Q learning

Roger Garcia
Mississippi State University

Reinforcement Learning (Model-Free)

- Off-Policy vs On-Policy
  - Function that maps states to actions
  - Probability of an action given a state

$$\pi : S \rightarrow A$$

- MDP?
  - A set of states
  - A transition model **Q(state,action) = R(state, action) + g*max[Q(next_s, all_actions)]**
  - A reward function
  - Acquire optimal policy [after convergence]
  - Off-Policy: Must actually try actions and states out to learn !

$$a = \pi(s)$$

- Value iteration
  - Utility value [Q value]= Q(s,a) for every action at every state
  - Utility of state *s?*
    - maxQ value over all possible actions at that state
  - Utilities of actions?
    - Progressively improve utilities of actions

$$a_t = \pi(s_t) = argmax_a Q(s_t, a)$$

Q learning

- Q learning implementation using Q-Table
  - Specifies the value of each action for each possible state
  - Helps map from values received from the environment to Q-Table indices.
- There are 4 possible actions: [up, down, left, right]
- Objective: To move from starting position to the green goal position
- Search Policy
  - e-Greedy; the robot chooses its next action as the best action it knows at the time
  - Look at the Q-Value for each action at the current state
- **DEMO TIME** (10x10 GridWorld)(https://github.com/aalind0/RL-Game-Bot)
  - Act randomly in search of reward
  - Remember the reward we get for being in state and taking Act randomly in search of a reward
  - The reward for the previous state and action is a combination of the current immediate reward of moving forward and also the best possible reward of any action from the current state, the future reward.
  - Prefer paths which result in positive rewards and avoid negative ones

The Q-Learning Algorithm

- Parameters
- Initialize
- For each World.iteration:
  - Select rand initial state
  - While not at Goal:
    - Select 1 among all actions at curr_state
    - Valid action? Consider transitioning to next state
    - Get maxQ value for this next state based on all possible actions
    - Compute Q(s,action) and update
    - Set the next state as the curr_state
  - End
- End

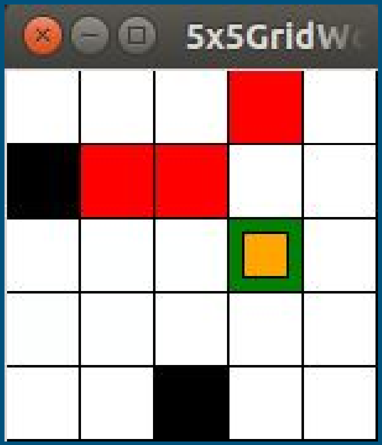
Learning Rate
Vs
Discount Factor

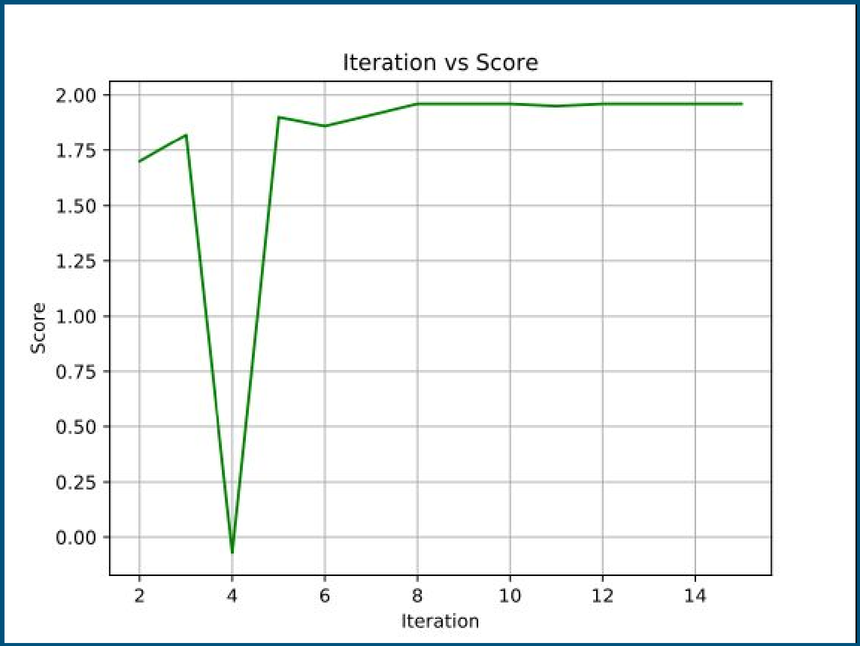
Q Learning Simulation Runs (Grid World)

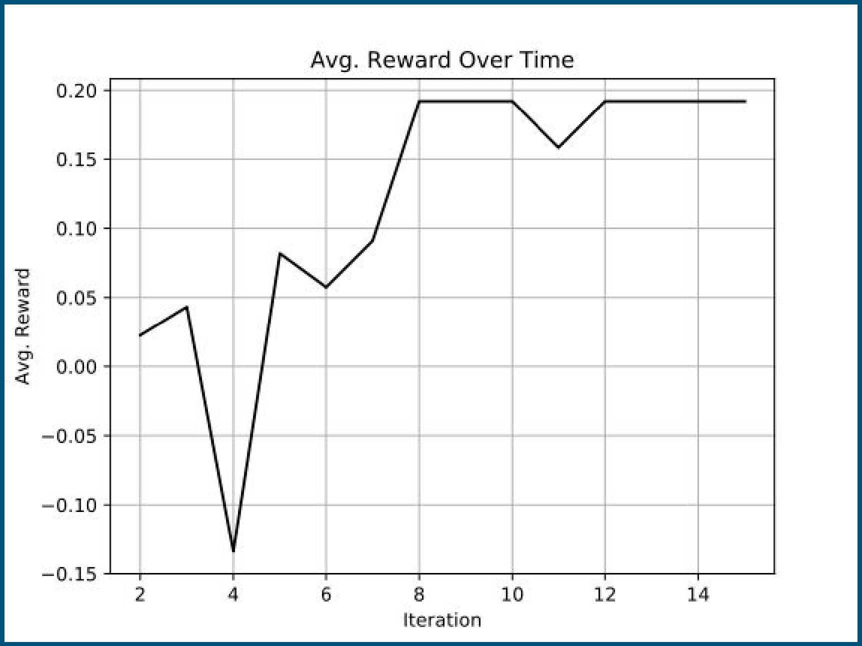
- Discount Factor (Gamma): **0.3**
- Initial Score, alpha : **1.0**
- Walk Reward : **- 0.01**
- Red Square (you die, you restart) : **-1.0**
- *Goal Square* : **+ 1.0**
- Actions: LEFT, DOWN, RIGHT, UP
- Policy ?
  - Explore environment ? random actions
  - Use Greedy Policy to evaluate states
  - Update gather information, [progressively reduce randomness]
- Highest Reward = Action to take
  - Take random action uniformly
  - If enough World iterations are completed, each action will be tried an infinite number of times, thus ensuring optimal actions to be discovered

