Exploring Gridworld Using REINFORCE Algorithm

John Sunny George
Aerospace Engineering
University of Illinois at Urbana-Champaign
Champaign, USA

***Abstract*—A policy to traverse a grid, with deterministic and stochastic state transition, for maximizing the reward was done using the REINFORCE [1] algorithm. The policy network was trained using Adam and Stochastic Gradient Descent (SGD) algorithms. Adam provided better policies when the state transition was deterministic, while policies were similar for Adam and SGD when it was stochastic.**

I. Introduction

A. Dynamics of a Gridworld

The grid consists of 25 distinct locations corresponding to 25 states, represented by integers from 0 to 24. In each state, four actions are available, move right, move up, move left, and move down, represented by the integers 0, 1, 2, and 3, respectively. For the "easy" version of the grid, there are two teleporting states, denoted by integers 1 and 3, where any action teleports the agent to states 21 and 13, respectively. Also, actions that move the agent out of the grid will not lead to a state transition, and all other state transitions are deterministic. An episode terminates when the 100 steps are taken.

B. Algorithm

REINFORCE is a policy gradient algorithm that aims to optimize the policy directly without using a value function. The policy is parametrized with a neural network, which, when given a state s_t, outputs the probability, $\pi_\theta(a_t|s_t)$, of the agent selecting action a_t. The objective of the agent is to maximize the expected return $J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}(R(\tau))$, where $R(\tau)$ is the finite horizon undiscounted return. For this, an episode τ is generated using policy π_θ and the gradient of the objective function J_θ is estimated, which is given as (using policy gradient theorem):

$$\nabla_\theta J(\theta) = \sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) R(\tau) \tag{1}$$

The gradients are then used to update the parameters of the policy network as follows:

$$\theta = \theta + \alpha \nabla_\theta J(\theta) \tag{2}$$

where α is the learning rate. Since the objective is maximization, a gradient ascent is performed. However, during implementation, gradient descent is performed on the negative of the objective function

C. Reinforcement Learning Objective

The reward signal of the environment is described as follows:

- For any action taken in state, $s = 1$ the agent receives a reward of 10 and transitions to state, $s = 21$.
- For any action taken in state, $s = 3$ the agent receives a reward of 5 and transitions to state, $s = 13$.
- Any attempt by the agent to move out of the grid is penalized by a reward of -1.
- In all other cases, the agent receives no reward.

The agent achieves the objective by interacting with the environment through the actions sampled from the output of the policy network to maximize future rewards.

II. Problem Approach

A. Neural Network Architecture

A fully connected neural network with one hidden layer, having 100 units, was used for the policy network. A $\tanh$ activation function was used for the hidden layer, with a softmax function at the output layer. The input vector is one-dimensional, corresponding to the location on the grid. The output vector is four-dimensional, with each element corresponding to the probability of taking that action denoted by the output vector index.

B. Training Details

Unless specified, all experiments were run for 50000 episodes, each containing 100 training steps with a discount factor (γ) of 1.0. Either stochastic gradient descent (SGD) optimizer with no momentum or Adam optimizer with a learning rate of 0.001 was used to update the policy network.

TABLE I shows the list of all hyperparameters used in the experiments.

C. Implementation

The code was written in Python and PyTorch. The best-performing neural network based on mean episodic returns is stored during training, and the results are obtained using it.

III. Results

A. REINFORCE with SGD

Fig. 1 shows the REINFORCE algorithm's learning curve using the stochastic gradient descent as the optimizer for 50000 episodes. Fig. 2 shows the policy obtained from the best-performing policy network where the length of the arrow

TABLE I
HYPERPARAMETERS AND THEIR VALUES.

Hyperparameter	Value
Discount factor	1.0
Learning rate	0.001

is proportional to the probability of the action. Fig. 3 shows a policy sampled from the policy network overlayed over an optimum policy obtained using value iteration. Fig. 4 shows a sample trajectory where the policy keeps guiding the agent back to the teleporting state 1, where the maximum reward is available.

In Fig. 3, grid locations with only a red arrow indicate the policy coincide. Though the policy does not coincide in states 5, 10, 15, and 20, the optimal policy is stochastic, and hence both moving up and moving right are optimal. However, in states 4, 8, and 9, the action obtained from the network is not optimal, especially in state 4, where the recommended action moves the agent out of the grid. For states 8 and 9, the policy leads to convergence to sub-optimal solutions as it guides the agent to the second teleporting state 3, where the reward is comparatively less than state 1. This is likely due to a lack of exploration, as once a sufficiently good policy guides the agent to state 1, it keeps bringing the agent back to it, preventing further exploration.

