Applied Machine Learning

Homework 13

Anand Kamble
Department of Scientific Computing
Florida State University

Introduction

In this report, we implement the value iteration algorithm to find the optimal policy for navigating a 24 $\times$ 32 grid map. The map contains walls, valid positions, and a goal state with a reward. The agent can move left, right, up, or down to adjacent valid positions. We use a discount factor $\gamma = 0.9$ and run the value iteration algorithm for up to 50 epochs, displaying the value function $V(s)$ every 5 epochs.

Approach and Implementation

Map Representation

```
map_data:np.ndarray=pd.read_csv('map_24x32.csv', header=None).values
rows,cols = map_data.shape
```

Initialization

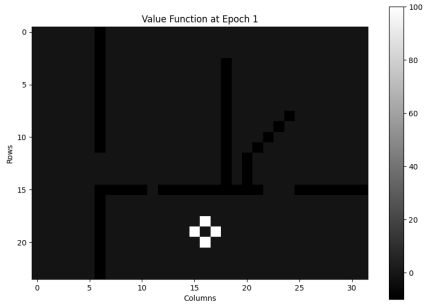
```
V:np.ndarray = np.copy(map_data)
gamma = 0.9
actions:dict[str, tuple[int, int]] = {'L': (0, -1),  # Left
           'R': (0, 1),   # Right
           'U': (-1, 0),  # Up
           'D': (1, 0)}   # Down
policy:np.ndarray = np.full((rows, cols), '', dtype=object)
```

Value Iteration Algorithm

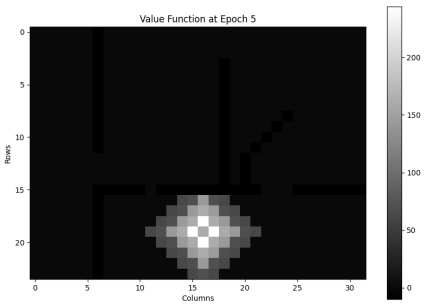
```
for epoch in range(1, 51):
    V_new = np.copy(V)
    for i in range(rows):
        for j in range(cols):
            if map_data[i, j]>= 0:
                action_values ={}
                for action,(di, dj) in actions.items():
                    ni, nj= i + di,j + dj
                    if (0<=ni<rows and 0<=nj<cols and map_data[ni, nj]>= 0):
                        if map_data[ni, nj]==1:
                            reward =100  # High reward at the goal
                        else:
                            reward= -1   # Penalty per move
                        value =reward + gamma *  V[ni, nj]
                        action_values[action]=value
                if action_values :
                    best_action =max(action_values, key=action_values.get)
                    V_new[i, j] =action_values[best_action]
                    policy[i, j] =best_action
    if np.allclose(V, V_new):
        print(f'Converged at epoch {epoch}')
        break
    V = V_new
```