Goal:
Maximize
Sum of Rewards

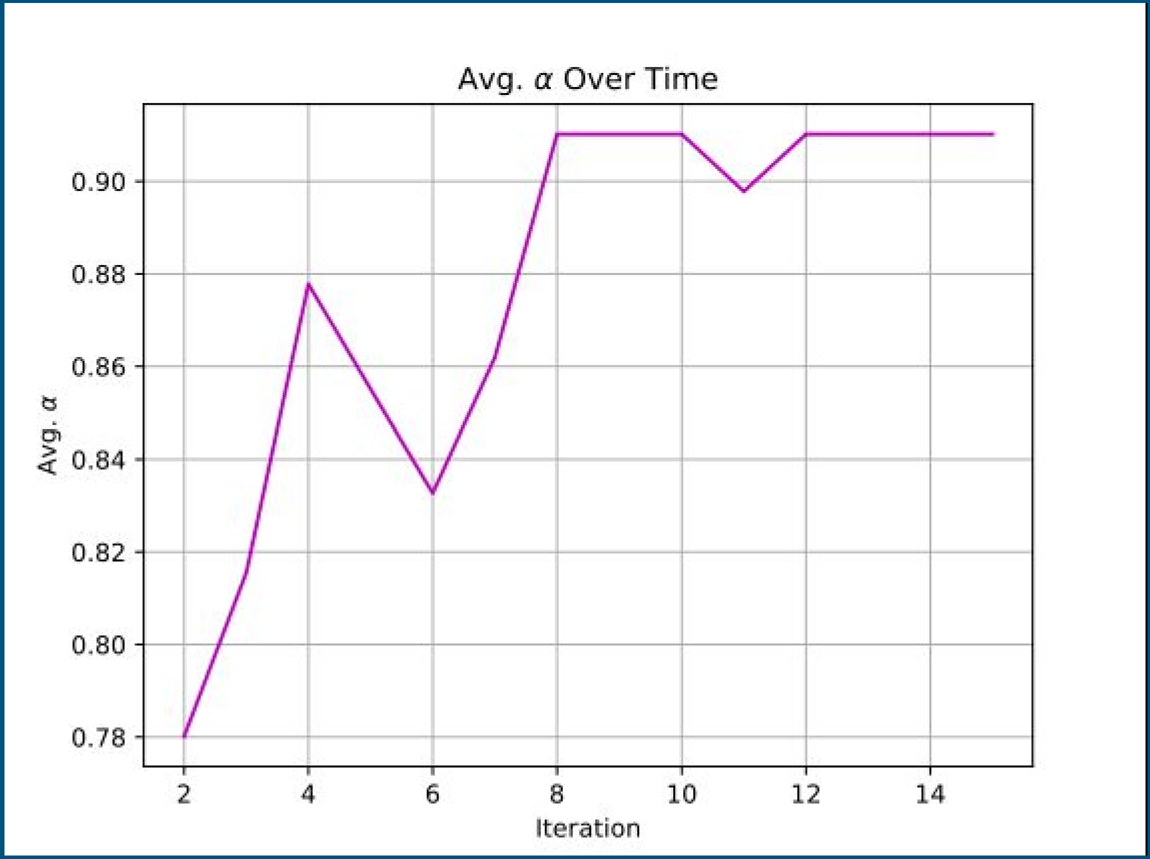
5x5 Grid World

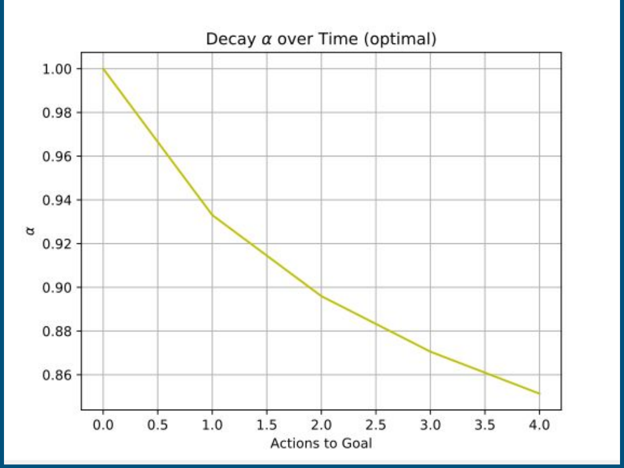


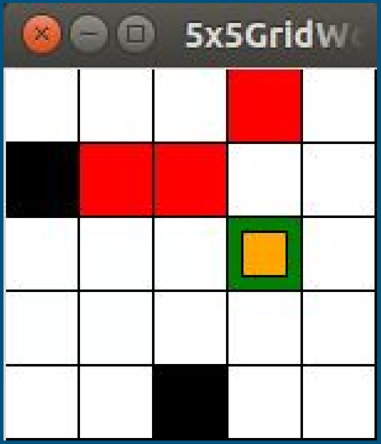




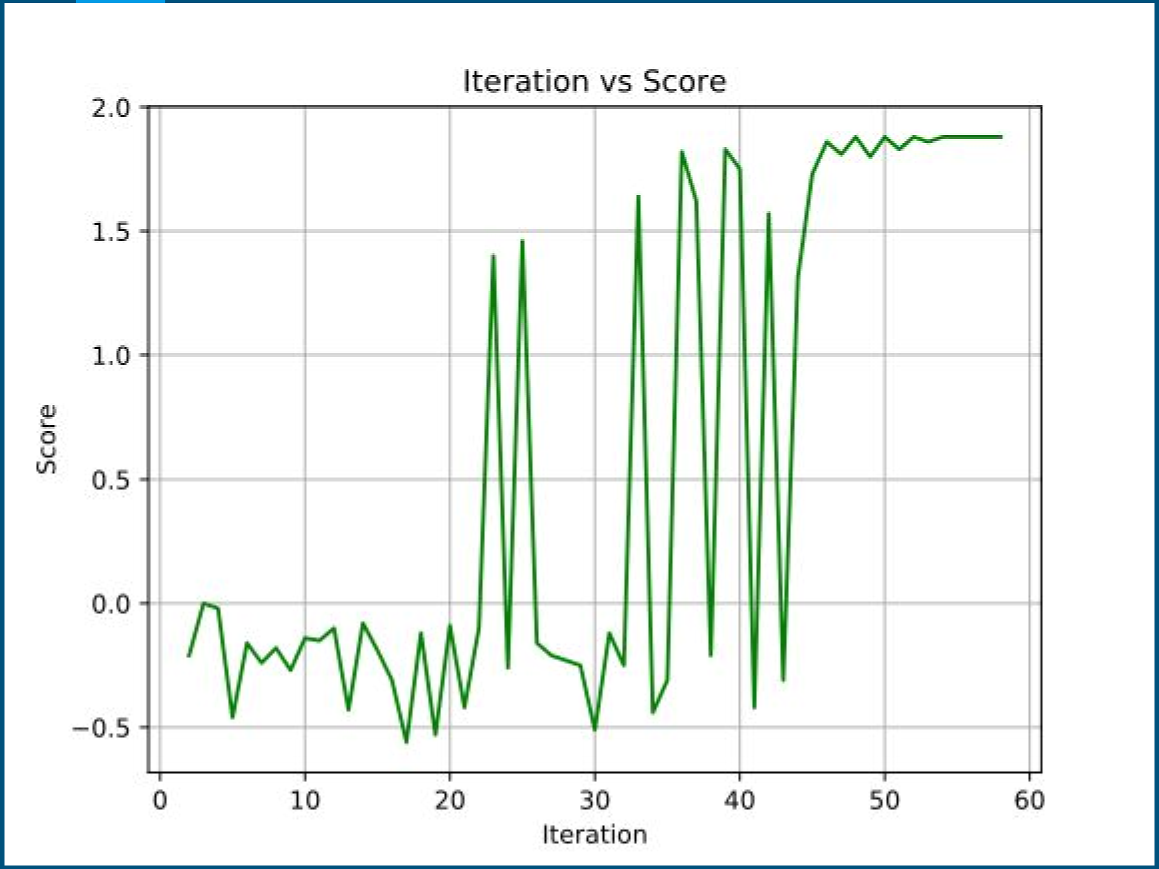
5x5 Grid World

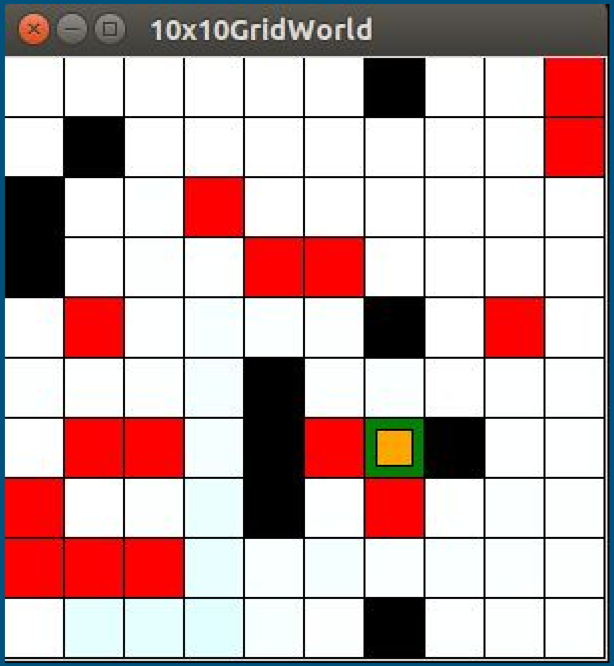




