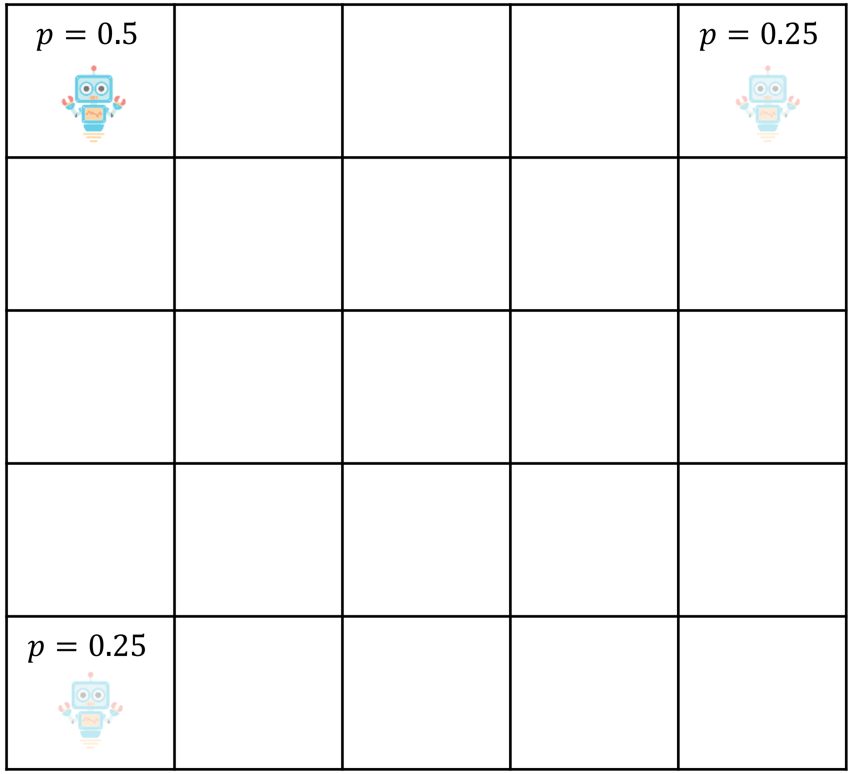
  - We will focus on this simplified setting.



Gridworld: Initial State Distribution

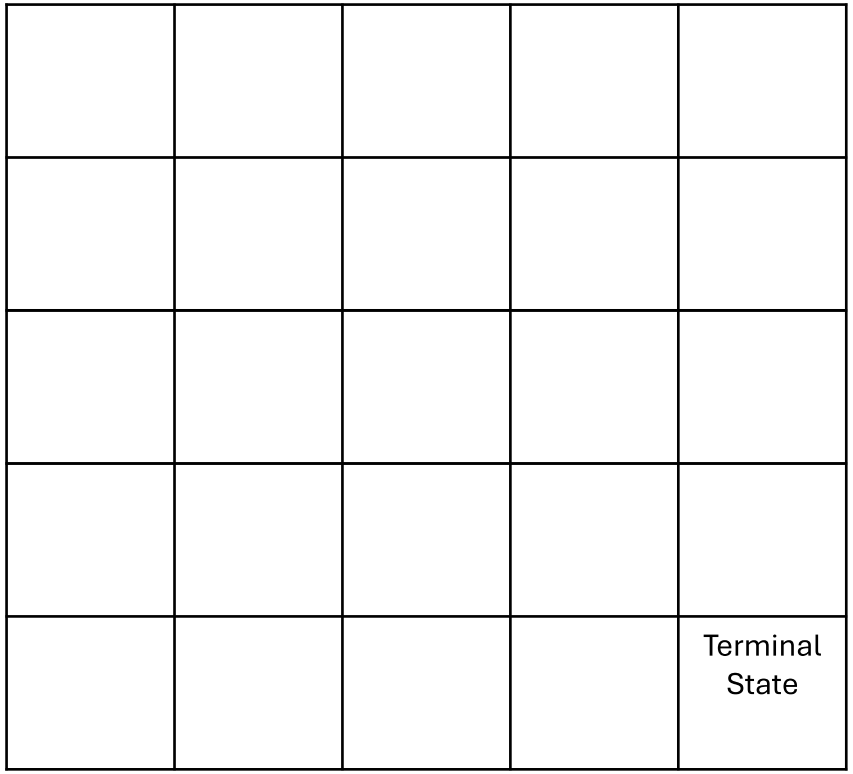
- The initial state S_0 need not be deterministic.
- The **initial state distribution** d_0 specifies the distribution of the initial state:

