Brad Rafferty
brjrafferty@gmail.com

Reinforcement Learning Implementation Summary

To learn the policy in example.csv, a Sarsa algorithm is implemented. The implementation follows the structure as outlined in Algorithm 5.3 of *Decision Making Under Uncertainty*, but with the Sarsa update. Sarsa was chosen for its model-free nature (especially in problems where states are missing) and its relative simplicity compared to Q-Learning (in that a max over next possible actions is not required).

The learning rate α is updated by counting the number of times a state-action pair is encountered, i.e. $\alpha = 1/N(s,a)$. To help drive the state-action values $Q(s,a)$ closer to convergence, the algorithm is wrapped in a while loop to allow "episodes" or "epochs". Each episode iterates through all the data (experience tuples / rows) in the given dataset. The $Q(s,a)$ from the end of one episode is fed into the start of the next episode.

A gridworld plot helps visualize that this approach indeed aids convergence. See Figure 1 for gridworld after three episodes. Although it is not an optimal solution (all grids show lead to the 10 reward and none go to 3), it is converged. A relatively simple extension of Sarsa to improve the scores of each policy would be implementing eligibility traces.

To determine the policy of unobserved or "hidden" states, i.e. those states missing from the data, the script applies the highest-scoring action of the nearest observed state to the hidden state. In other words, the policy of a given hidden state is chosen to be the policy of the nearest observed state. This is far from optimal, especially in larger problems where most states are likely unobserved.

example.csv		
No. Episodes	*Time per Episode*	*Total Time*
3	1.0668	3.2004

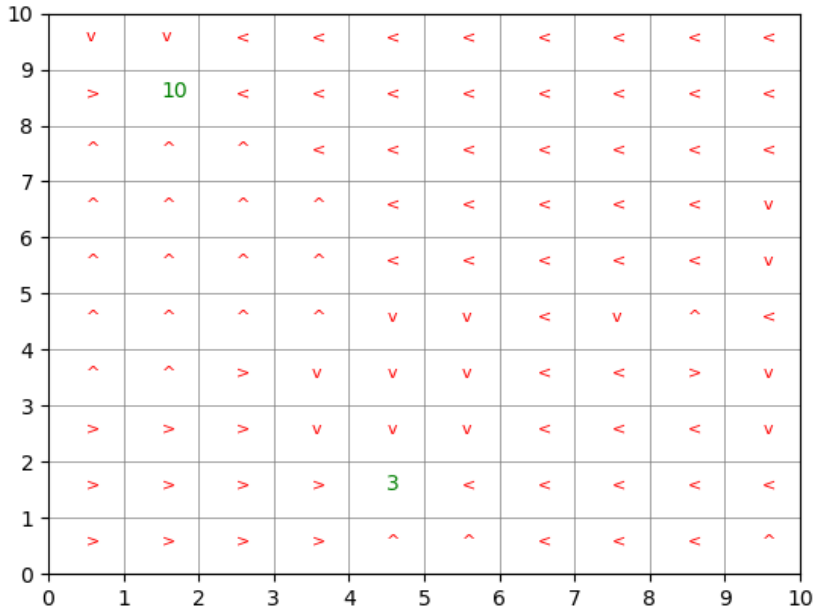

Figure 1. Gridworld policy after three episodes