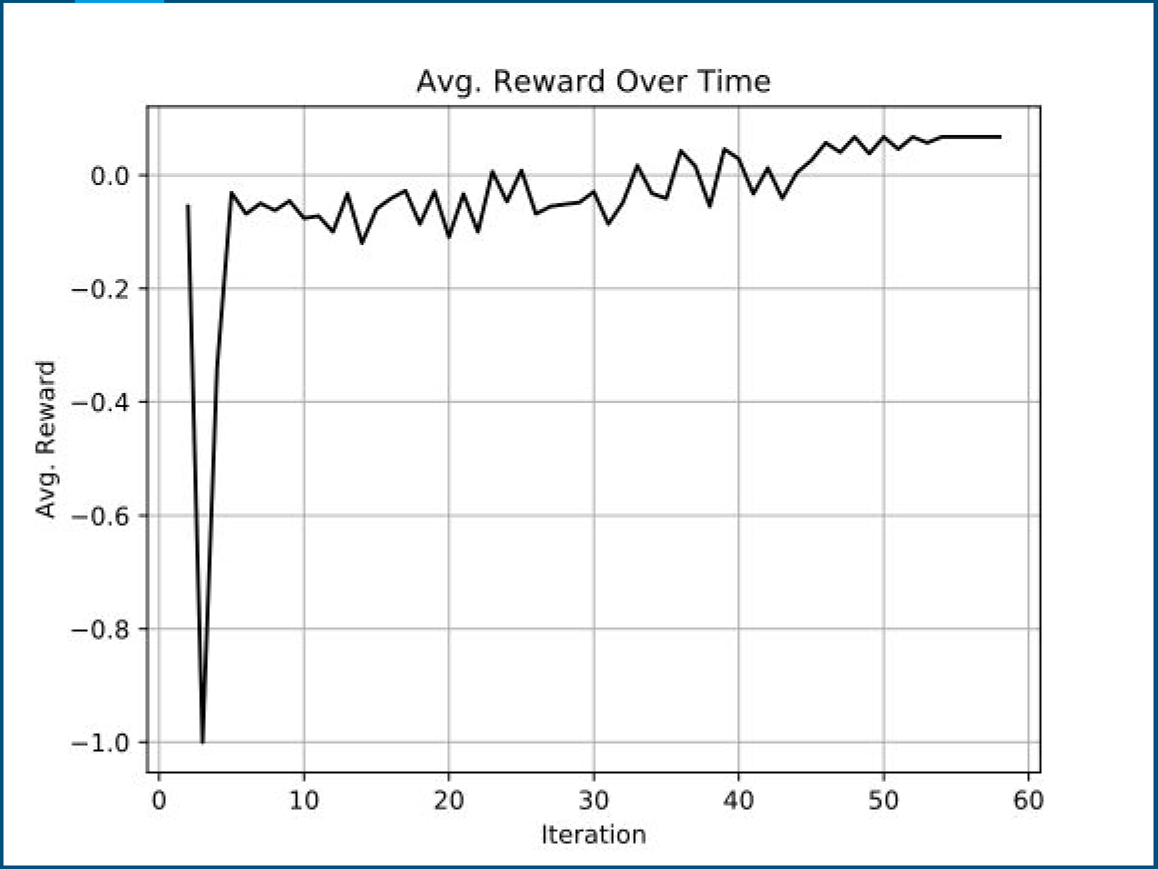


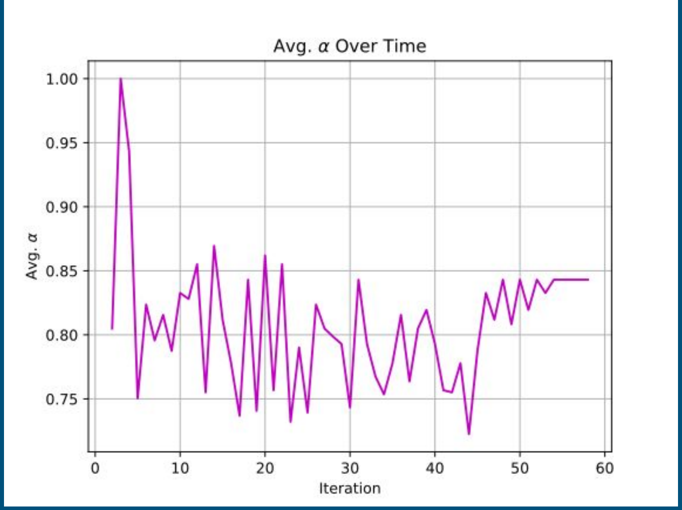
10x10 Grid World

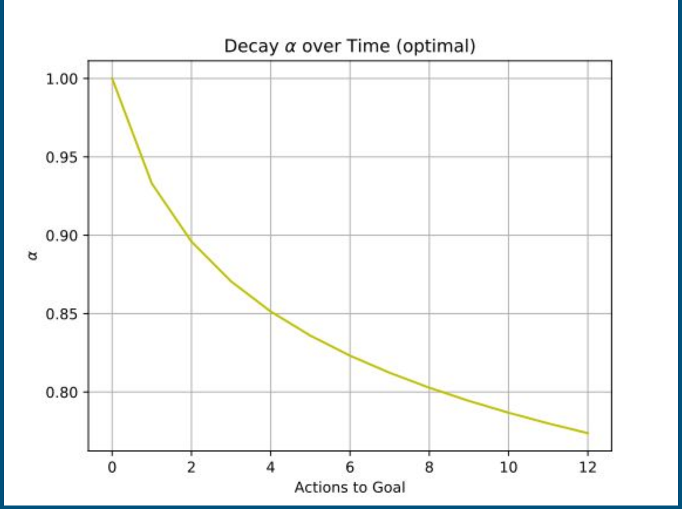




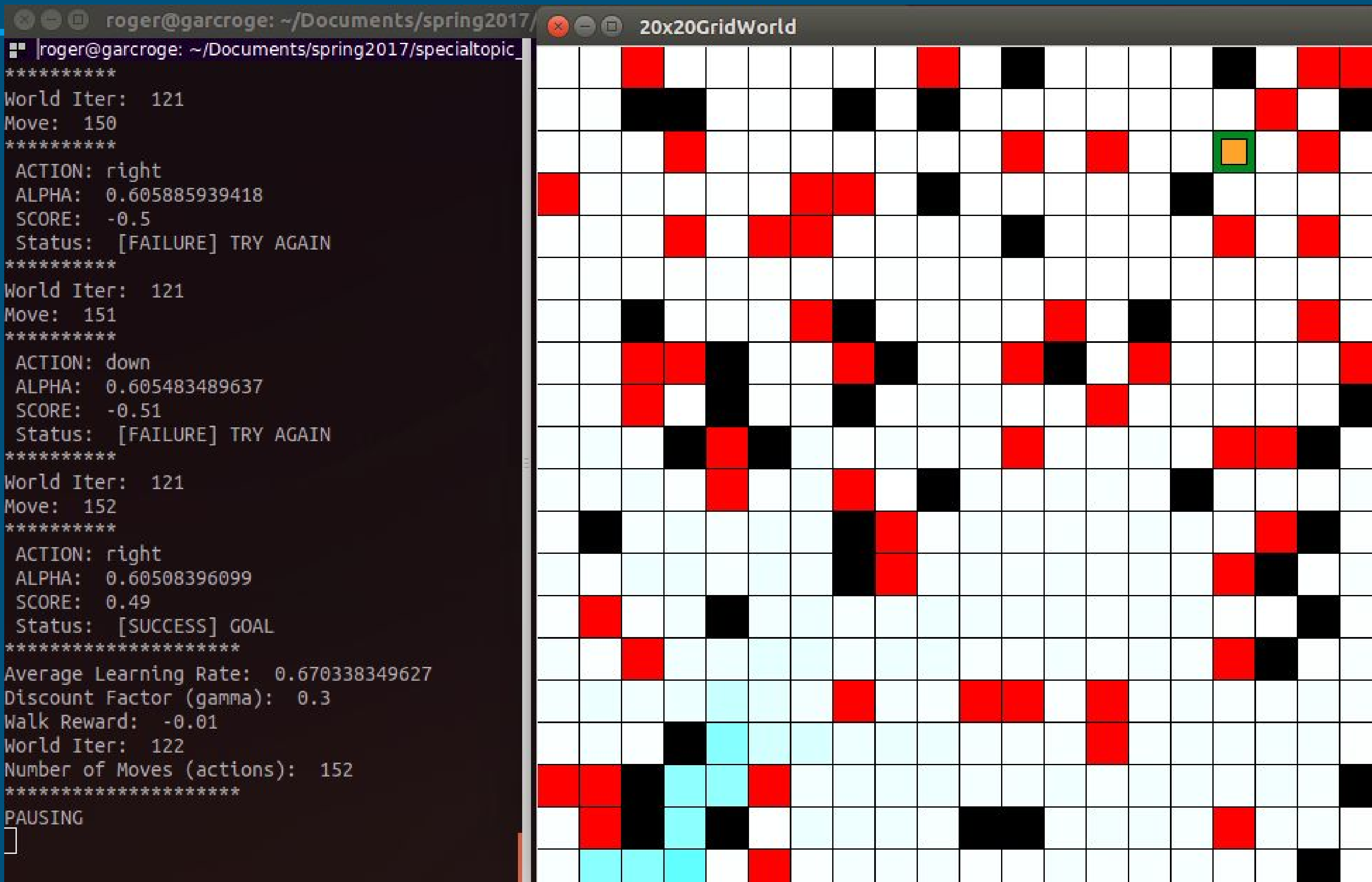
10x10 Grid World



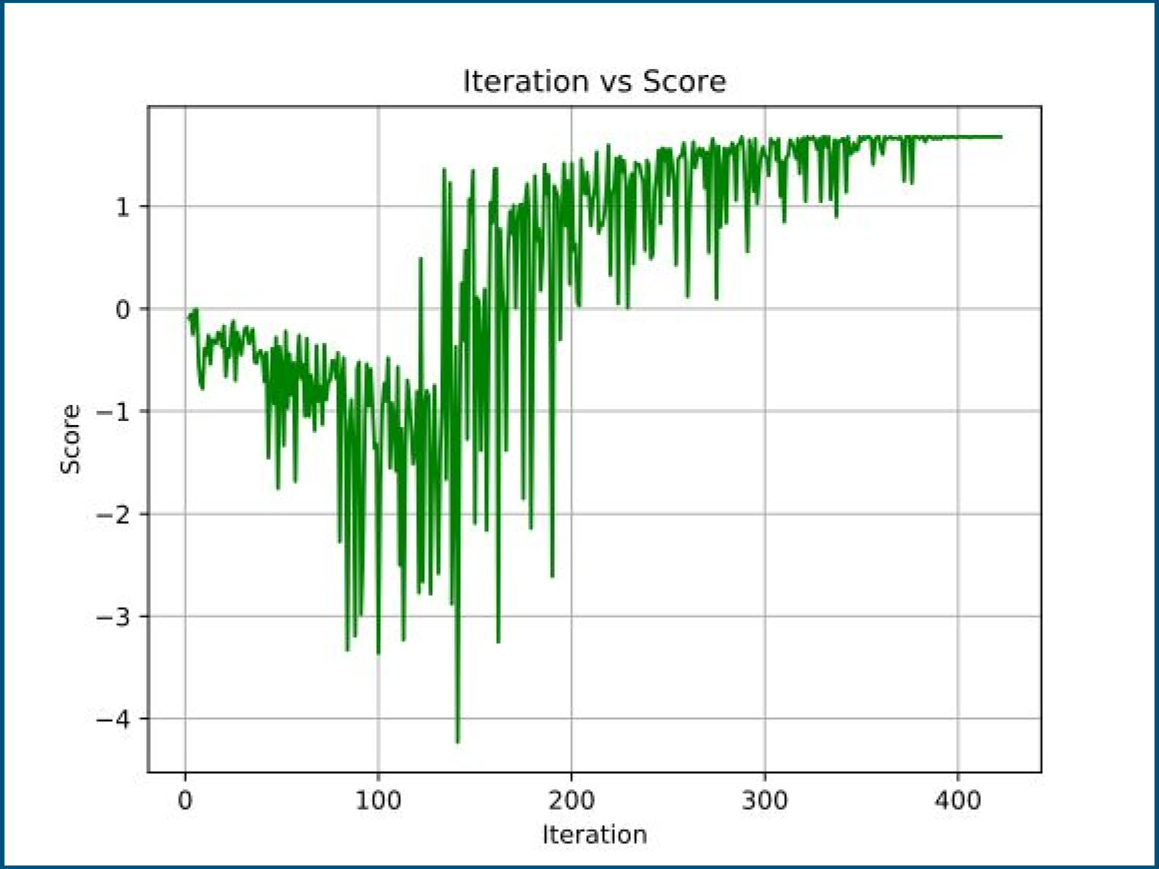




