Minor in AI

Reinforcement Learning

Exploring Grid World

April 17, 2025

1 Grid World

We begin with a **Grid World** environment consisting of a 3×3 grid where each cell represents a state. Below is the layout of the states, each labeled with a unique value of s (state number):

s = 0.0	**s = 1.0**	**s = 2.0**
s = 3.0	**s = 4.0**	**s = 5.0**
s = 6.0	**s = 7.0**	**s = 8.0**

Figure 1: Grid World

The agent can take actions `Up`, `Down`, `Left`, or `Right`, and the goal is to maximize the cumulative reward over time.

2 Policy

A **policy** π is a mapping from states to actions:

$$\pi(s) = a$$

It determines the action an agent takes when in state s. Policies can be:

- **Deterministic:** Always takes the same action for a given state.
- **Stochastic:** Defines a probability distribution over actions.

3 Value Iteration Algorithm

Value Iteration is a dynamic programming technique used to find the optimal policy. It iteratively updates the **value function** $V(s)$ using the Bellman optimality equation:

$$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a) \left[R(s,a,s') + \gamma V_k(s') \right]$$

What is happening here?

We are updating $V(s)$ based on future estimates. This process is called bootstrapping.
Key idea: At each iteration $k+1$, we estimate the value of each state s assuming we act optimally from now on.
We look at each possible action, consider its outcomes, and take the maximum expected value.
Analogy: Think of it like figuring out the best move in chess — you evaluate all next moves and pick the one leading to the best outcome.

Steps of Value Iteration

1. Initialize $V(s)$ arbitrarily (e.g., $V(s) = 0$ for all s)
2. Repeat until convergence:
   - For each s, update $V(s)$ using the Bellman equation.
3. Derive the optimal policy $\pi^*(s) = \arg\max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V(s')\right]$

4 Bellman Optimality Equation

The Bellman equation for the optimal value function $V^*(s)$ is:

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V^*(s')\right]$$

This equation captures the recursive nature of decision making: the value of a state is the best expected return achievable by any action from that state onward.

Understanding the Bellman Equation

This formula computes the expected return of each action, and picks the best one via $\max_a$.

- The term $P(s'|s,a)$ is the probability of ending up in state s' after taking action a in state s.
- $R(s,a,s')$ is the immediate reward received for that transition.
- $V^*(s')$ is the value of the next state — i.e., how good it is to be in s'.
- γ is the discount factor that downweights future rewards.

It's essentially saying: "*Pick the action whose outcomes (weighted by probability) give the best combination of immediate and future rewards.*"

4.1 Intuition

The Bellman Optimality Equation expresses the idea that:

The best value I can get from state s is by choosing the best action a, considering all possible next states s' , and weighing the immediate reward plus the discounted value of the next state.

The optimal value of a state s is the maximum expected return achievable by choosing the best action a, accounting for both the immediate reward and the discounted future value.

Let's understand this through the context of the above 3×3 grid world:

4.2 Assumptions

- Actions: Up, Down, Left, Right
- Reward: Each move gives a reward of -1
- Terminal State: $s = 8$ with reward 0
- Transition probabilities:
  - Intended move: 80% probability
  - Slip (stay in same state): 20% probability
- Discount factor: $\gamma \in [0, 1)$

Example from State $s = 4$

From state 4 (center of the grid), consider taking the `Right` action:

- With probability 0.8, move to state $s' = 5$
- With probability 0.2, stay in state $s' = 4$

Then, the Q-value for action `Right` from state 4 is:

$$\begin{aligned} Q(4, \texttt{Right}) &= 0.8 \cdot [-1 + \gamma V^*(5)] + 0.2 \cdot [-1 + \gamma V^*(4)] \\ &= -1 + \gamma \left(0.8 V^*(5) + 0.2 V^*(4)\right) \end{aligned}$$

Similarly, compute $Q(4, a)$ for all actions $a \in \{\texttt{Up}, \texttt{Down}, \texttt{Left}, \texttt{Right}\}$, and take the maximum:

$$V^*(4) = \max_a Q(4, a)$$

You're essentially asking:

" Which direction leads me to the goal quickest, considering both the reward now and future value? "

4.3 Interpretation

The Bellman equation captures the following intuition:

Standing at a state s, you evaluate each possible action a, considering:

- The probability of reaching each next state s'
- The immediate reward $R(s, a, s')$
- The long-term value $V^*(s')$ of where you might end up

Then, you pick the action that gives the best combined expected value.

This recursive structure ensures that the value of each state incorporates the future consequences of the decisions made.

5 Stochastic vs Deterministic Environments

1. **Deterministic:** The next state is uniquely determined by the current state and action.
2. **Stochastic:** The next state is determined by a probability distribution.

In stochastic environments, the transition is defined as:

$$P(s'|s, a) = \text{Probability of transitioning to } s' \text{ given } s \text{ and } a$$

6 Discount Factor γ

The **discount factor** $\gamma \in [0, 1]$ determines how much future rewards are valued:

$$\text{Total return } = R_0 + \gamma R_1 + \gamma^2 R_2 + \cdots$$

- $\gamma = 0$: Only immediate rewards matter.
- $\gamma \approx 1$: Long-term rewards are important.