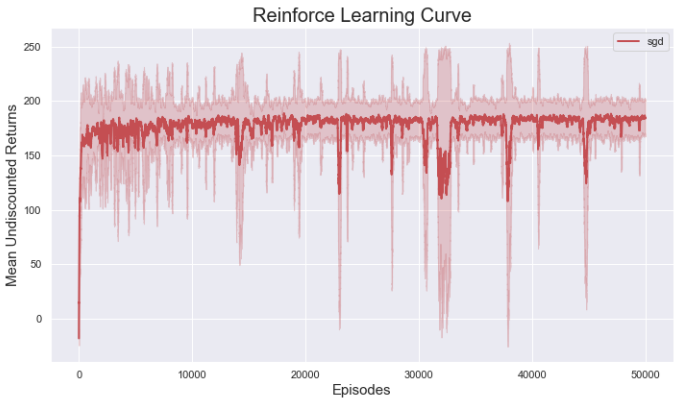

Fig. 1. REINFORCE learning curve. The plot shows a running average of over 100 episodic returns. Shaded bands indicate one standard deviation over 100 episodes. For the first 100 episodes, the mean and standard deviation at n-th episode is computed from all episodes $\leq n$.

B. REINFORCE with Adam

Fig. 5 shows the REINFORCE algorithm's learning curve using Adam and SGD as the optimizer for 50000 episodes. Though both optimizers could generate similar rewards, Adam was more stable, as shown by the variance bands.

Plots similar to the ones obtained for SGD are shown in Fig. 6, Fig. 7, and Fig. 8. One noticeable difference compared to SGD is that in state 4, the most probable action is to move left, which is the optimal action shown in Fig. 7, while it

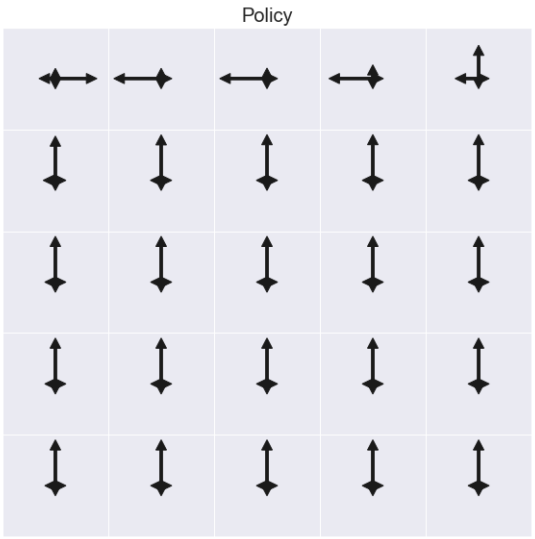

Fig. 2. Policy obtained using SGD. The length of the arrow is proportional to the probability of the action.

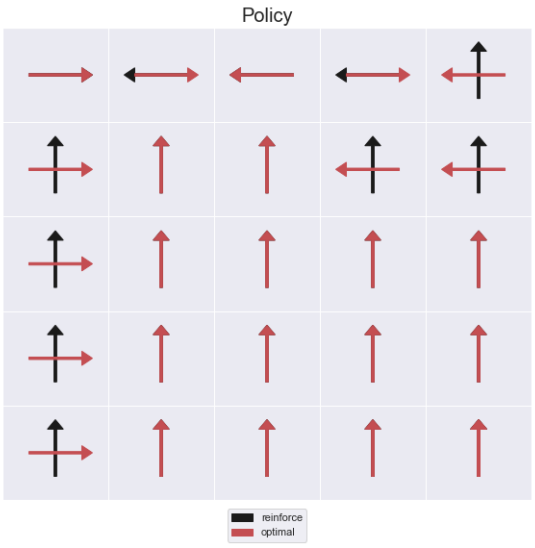

Fig. 3. Policy sampled from policy network trained using SGD overlayed on the optimal policy obtained using value iteration.

was to move out of the grid in SGD. This is likely because Adam uses a moving average of the gradients to update the network parameters, while SGD without momentum does not. The average gradients will help retain information in the early episodes where exploration was stronger, as later episodes might lack exploration once the policy is sufficiently good.

However, for states 8 and 9, the policy is still sub-optimal and will lead to suboptimal trajectories, as shown in Fig. 9, where the agent never discovers state 1.