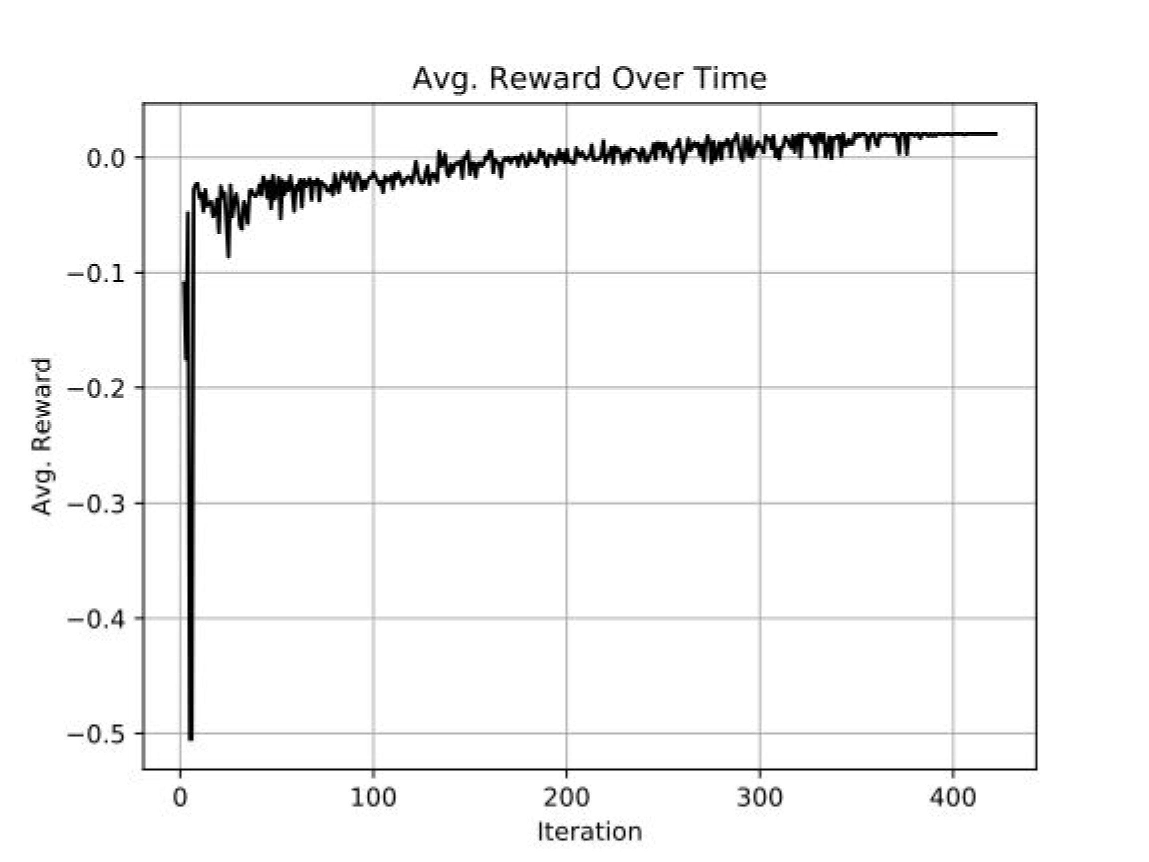
20x20 Grid World

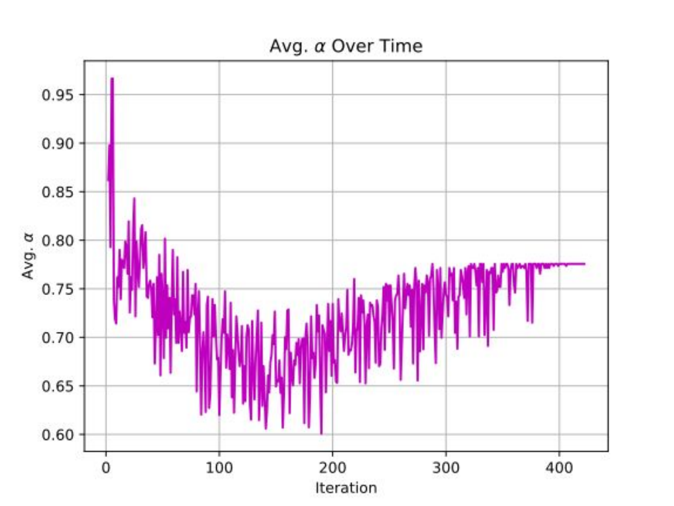


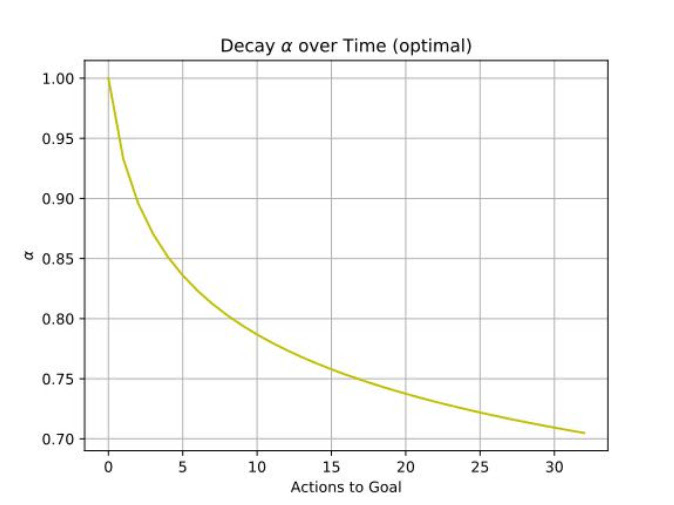
20x20 Grid World



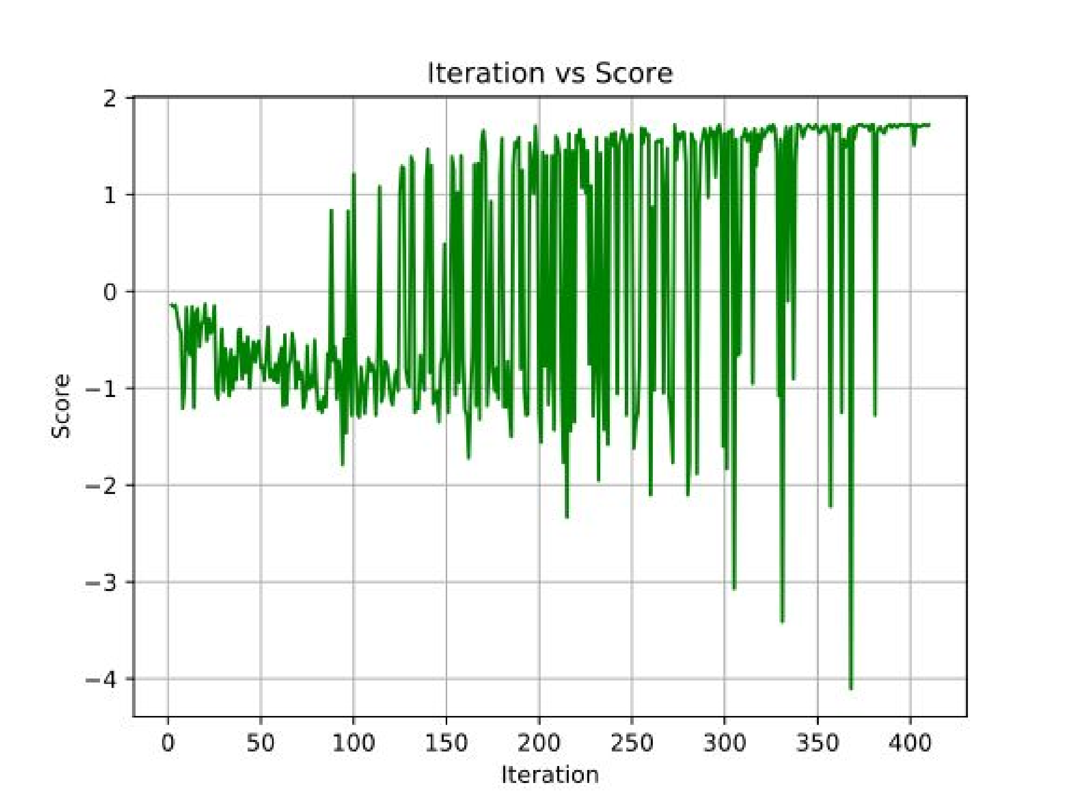
20x20 Grid World

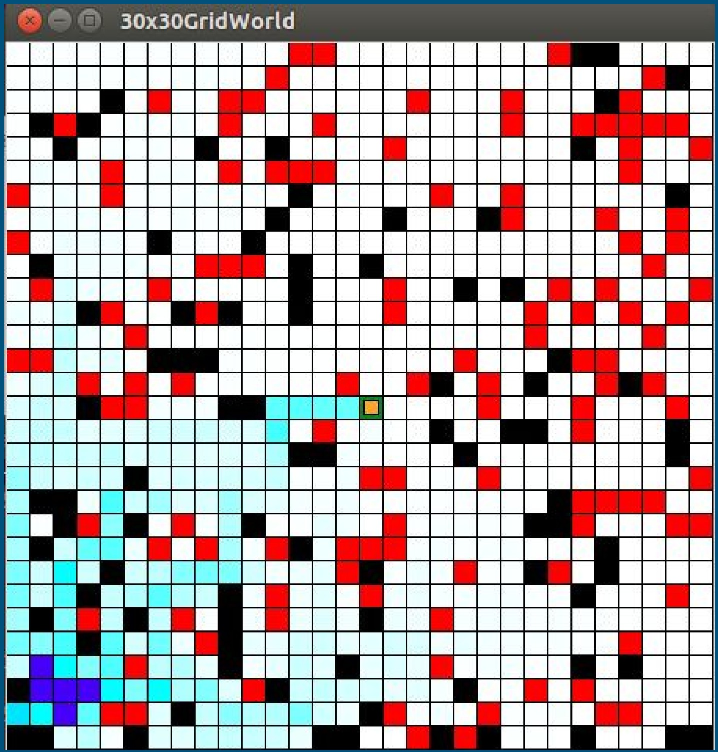