The discount factor controls how much we care about the future.
If $\gamma = 0$: We only care about immediate rewards.
If $\gamma = 1$: We value future rewards equally.
Typical range: 0.9 or 0.99 — this encourages looking ahead but not indefinitely.

7 Grid World Example for Value Iteration

Let us define:

- Reward $R(s, a, s') = -1$ for every transition (penalty for moving).
- Terminal state at $s = 8$ with reward 0 and no outgoing transitions.
- $\gamma = 0.9$

We initialize $V(s) = 0$ for all s. After one iteration, value updates begin as follows:

$$V_1(s) = \max_a \sum_{s'} P(s'|s,a)\,[-1 + 0.9V_0(s')] = \max_a \sum_{s'} P(s'|s,a)(-1)$$

As iterations proceed, $V(s)$ begins to reflect proximity to the terminal state.

8 Q-values

Q-values represent the value of taking action a in state s:

$$Q(s,a) = \sum_{s'} P(s'|s,a)\,[R(s,a,s') + \gamma V(s')]$$

What is Q-value ?

$Q(s,a)$ estimates the expected return of taking action a in state s and then behaving optimally.
You average over all possible next states s', using $P(s'|s,a)$, and account for both:

- Immediate reward $R(s,a,s')$
- Future return $\gamma V(s')$

This helps separate the value of being in a state from the value of taking a specific action.

The optimal policy is derived from Q-values:

$$\pi^*(s) = \arg\max_a Q(s,a)$$

The arg max operator returns the argument (in this case, the action a) that gives the maximum value of $Q(s,a)$.
In simple terms: *"Which action gives the highest expected return from this state?"*

8.1 Intuition

The Q-value $Q(s,a)$ represents the expected total return if:

- The agent is in state s,
- Takes action a,
- Then follows the optimal policy thereafter.

It incorporates:

1. Immediate reward $R(s,a,s')$
2. Future discounted value $\gamma V(s')$ of the next state
3. Averaged over all possible next states s' using transition probabilities $P(s'|s,a)$

8.2 Optimal Policy from Q-values

$$\pi^*(s) = \arg\max_a Q(s, a) \tag{1}$$

The optimal policy $\pi^*(s)$ is the action that maximizes the expected return from state s. In other words, it selects the action with the highest Q-value.

Let's return to our 3×3 grid world setup:
Assume:

- Transition: Intended move with probability 0.8, staying in the same state with 0.2
- Reward: -1 per move, 0 at terminal state $s = 8$
- $\gamma = 0.9$
- From state $s = 4$, taking action `Right`:
  - With 0.8 probability: go to $s' = 5$
  - With 0.2 probability: remain in $s' = 4$

Then,

$$Q(4, \texttt{Right}) = 0.8 \cdot [-1 + \gamma V(5)] + 0.2 \cdot [-1 + \gamma V(4)]$$

Repeat for all possible actions (Up, Down, Left, Right), and derive:

$$\pi^*(4) = \arg\max_a Q(4, a)$$

8.3 Policy Interpretation

The optimal policy map $\pi^*(s)$ tells the agent:

> *"At each state s, choose the action a that gives the best trade-off between immediate reward and future value."*

This is how an intelligent agent decides what to do at each step — by computing Q-values for each action and picking the best one.

9 Probability-based State Transitions

Example: If in state $s = 4$, and action `Up` is chosen:

$$P(s' = 1 | s = 4, a = \texttt{Up}) = 0.8$$
$$P(s' = 4 | s = 4, a = \texttt{Up}) = 0.2 \quad \text{(slip/stay)}$$

What are Stochastic Transitions?

In stochastic environments, an action doesn't always lead to a fixed result.
Here, taking `Up` from state 4 usually goes to state 1 — but there's a 20% chance it doesn't work and the agent stays in place.
This models real-world uncertainty, like wind blowing the agent off course or a button failing to respond.

In value iteration, we compute:

$$V(s) = \max_a \left(\sum_{s'} P(s'|s, a) \left[R(s, a, s') + \gamma V(s') \right] \right)$$

10 Code Implementation

The code is available for hands-on exploration at the following link: Click here!.

1. GridWorld State Numbering

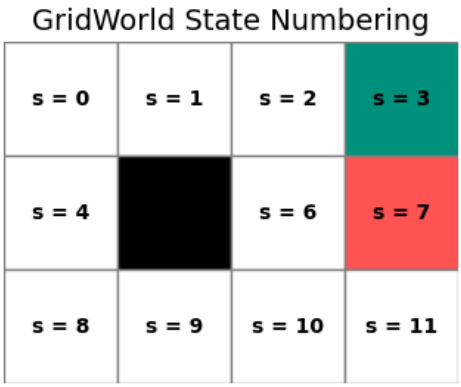

Figure 2: Grid World

Each cell is assigned a state number. For example:

- State 3: terminal state with positive reward,
- State 7: terminal state with negative reward,
- State 5: obstacle (black cell), which is impassable.