C. REINFORCE with stochastic state transition

Similar experiments were done for the "hard" version of the grid world, where the state transition is stochastic. Fig 10

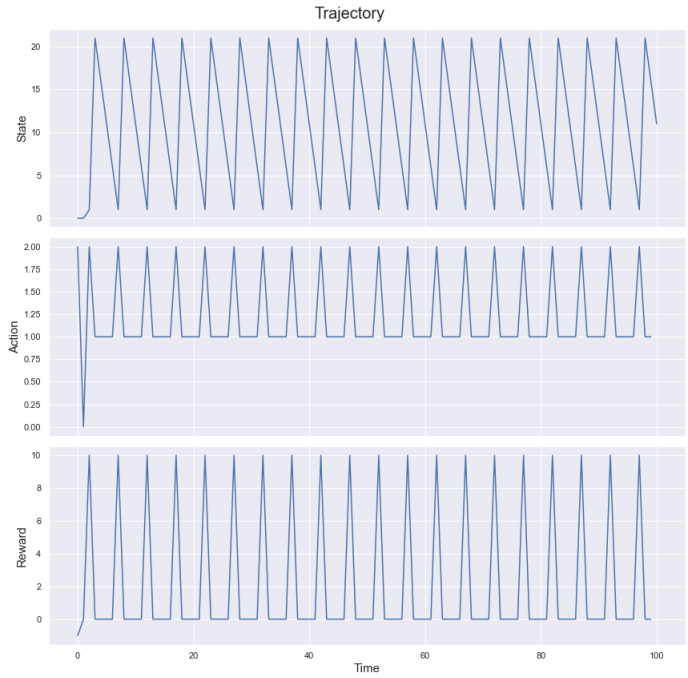

Fig. 4. A sample trajectory using SGD.

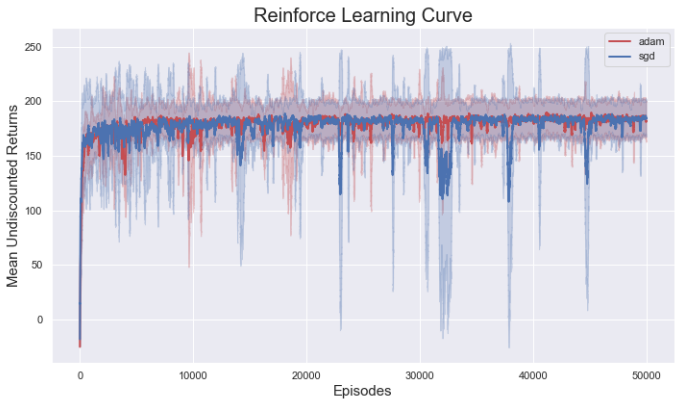

Fig. 5. REINFORCE learning curve for Adam and SGD optimizer.

shows the learning curve using Adam and SGD optimizer. As expected, the cumulative reward is less than the "easy" version since inherent stochasticity will delay the agent from reaching the teleporting state. Fig. 11 and Fig. 12 show the policy obtained using Adam and SGD optimizers, respectively, and are similar. Unlike deterministic transition, the policy obtained using SGD provides the optimal action in state 4 for the stochastic transition. This is likely because of further exploration facilitated through inherent stochasticity in the environment.

Fig. 13 shows a sample trajectory obtained using the Adam optimizer.

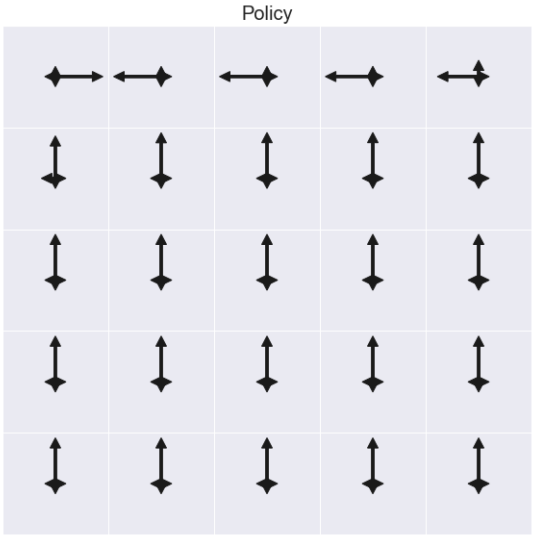

Fig. 6. Policy obtained using Adam. The length of the arrow is proportional to the probability of the action.

