**CONTROL**

    learning\_rate=0.8,

    gamma=0.95,

    epsilon=0.1,

A graph of a graph of a graph

AI-generated content may be incorrect.

**LOWERED LEARNING RATE**

    learning\_rate=0.6,

    gamma=0.95,

    epsilon=0.1,

A graph of a graph and a graph of a graph

AI-generated content may be incorrect.

**LOWERED GAMMA**

    learning\_rate=0.8,

    gamma=0.70,

    epsilon=0.1,

**A graph of a graph and a graph of a graph

AI-generated content may be incorrect.**

**INCREASED EPSILON**

    learning\_rate=0.8,

    gamma=0.95,

    epsilon=0.3,

**A graph of a graph showing a graph

AI-generated content may be incorrect.**

Learning rate seems to control how fast the agent can override old information. Because we lowered the learning rate, the model comes to convergence a little slower than control.

Gamma seems to control the value of future vs immediate rewards. The lowered gamma made the agent care more about finding immediate reward, and looking at graph with lower gamma I interpret that because it cares about immediate reward, it goes on a path that doesn’t really work at the end and so we have many more spikes going up on the graph.

Epsilon seems to make to converge down to a solution slower, but the peaks and lows in the chart seem to be larger. Epsilon gives the agent more flexibility to explore and take “risks”. Epsilon on this context determines if agent will pick random action to explore or just take best known path.

**POLICY ITERATION vs Q-LEARNING**

Q-learning: typically finds moves after training 12-15 steps to complete

Policy Iteration: consistently lands around 12 steps to completion

Policy iteration is super fast! With same map size of 7, policy iteration does it like instantly. Q-learning, you still have to wait a bit for it to finish. Q-Learning does have quite more statistics you can track. But when ramping up size for policy iteration, it takes forever and doesn’t seem to finish in same timescale as q-learning.

SKII with Q DEEP LEARNING

A graph with orange and blue lines

AI-generated content may be incorrect.

Meh, maybe if we trained for a lot longer results would show