Reinforcement Learning

Project #2 Report

Class and Section: CPE471.1001

Name: Lucas Pinto

Due Date: February 28, 2025

GitHub Repository: https://github.com/PieFlavr/CPE471-Project-2

World-Agent Design

The default environment is modeled by a 5x5 “Grid World”, with an “Agent” that can do *‘up’, ‘down’, ‘right’ and ‘left’* actions. Any invalid actions such as attempting to move off the grid can be attempted by the Agent, but they do not let the Agent out. The Agent and Grid World are implemented in the `agent.py` and `grid_world.py` file respectively.

Agent Rewards

A reward based on the grid, (*grid_length*grid_width*), continues to work successfully for training. Similarly, punishments are always (-1) for every step and (-5) for attempting to move out of bounds.

With newly implemented decaying epsilon and softmax action selection, the Agent no longer performs invalid moves once convergent (at least in most cases). This even applies to incredibly large grid sizes, although certain algorithms such as RBF have a particularly difficult time on grid sizes larger than 25x25.

Learning Algorithms

There are now a large variety of learning algorithms currently implemented in this project. Previously known Q-Table variables for this project were generalized to a “weights” variable instead of being strictly a “Q-table”, as now there exists multiple algorithms with different weight storage methods. Additionally there is now more than one selection type, although they are weight-type specific. For the most part, all of these were implemented in the `learning.py` file.

The following are all implemented algorithms and selection types:

ALGORITHM	COMPATIBLE SELECTION FUNCTIONS	WEIGHT TYPE
$Q - Learning$	*epsilon greedy Q* *decaying epsilon greedy Q* *softmax Q*	*Q-Table*
$Q(\lambda)$	*epsilon greedy Q* *decaying epsilon greedy Q* *softmax Q*	*Q-Table*
$RBF\,Q - Learning$	*softmax P selection*	*Weight Vector*
$FSR\,Q - Learning$	*softmax FSR selection*	*Weight Vector*

Of note with the above, RBF and FSR’s Weight Vectors are each formatted differently. Additionally, the *softmax* implementations have an optional “greedy-cutoff” where it switches to a full-greedy selection. This likely should have just been implemented per algorithm as opposed to the selection, but at the time it was easier to debug this way due to the different disambiguations of each weight type.

Additionally, the following table is a compilation of all the defaultly run algorithms in addition to their default selection types…

DEFAULT RUN	DEFAULT SELECTION	WEIGHT TYPE
$Q - Learning$	*softmax Q*	*Q-Table*
$Q(\lambda)$	*softmax Q*	*Q-Table*
$4RBF\ Q - Learning$	*softmax P selection*	*Weight Vector*
$9RBF\ Q - Learning$	*softmax P selection*	*Weight Vector*
$FSR\ Q - Learning$	*softmax FSR selection*	*Weight Vector*

With the newly refactored algorithm selection in the code, you can now set a *global parameters* dictionary for all algorithm functions to run by default, a *local parameters* dictionary which overrides each particular argument passed from *global*, and a *function dictionary* that decides which functions to run and how.

There is also an optional $nRBF\ Q - Learning$ *algorithm* which automatically generates an appropriate number of RBF as evenly as possible throughout any given grid configuration.

Report Data Parameters

For this report, the training data for the 5x5 Grid World was produced with the following parameters (applied to all algorithms if applicable):

α	γ	ϵ	τ	λ
0.15	0.9	0.05	0.5	0.5
Agent Start Position	Goal Position	Grid Size	Episodes	Full-Greedy Cut-off
$(0,0)$	$(4,4)$	$5x5$	60	40

With these parameters, default runs should take no less than a few seconds to generate the data required for the program to make its figures.

Known Issues

There are a number of issues especially with the new algorithms and selection functions that are unresolved or fixed, but to the exact extent is unknown. Here they are as follows:

1) The implementation of $softmax$ is not “pure”, and required “stabilization” from numbers blowing up or erroring due to floating point precision errors. This involved a “stabilization” step of the Q-value by subtracting the max Q-value, which theoretically preserves the relative probabilities, but has not been rigorously confirmed.