$$d_0(s) = \Pr(S_0 = s)$$



Gridworld: Teriminal States

- The definition of **terminal states** varies by source.
- For this course, an episode ends when the agent enters a **terminal state**.
- Sometimes the goal is for the agent to avoid the terminal state
  - Episodes end when the robot falls over
- Sometimes the goal is for the agent to reach the terminal state
  - Episodes end when the robot escapes the maze
  - Sometimes these terminal states are called **goal states**.
- Sometimes the goal does not relate to terminal states.



Gridworld: Policy and Optimal Policies

- A **policy** is one way for an agent to select actions, and is denoted by π, where
$$\pi(s,a) = \Pr(A_t = a | S_t = s).$$
- The agent's goal is to find an **optimal policy** π^*, which is one that maximizes the expected discounted sum of rewards:

Note: There could be more than one optimal policy!

$$\pi^* \in \arg\max_{\pi} \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right].$$

  - $\gamma \in [0,1]$ is the **reward discount parameter**.
  - Smaller values of gamma result in a smaller weight on rewards that occur farther in the future.
    - Most people would take one cookie today rather than two cookies a year from now!

Markov Decision Process

- A **Markov decision process** (MDP) is a mathematical formulation of an RL problem.
- It is a tuple $(\mathcal{S}, \mathcal{A}, p, R, d_0, \gamma)$
  - $\mathcal{S}$ is the set of possible states or **state set**
  - $\mathcal{A}$ is the set of possible actions or **action set**
  - p is the **transition function**, where $p(s, a, s') = \Pr(S_{t+1} = s' | S_t = s, A_t = a)$
  - R is the **reward function**, where $R(s, a) = \mathbf{E}[R_t | S_t = s, A_t = a]$
  - d_0 is the **initial state distribution**, where $d_0(s) = \Pr(S_0 = s)$
  - $\gamma \in [0,1]$ is the **reward discount parameter**
- A policy π characterizes action selection: $\pi(s, a) = \Pr(A_t = a | S_t = s)$
- The agent's goal when faced with an MDP is to find an optimal policy:

$$\pi^* \in \arg\max_{\pi} \mathbf{E}\left[\sum_{t=0}^{\infty} \gamma^t R_t\right].$$

Why “Markov” decision process?

- The Markov property means that the *future* is independent of the *past* given the *present*.
- The transition function satisfies the Markov property:

$$p(s, a, s') = \Pr(S_{t+1} = s' | S_t = s, A_t = a)$$

  - The distribution of the “next state” S_{t+1} does not depend on any of the states, actions, or rewards prior to S_t (when S_t is known)

Parameterized Policy

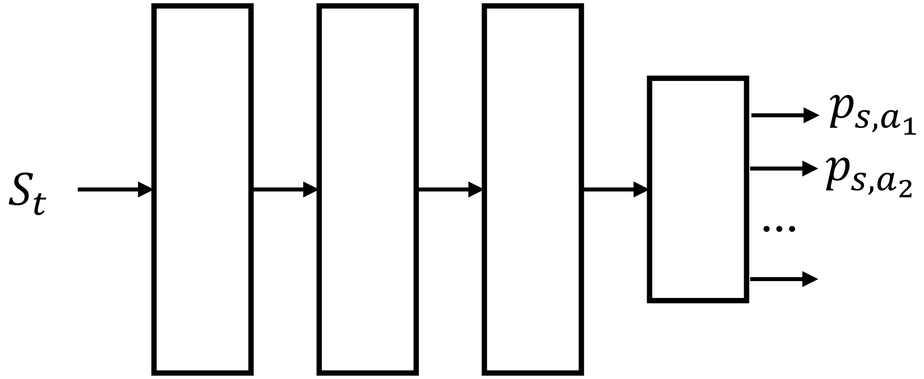
- A **parametric policy** π is like a “parametric model” in supervised learning - a policy that has **policy parameters** θ.
  - This is akin to a parametric model for supervised learning that has *model parameters* w.