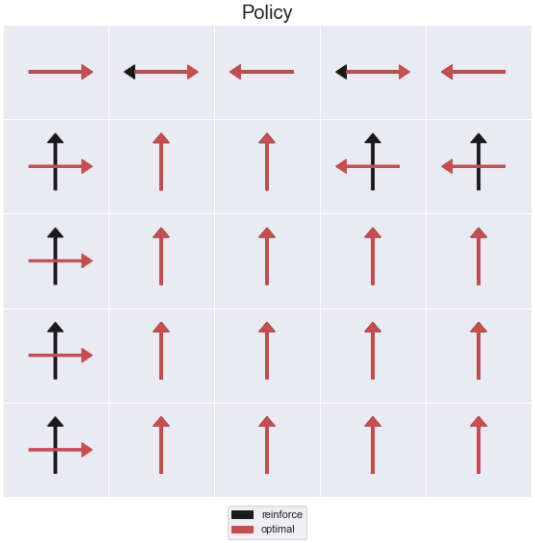

Fig. 7. Policy sampled from policy network trained using Adam overlayed on the optimal policy obtained using value iteration.

IV. Conclusions

REINFORCE algorithm is slow to converge and can suffer from a lack of exploration, as it only updates the policy based on the observed returns. This can lead to a policy stuck in a suboptimal local maximum. Both Adam and SGD optimizer successfully trained the policy network; however, Adam was more stable and helpful in compensating for the REINFORCE algorithm's lack of exploration when the environment's state transition was deterministic.

The figures can be found in GitHub Repository inside the folders test4, test5, and test6, with 50000 episodes and 100 units.

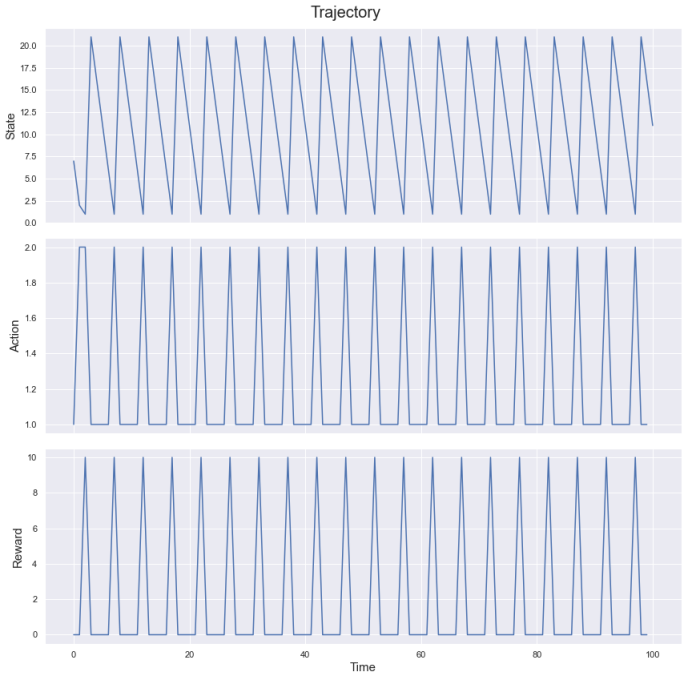

Fig. 8. A sample trajectory using Adam optimizer.

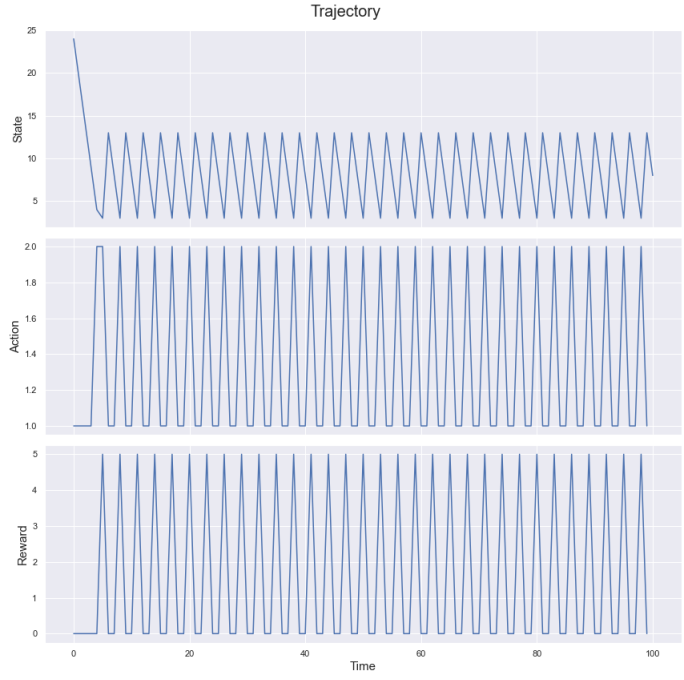

Fig. 9. A suboptimal trajectory where the agent keeps visiting the second teleporting state without ever discovering state 1.