2) **4-RBF** and **9-RBF** on grid sizes larger than $25x25$ appear to encounter stability issues, potentially experiencing progressive degradation of behavior but not infinite loop

3) **FSR** seems to also have specific edge case loop issues with the default settings, where it seemingly gets stuck near the starting position. This is likely due to floating point precision errors, although the cause is not entirely known yet. This has been somewhat fixed by forcing episodes above a certain step size to terminate early.

4) The stability of this particular RBF generation has led to quite unstable convergence even with full-greedy cutoff. However, the other RBF functions are particularly terrible at converging on grid sizes any larger than $10x10$, noticeable growing much worse each episode as opposed to properly converging. Still, this behavior can likely be attributed to bad RBF coverage and the fact σ remains constant in spite of larger grid sizes.

RESULTs

The results of the above runs are shown below, but more extensively additional images and .csv data were recorded in the `project_data` folder. More detail, figures, and data are stored in the submitted .zip file and also the GitHub project repository: https://github.com/PieFlavr/CPE471-Project-2

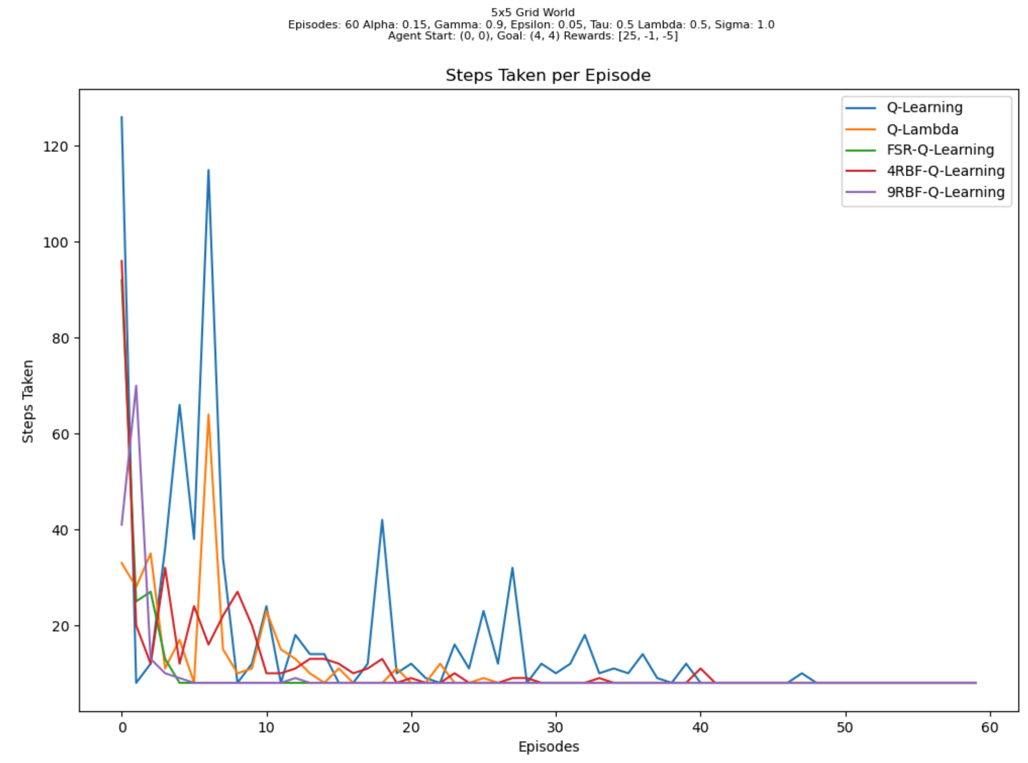
RBF Results Summary

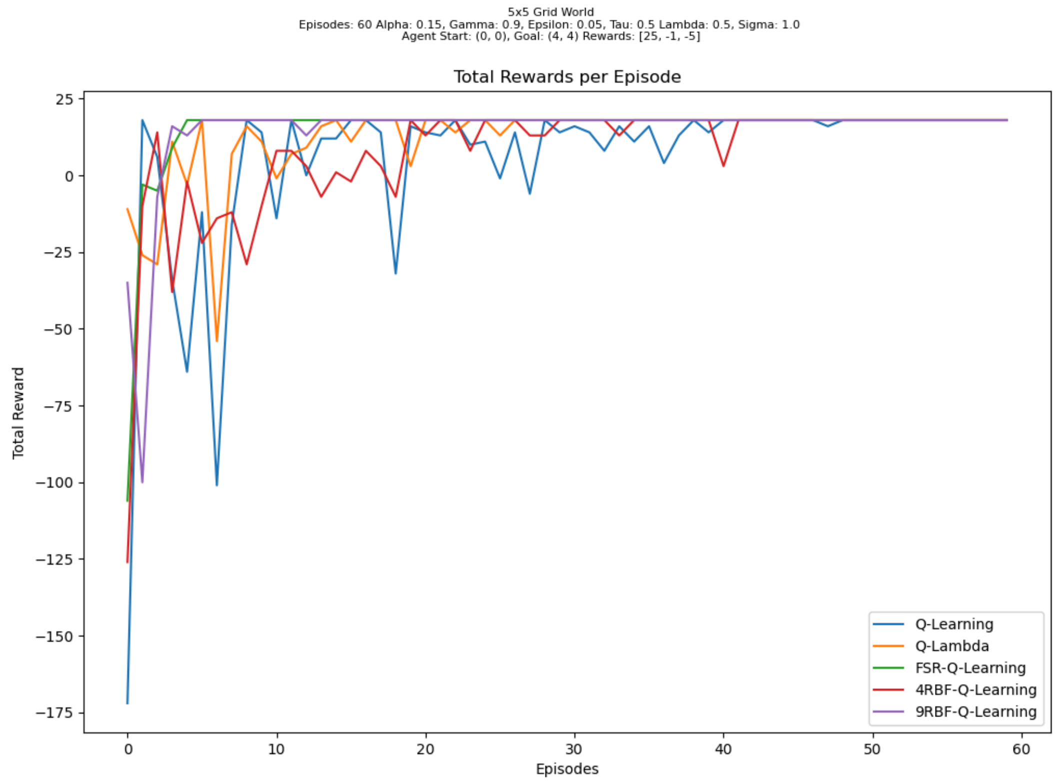
The RBF functions had the most unique convergence behaviors. When testing with larger grids for the same relative positions of RBF centers and sigma value, convergence simply just did not happen. However, if accounting for the additional distances with higher sigma or simply more RBFs, convergence returns. When it does converge it converges quicker than Q-Learning or Q-Lambda, although the more RBFs the faster it appears to converge, with 9 RBFs converging almost as quickly as FSR Q. The precise position of the RBF centers (algorithmically or manually) affect its convergence and the behaviors stemming from it, noticeably preferring “curved” paths that “dance” with the RB centers. Which, given my gaussian implementation of RBF, makes sense. RBFs appear to have the highest potential for weight compression of all the algorithms, simply due to encoding so few features. But as a result of saving space, it requires somewhat more complicated and/or expensive retrieval procedures as well as a more delicate feature selection via RBF placements.