How to represent π?

- Tabular softmax:
  - Store a value $\theta_{s,a}$ for each state s and action a
  - $\pi(s,a) = \Pr(A_t = a | S_t = s) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}$
  - **Note**: Limited to problems with finite state and action sets
- Linear softmax:
  - Store a vector of weights θ_a for each action a.
  - Define a feature generating function ϕ that takes states as input
    - $\phi(s)$ is a vector of features for state s
  - $\pi(s,a) = \frac{e^{\theta_a \cdot \phi(s)}}{\sum_{a'} e^{\theta_{a'} \cdot \phi(s)}} = \frac{e^{\left(\sum_{i=1}^{m} \theta_{a,i} \phi_i(s)\right)}}{\sum_{a'} e^{\left(\sum_{i=1}^{m} \theta_{a',i} \phi_i(s)\right)}}$
  - **Note**: Limited to problems with finite action sets (but works with continuous states!)

How to represent π?

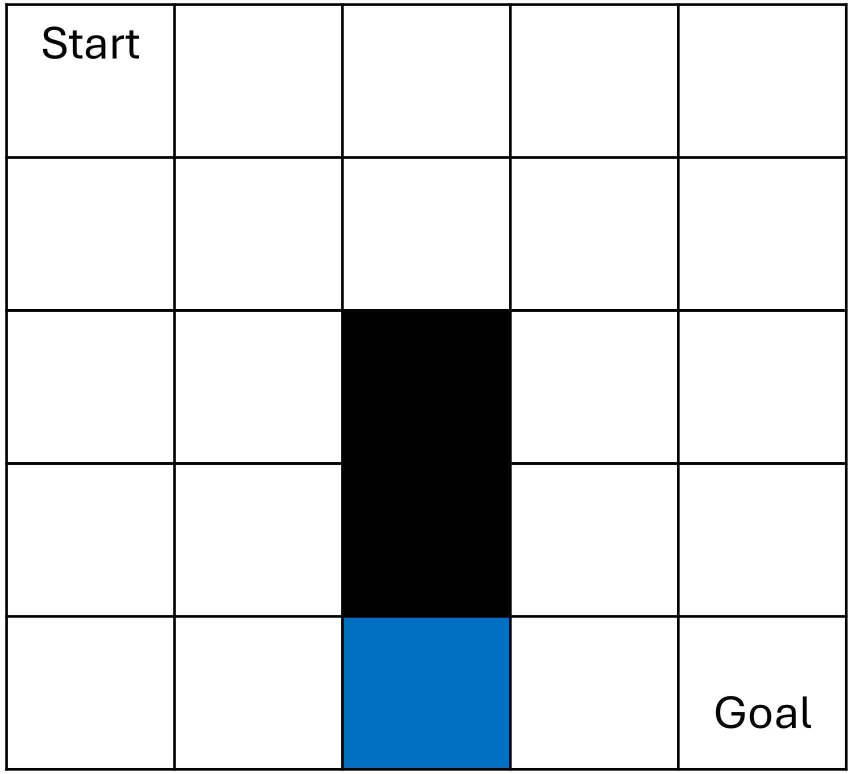
- Artificial Neural Network (with weights θ)