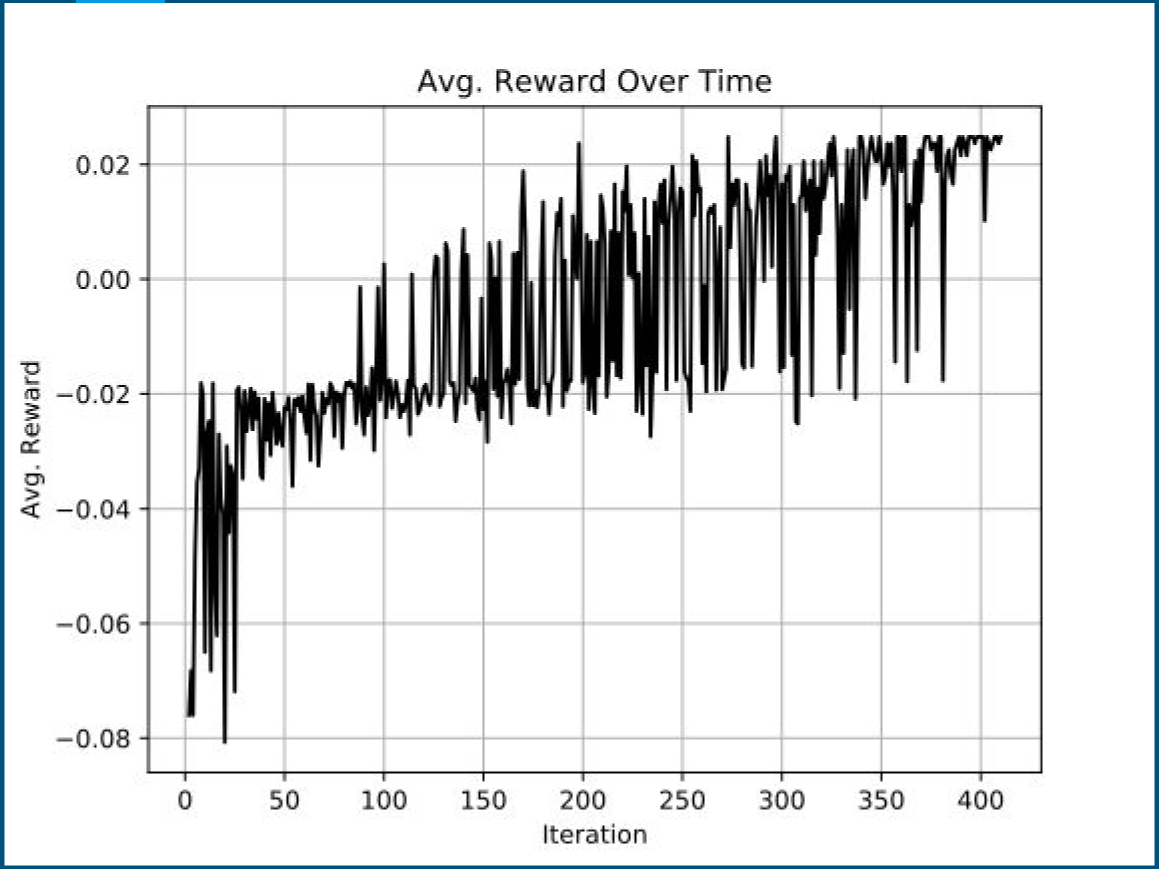


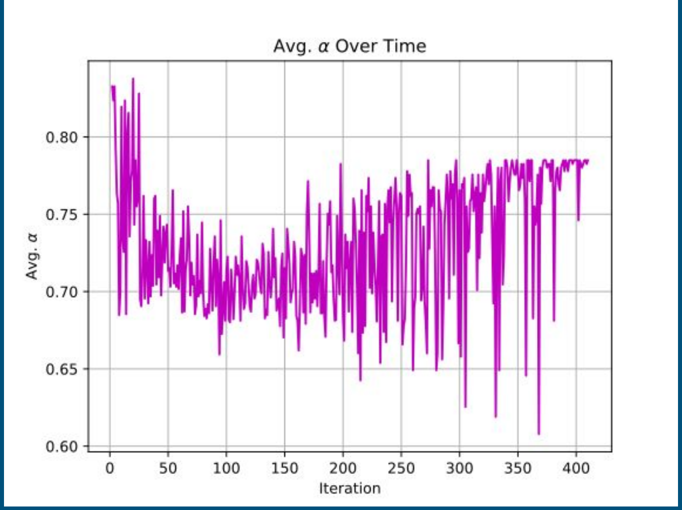
30x30 Grid World

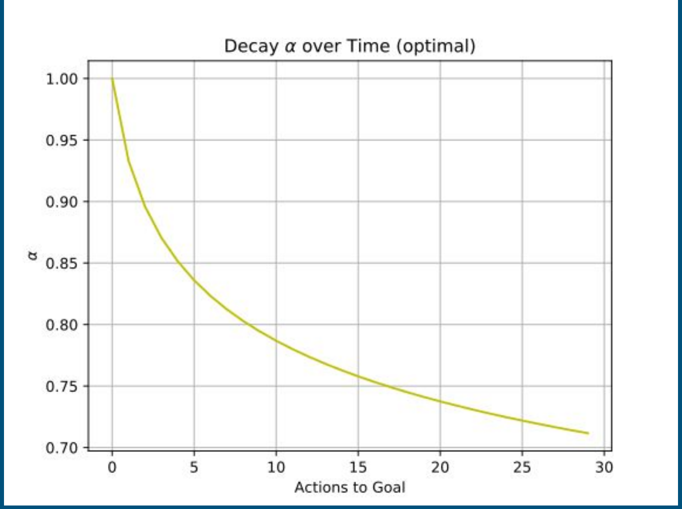




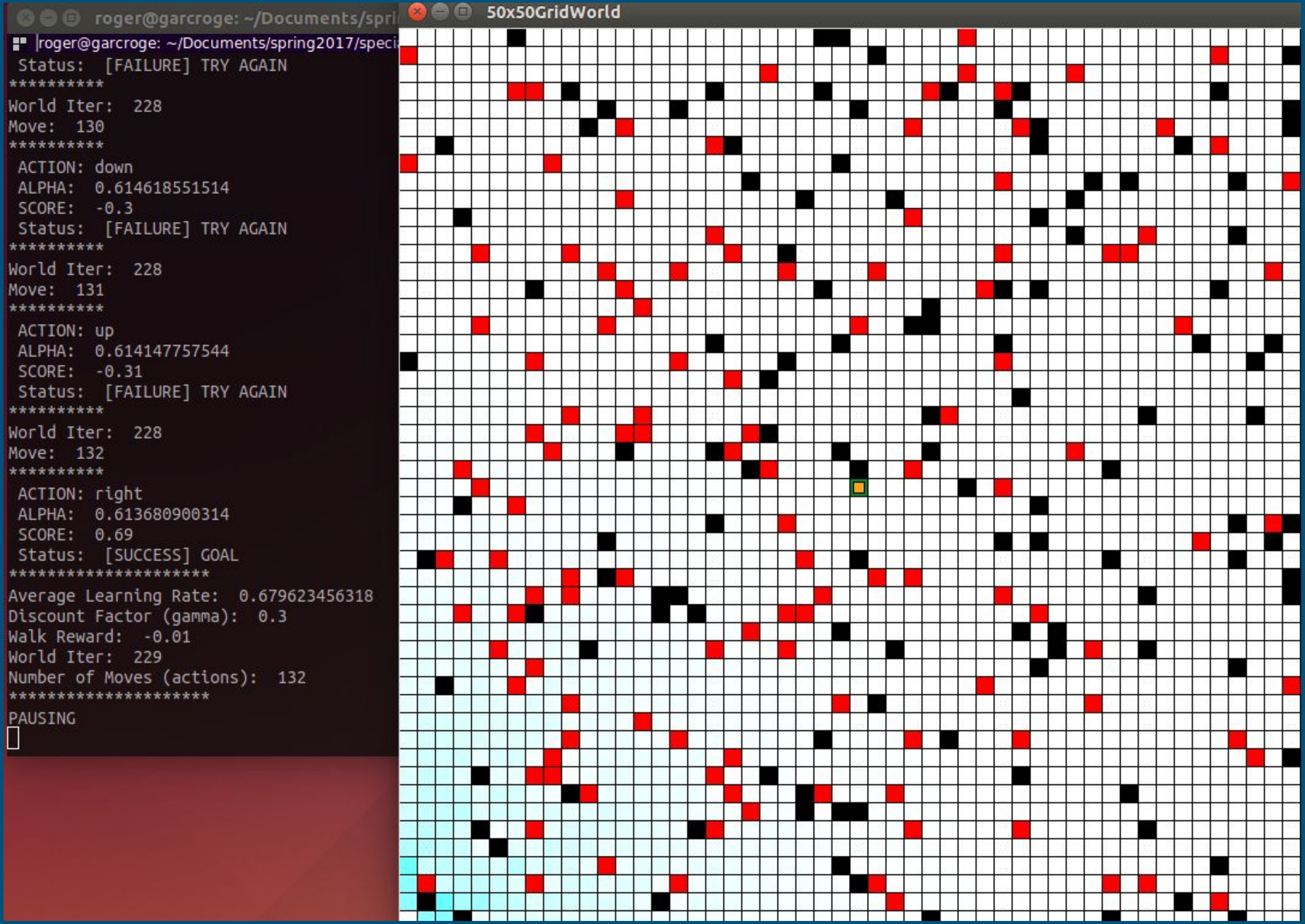
30x30 Grid World



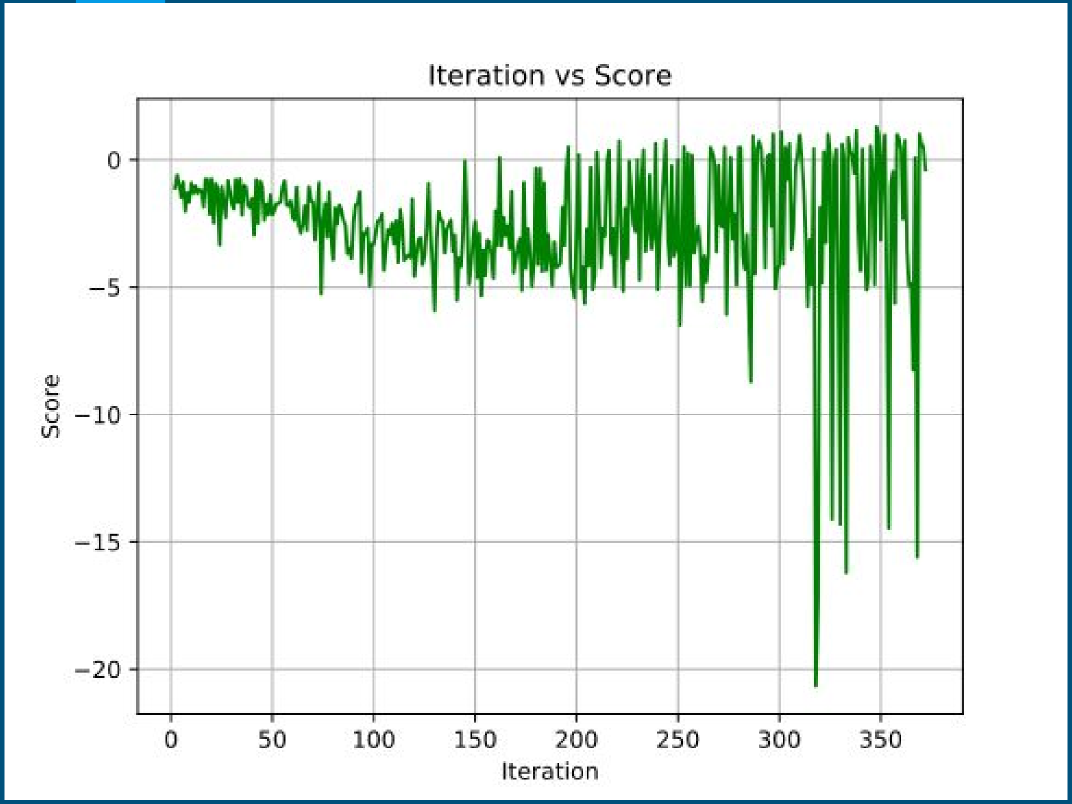