2. Reward Function Map

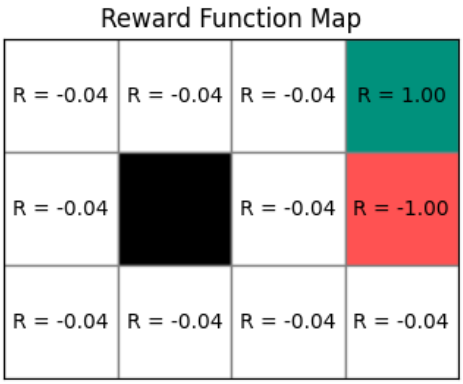

Figure 3: Reward at each step

Rewards:

- Most cells: step penalty $R = -0.04$,
- Goal state (state 3): $R = +1.0$,

- Pit state (state 7): $R = -1.0$,
- Obstacles: no transitions possible.

3. Transition Dynamics

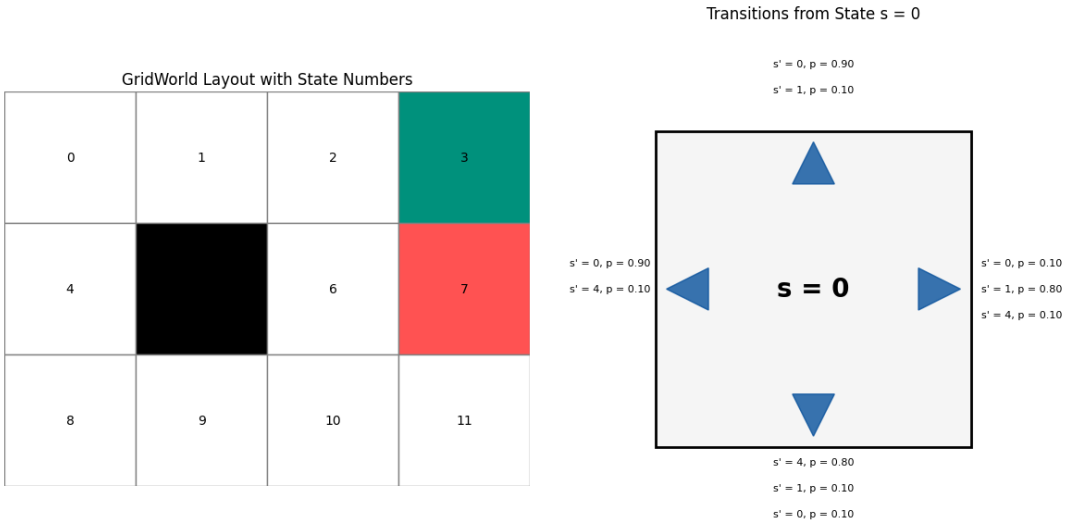

Figure 4: What to do when you are at state s?

We assume stochastic transitions. From state $s = 0$, attempting to move down:

- $s' = 4$ with probability 0.8,
- $s' = 1$ with probability 0.1,
- Staying in $s = 0$ with 0.1 (if blocked).

Value function update:

$$V(s) = \max_a \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma V(s')\right]$$

4. Random Policy Behavior

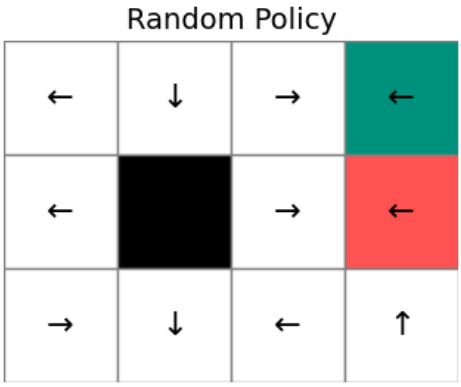

Figure 5: What will happen if you take a random policy?

Initially, a random policy is used. Each state takes an action uniformly at random, which is iteratively improved.

5. Converged Value Function and Optimal Policy

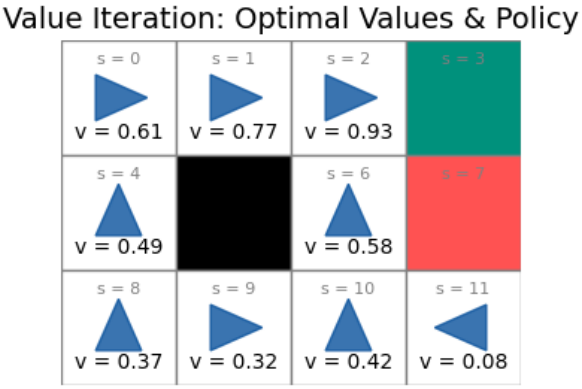

Figure 6: Optimal Situation

After convergence:

- The value function $V(s)$ reflects expected rewards,
- Arrows represent the optimal action $\pi^*(s)$,
- The agent is guided toward high-value states.