FSR Results Summary

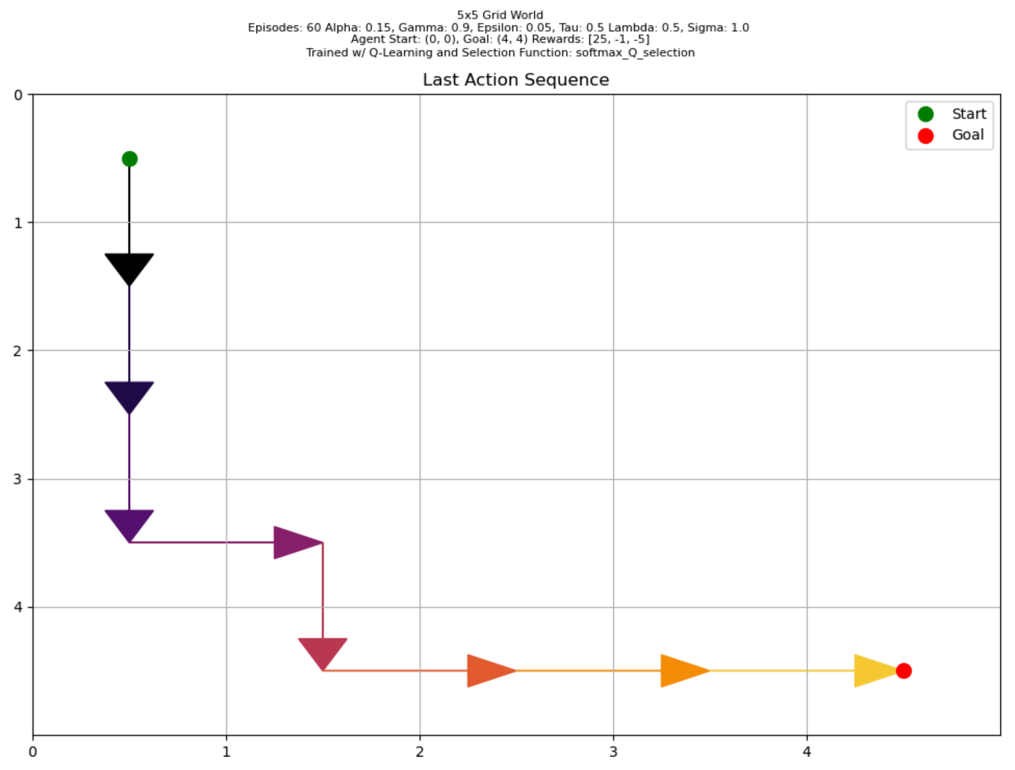
FSR initially had the most errant behavior, although that was only due to a mistake in the encoding interpretation. Barring that, FSR converges the fastest most consistently among all of the learning methods so far. It also tends to converge to using straight paths as opposed to obviously curved (RBFs), random (Q-learning), or broken up strides ($Q(\lambda)$). This is likely due to the fact that the features I decided to encode for are just the coordinates. As a result, reward propagates per row and column, naturally tending towards aligning to a particular axis for traversal. Due to the nature of the grid, being a space made of squares, this was expected. Still, considering how small the FSR encoding is, the fast convergence is incredible.