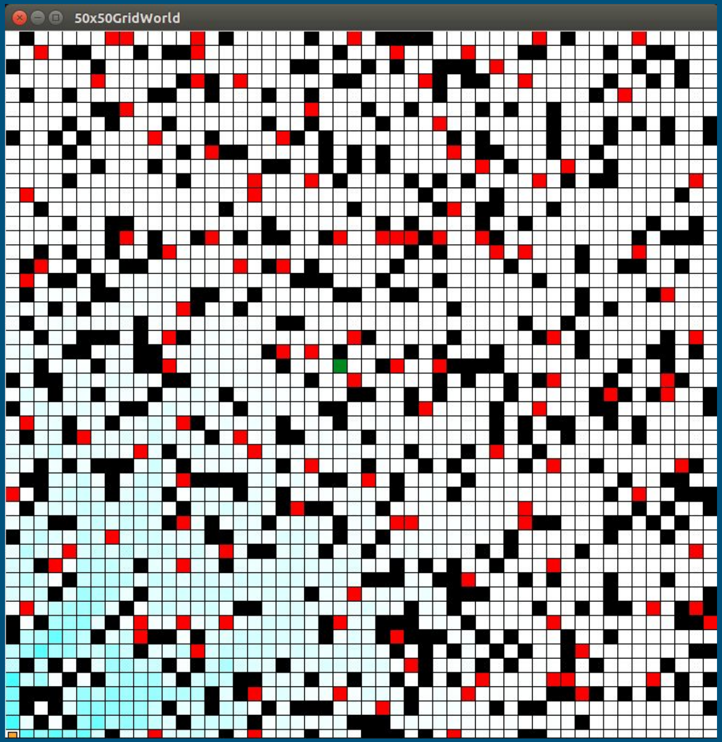


50x50 Grid World

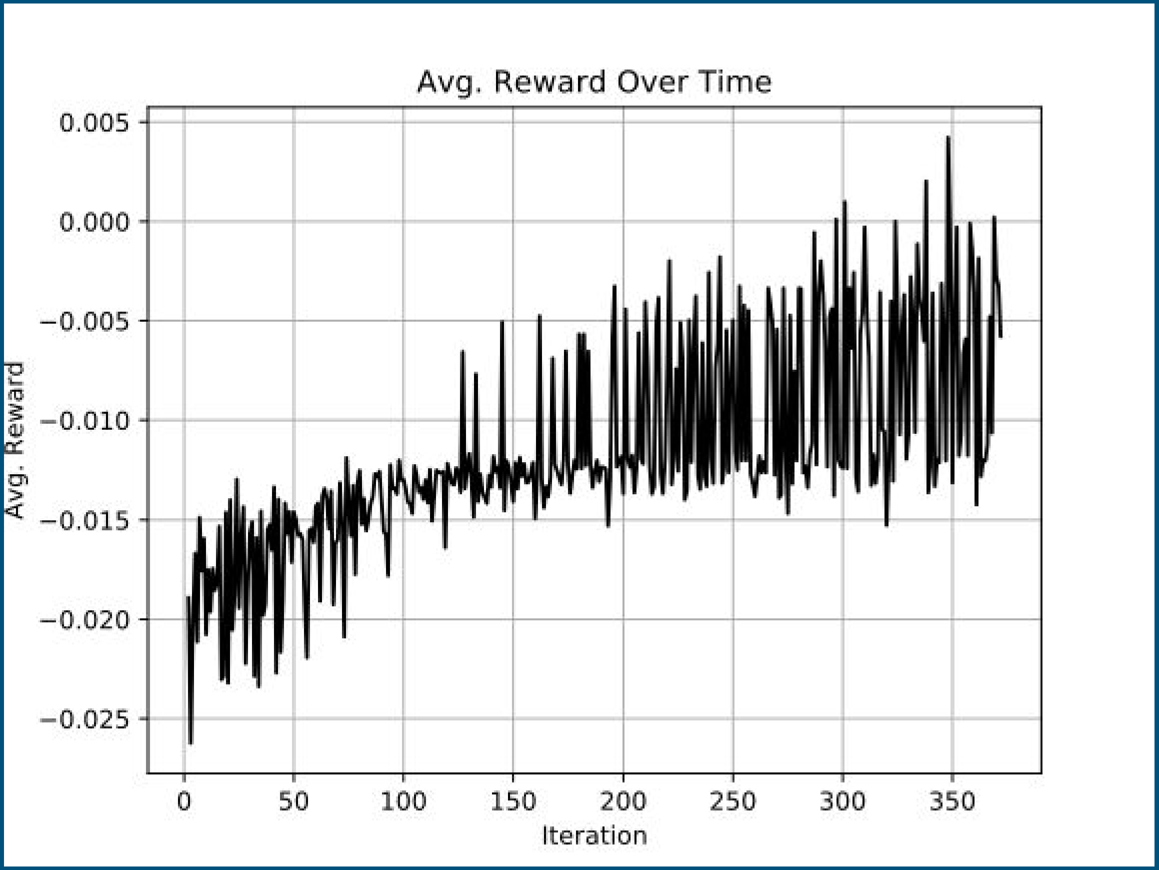


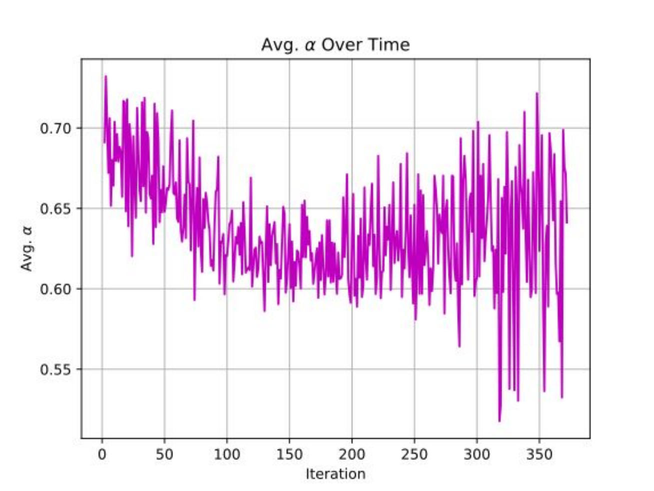
50x50 Grid World

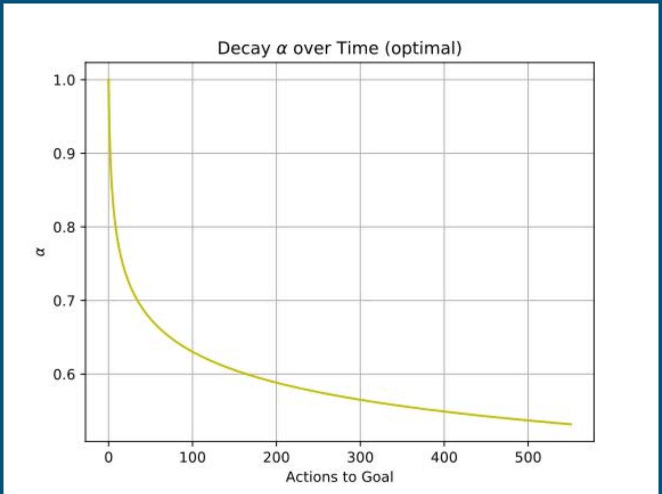




50x50 Grid World







Conclusion

- Updating policy *always* even when exploring far from Goal
- Greedy Policy used (with exploration)
  - e-soft
  - Softmax (rank or weight)
- Future work
  - Explore tradeoff between policies, to produce best results overall
  - Impact of policy to learning
  - On-Policy vs Off-Policy
- References

  - http://mnemstudio.org/path-finding-q-learning-tutorial.htm
  - https://github.com/aalind0/RL-Game-Bot
  - http://mbb-team.github.io/VBA-toolbox/wiki/Fast-demo-Q-learning-model/
  - http://www.cse.unsw.edu.au/~cs9417ml/RL1/tdlearning.html#aselection