**Vivekanand Education Society’s Institute of Technology**

**Department of AIDS Engineering**



**Subject: Reinforcement Learning**

**Class: D16AD**

| **ROLL NO: 30** | **NAME: SUHANEE KANDALKAR** | | |
| --- | --- | --- | --- |
| **EXP NO: 07** | T**ITLE:**  **Value iteration** | | |
| **DOP:** |  | **DOS: 29/03/25** |  |
| **GRADES:** | **LOs MAPPED:** | | **SIGNATURE:** |

## 

## Aim**:**

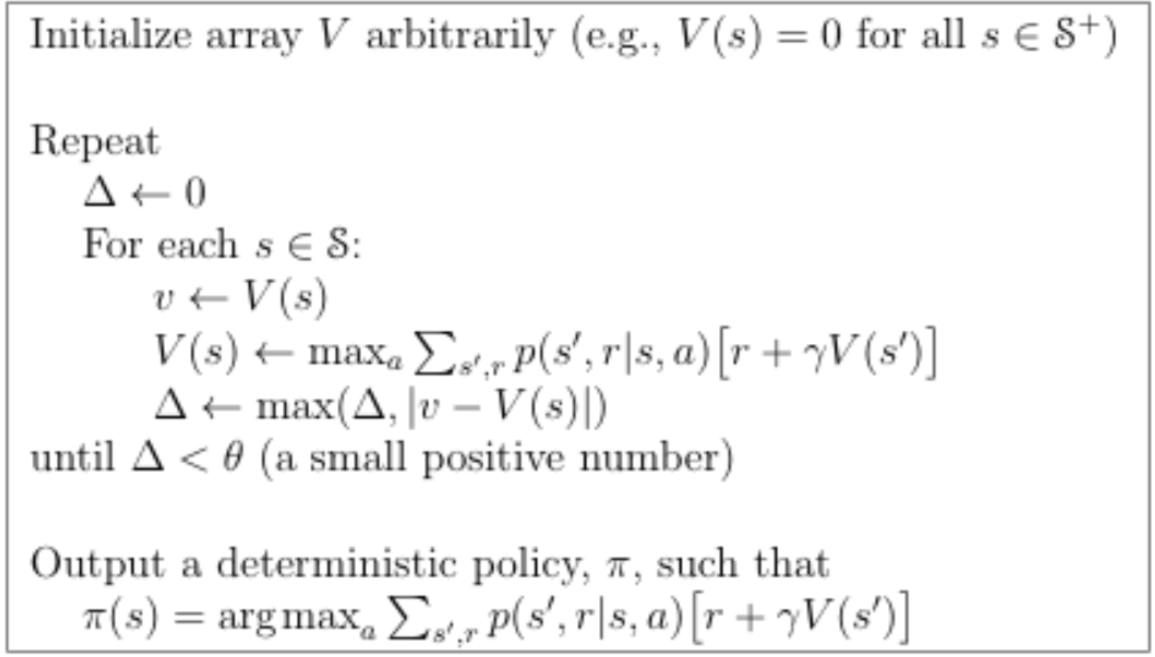
Apply a value iteration algorithm to find optimal policies for the grid word environment.

## Theory**:**

Value Iteration for Grid World Environment

Value iteration is a fundamental algorithm in Reinforcement Learning (RL) used to compute the optimal policy for an agent navigating an environment. It is based on Dynamic Programming (DP) and iteratively updates the state-value function until convergence.

📖 Reference: Sutton & Barto, \*Reinforcement Learning: An Introduction\* ([Read more](https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf))



1️⃣ Markov Decision Process (MDP)

A Grid World environment can be modeled as a Markov Decision Process (MDP) , which consists of:

- States (S): Grid positions \((x, y)\).

- Actions (A): {UP (↑), DOWN (↓), LEFT (←), RIGHT (→)}.

- Transition function \( P(s' | s, a) \): Probability of moving from state \( s \) to \( s' \) given action \( a \).

- Rewards (R): Immediate feedback for taking action \( a \) in state \( s \).

- Discount factor \( \gamma \): Balances short-term vs. long-term rewards.

📖 Reference: Martin Osborne, \*An Introduction to Game Theory\* ([Book link](https://epubs.siam.org/doi/book/10.1137/1.9780898718577))

2️⃣ Bellman Optimality Equation

The Value Iteration Algorithm is based on the Bellman Optimality Equation :

\[

V(s) = \max\_a \sum\_{s'} P(s' | s, a) [R(s, a, s') + \gamma V(s')]

\]

This equation recursively updates the state-value function \( V(s) \), choosing the action that maximizes expected future rewards .