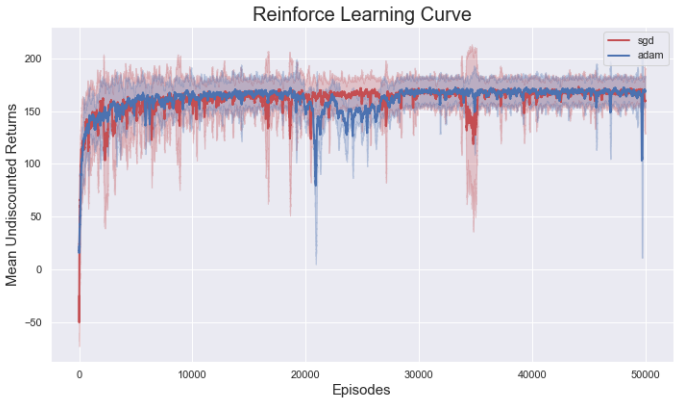

Fig. 10. REINFORCE learning curve for stochastic state transition using Adam and SGD optimizer.

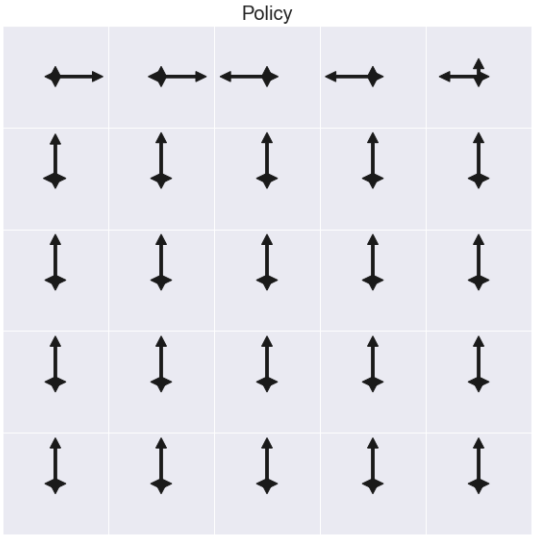

Fig. 11. Policy using Adam.

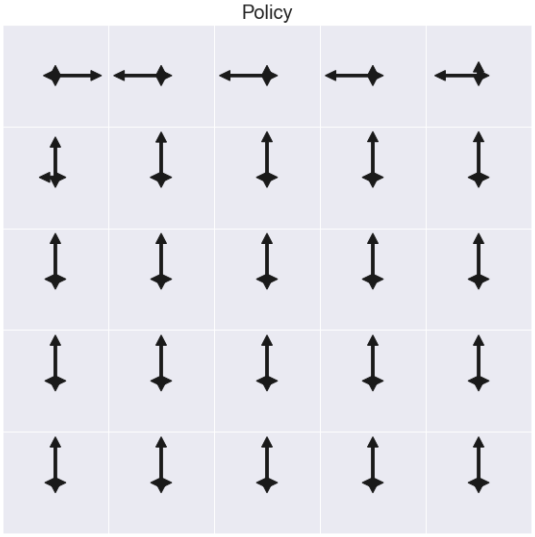

Fig. 12. Policy using SGD.

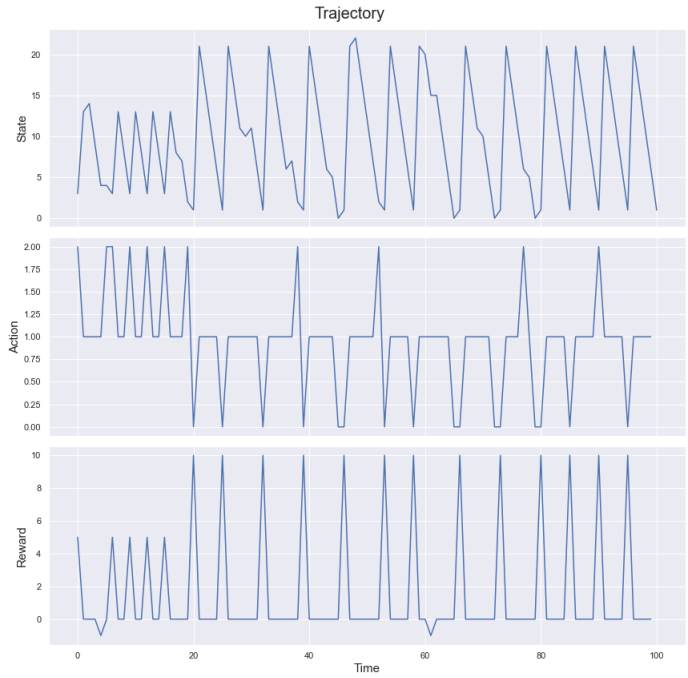

Fig. 13. A sample trajectory using Adam optimizer.

REFERENCES

[1] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, May 1992.