AGGREGATE DATA PLOTs

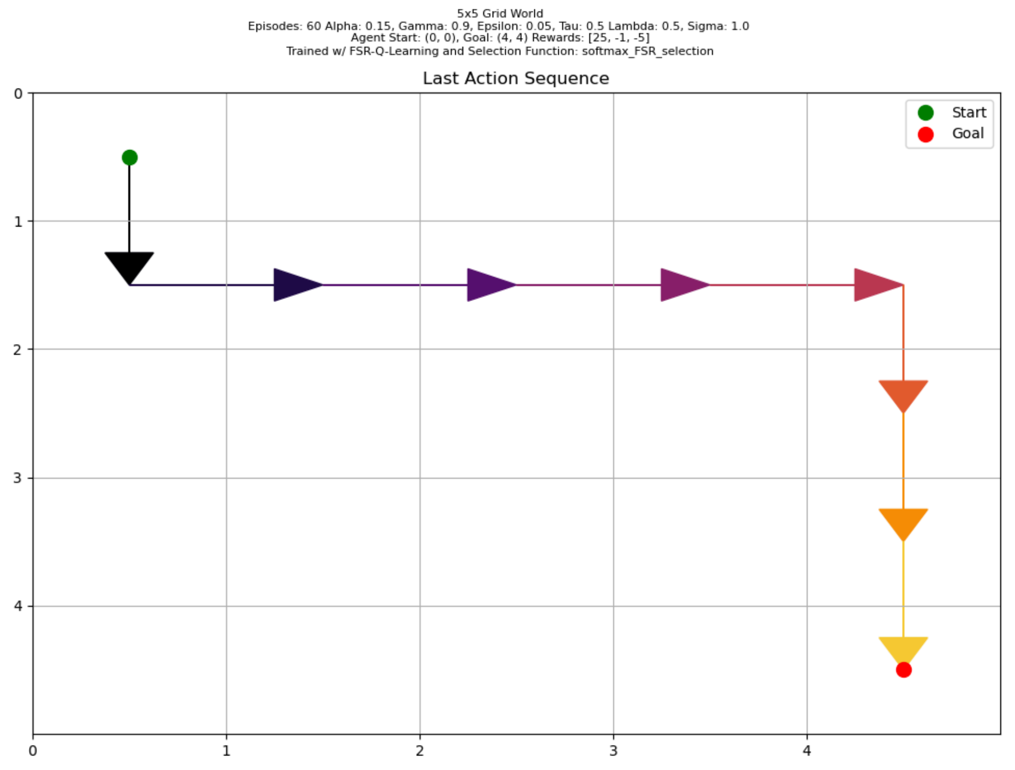




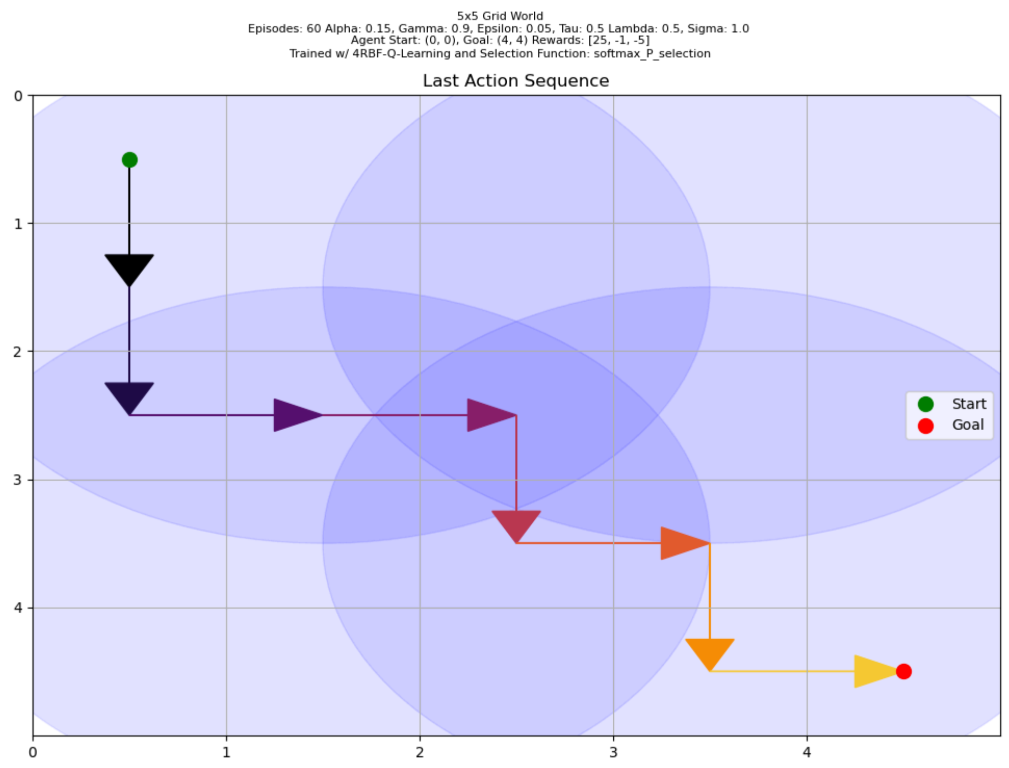
Final Action Sequences

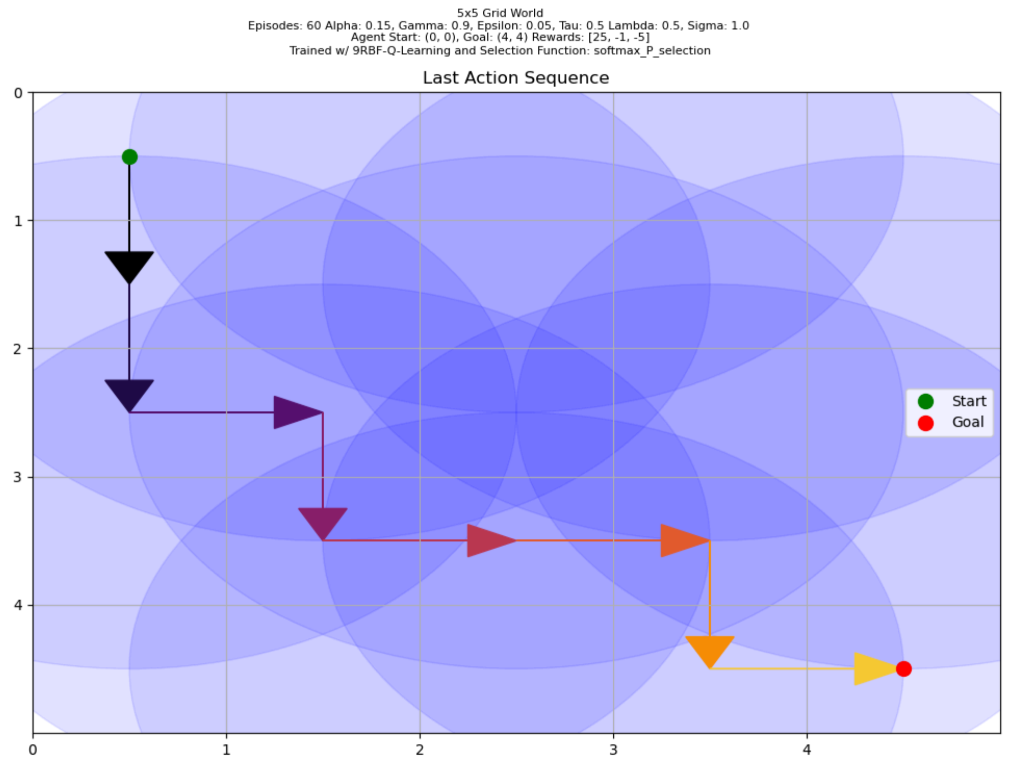


5x5 Grid World
Episodes: 60 Alpha: 0.15, Gamma: 0.9, Epsilon: 0.05, Tau: 0.5 Lambda: 0.5, Sigma: 1.0
Agent Start: (0, 0), Goal: (4, 4) Rewards: [25, -1, -5]
Trained w/ Q-Learning and Selection Function: softmax_Q_selection



5x5 Grid World
Episodes: 60 Alpha: 0.15, Gamma: 0.9, Epsilon: 0.05, Tau: 0.5 Lambda: 0.5, Sigma: 1.0
Agent Start: (0, 0), Goal: (4, 4) Rewards: [25, -1, -5]
Trained w/ FSR-Q-Learning and Selection Function: softmax_FSR_selection


5x5 Grid World
Episodes: 60 Alpha: 0.15, Gamma: 0.9, Epsilon: 0.05, Tau: 0.5 Lambda: 0.5, Sigma: 1.0
Agent Start: (0, 0), Goal: (4, 4) Rewards: [25, -1, -5]
Trained w/ 4RBF-Q-Learning and Selection Function: softmax_P_selection


5x5 Grid World
Episodes: 60 Alpha: 0.15, Gamma: 0.9, Epsilon: 0.05, Tau: 0.5 Lambda: 0.5, Sigma: 1.0
Agent Start: (0, 0), Goal: (4, 4) Rewards: [25, -1, -5]
Trained w/ 9RBF-Q-Learning and Selection Function: softmax_P_selection