📖 Reference: Richard Bellman, \*Dynamic Programming\* ([Read Bellman's work](https://www.rand.org/pubs/commercial\_books/CB137-1.html))

3️⃣ Value Iteration Algorithm

-> Step 1: Initialize Value Function

- Set \( V(s) = 0 \) for all states except terminal states.

-> Step 2: Update Values Until Convergence

- Use the Bellman equation to update each state’s value:

\[

V(s) \leftarrow \max\_a \sum\_{s'} P(s' | s, a) [R(s, a, s') + \gamma V(s')]

\]

- Stop when the change in values is below a threshold \( \theta \).

-> Step 3: Extract Optimal Policy

- Select the action that maximizes value:

\[

\pi^\*(s) = \arg\max\_a \sum\_{s'} P(s' | s, a) [R(s, a, s') + \gamma V(s')]

\]

- The optimal policy directs the agent toward the goal state .

📖 Reference: Sutton & Barto, \*Policy Evaluation and Value Iteration\* ([MIT OpenCourseWare](https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-231-dynamic-programming-and-stochastic-control-fall-2015/lecture-notes/MIT6\_231F15\_LecNote8.pdf))

4️⃣ Visualization of the Results

After running value iteration , we obtain:

-> ✅ Grid World (Rewards)

```

-1 -1 -1 -1 -1

-1 -1 -1 -1 -1

-1 -1 -1 -1 -1

-1 -1 -1 -1 -1

-1 -1 -1 -1 10

```

-> ✅ Optimal Value Function

```

-5.70 -4.90 -4.10 -3.40 -2.80

-4.90 -4.10 -3.40 -2.70 -2.20

-4.10 -3.40 -2.70 -2.10 -1.60

-3.40 -2.70 -2.10 -1.50 -1.10

-2.80 -2.20 -1.60 -1.10 10.00

```

-> ✅ Optimal Policy

```

↓ ↓ ↓ ↓ ↓

↓ ↓ ↓ → ↓

↓ ↓ → → ↓

↓ → → → ↓

→ → → → G

```

🔹 Rightmost column moves DOWN (↓) towards the goal .

🔹 Other paths move RIGHT (→) or DOWN (↓) optimally .

📖 Reference: Stanford AI Course, \*Markov Decision Processes & Value Iteration\* ([Stanford AI Lecture](https://cs.stanford.edu/people/karpathy/reinforcejs/gridworld\_dp.html))

5️⃣ Applications of Value Iteration

- Robotics 🤖: Autonomous navigation (e.g., self-driving cars).

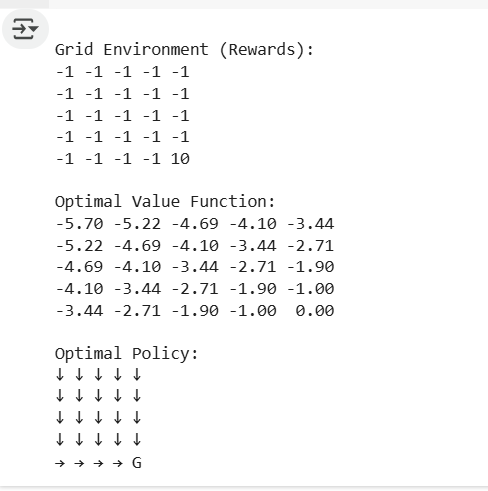
- Finance 💰: Optimal decision-making in investment strategies.

- Healthcare 🏥: AI-assisted medical decision-making.

- Gaming 🎮: AI in strategic games (e.g., chess, Go).

📖 Reference: Reinforcement Learning Applications ([DeepMind Research](https://deepmind.com/research/highlighted-research/))

# Code:



[RL\_exp7\_30.ipynb](https://colab.research.google.com/drive/1VE6IGaNXwSsKVQemHiiU2qooM-ngP0wb?authuser=0#scrollTo=1-NWpUuJs36b)

# Conclusion :

Value Iteration is a powerful Dynamic Programming approach for computing optimal policies in Grid World and other MDP environments. By iteratively updating state values and extracting the best actions, we obtain a robust policy for maximizing cumulative rewards. 🚀