Results

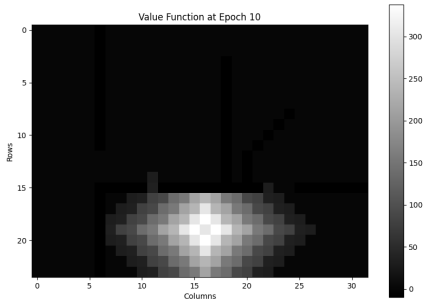
Value Function Evolution



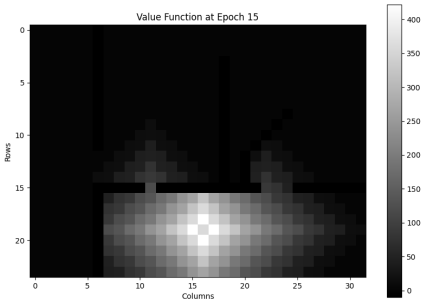
(a) Value Function at Epoch 1



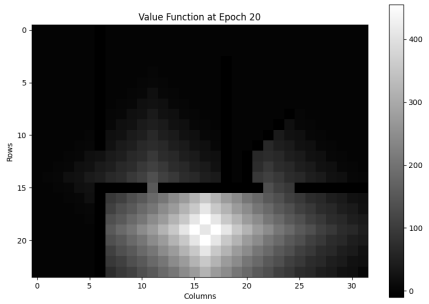
(b) Value Function at Epoch 5



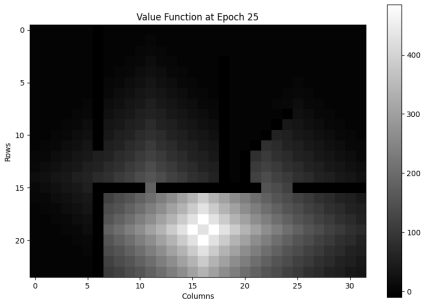
(c) Value Function at Epoch 10



(d) Value Function at Epoch 15

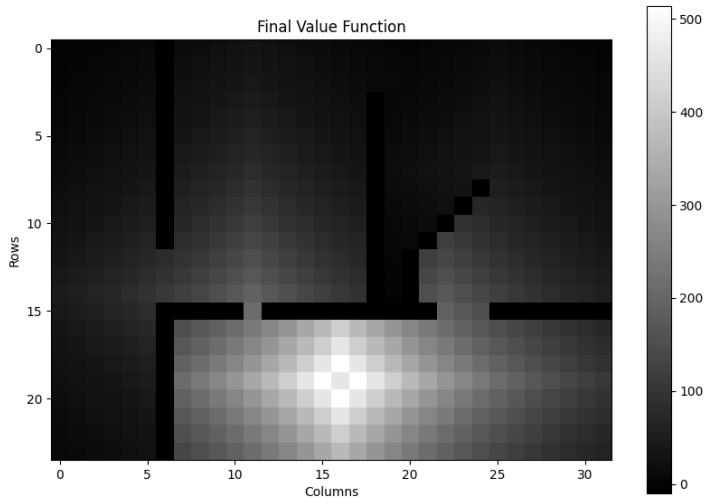


(e) Value Function at Epoch 20



(f) Value Function at Epoch 25

Figure 1: Value Function Evolution Over Epochs



(a) Final Value Function after 50 Epochs

Figure 2: Final Results

Final Learned Policy

```
R R R R R D X R R R R D L L L L L L L L R R R R D L L L L L L
R R R R R D X R R R R D L L L L L L L L R R R R D L L L L L L
R R R R R D X R R R R D L L L L L L L L R R R R D L L L L L L
R R R R R D X R R R R D L L L L L L X R R R R R R D L L L L L L
R R R R R D X R R R R D L L L L L L X R R R R R R D L L L L L L
R R R R R D X R R R R D L L L L L L X R R R R R R D L L L L L L
R R R R R D X R R R R D L L L L L L X R R R R R R D L L L L L L
R R R R R D X R R R R D L L L L L L X R R R R R R D L L L L L L
R R R R R D X R R R R D L L L L L L X R R R R R R D L L L L L L
R R R R R D X R R R R D L L L L L L X R R R R U X D L L L L L L
R R R R R D X R R R R D L L L L L L X R R R U X D L L L L L L L
R R R R R D X R R R R D L L L L L L X R R U X D L L L L L L L L
R R R R R D X R R R R D L L L L L L X R U X D L L L L L L L L L
R R R R R R R R R R D L L L L L L X U X R D L L L L L L L L L
R R R R R R R R R R D L L L L L L X U X R D L L L L L L L L L
R R R R R R R R R R D L L L L L L X U X R D L L L L L L L L L
R R R R R U X X X X X D X X X X X X X X X X D L L X X X X X X X
R R R R R U X R R R R R R R R R D L L L L L L L L L L L L L L L
R R R R R U X R R R R R R R R R D L L L L L L L L L L L L L L L
R R R R R U X R R R R R R R R R D L L L L L L L L L L L L L L L
R R R R R U X R R R R R R R R R L L L L L L L L L L L L L L L L
R R R R R U X R R R R R R R R R U L L L L L L L L L L L L L L L
R R R R R U X R R R R R R R R R U L L L L L L L L L L L L L L L
R R R R R U X R R R R R R R R R U L L L L L L L L L L L L L L L
R R R R R U X R R R R R R R R R U L L L L L L L L L L L L L L L
```

Appendix: Full Code

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
map_data:np.ndarray=pd.read_csv('map_24x32.csv', header=None).values
rows:int = 0
cols:int = 0
rows, cols = map_data.shape
V:np.ndarray = np.copy(map_data)

gamma = 0.9
actions:dict[str, tuple[int, int]] = {'L': (0, -1), 'R': (0, 1), 'U': (-1, 0), 'D':
↪  (1, 0)}
policy:np.ndarray = np.full((rows, cols), '', dtype=object)
max_epochs = 50
for epoch in range(1, max_epochs + 1):
    V_new:np.ndarray = np.copy(V)
    for i in range(rows):
        for j in range(cols):
            if map_data[i, j] >= 0:
                action_values = {}
                for action, (di, dj) in actions.items():
                    ni, nj = i + di, j + dj
                    if 0 <= ni < rows and 0 <= nj < cols and map_data[ni, nj] >= 0:
                        if map_data[ni, nj] == 1:
                            reward = 100
                        else:
                            reward = -1
                        value = reward + gamma * V[ni, nj]
                        action_values[action] = value
                if action_values:
                    best_action = max(action_values, key=action_values.get)
                    V_new[i, j] = action_values[best_action]
                    policy[i, j] = best_action
    if np.allclose(V, V_new):
        print(f'Converged at epoch {epoch}')
        break
    V = V_new
    if epoch % 5 == 0 or epoch == 1:
        plt.figure(figsize=(10, 7))
        plt.imshow(V, cmap='gray')
        plt.title(f'Value Function at Epoch {epoch}')
        plt.colorbar()
        plt.show()

print("Final Learned Policy:")
for i in range(rows):
    for j in range(cols):
        if map_data[i, j] >= 0:
            print(policy[i, j] if policy[i, j] else ' ', end=' ')
        else:
            print('X', end=' ')
    print()
```