- $\pi(s,a) = \frac{e^{p_{s,a}}}{\sum_{a'} e^{p_{s,a'}}}$

Reward Design

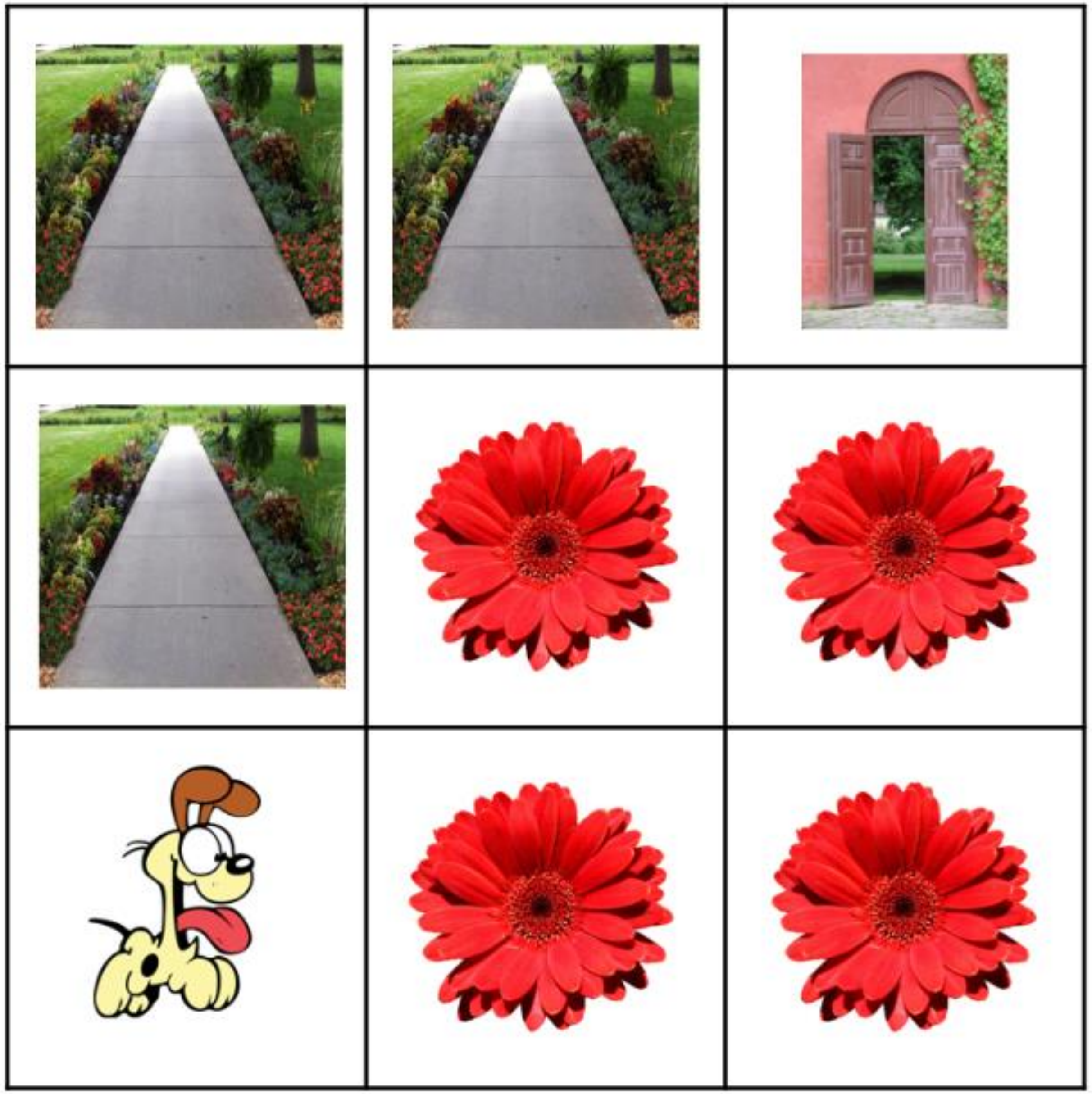
- The agent always starts in the top-left.
- The agent's goal is to reach the bottom right state (which is terminal)
- Actions succeed with probability 0.7
- Actions fail with probability 0.3
  - When actions fail, one of the other three actions is applied (each with probability 0.1)
- There are two obstacle cells (black).
- There is one water-filled cell (blue) that should be avoided.
- **Question**: How would you define rewards (and γ) for this problem?



Reward Design

- Do **not** reward the agent based on how you *think* it should solve the problem.
  - This often results in completely different undesirable behavior.
- Provide rewards based only on the main goal.
- **Shaping rewards** are rewards designed to encourage an agent towards specific behavior.
  - There are rules that can be followed to ensure they do not change optimal behavior.
  - Avoid shaping rewards otherwise!

Example:



End