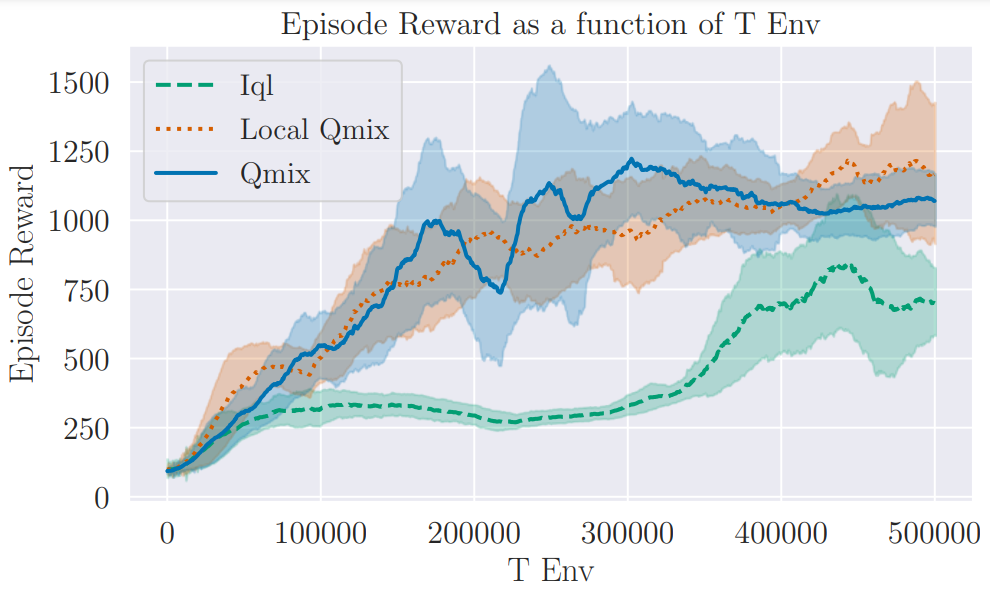
**Testing Documentation**

I’ve started to run quite a lot of tests and I felt the need to start documenting everything in a more organized manner.

**First Working Test:**

I can finally conclude that my code is up and running after checking and debugging quite a few things. This is the first proof that it is really working:



This was tun on a test with 4 cartpoles, and as expected for this scenario the architecture is quite similar to Qmix. This is probably why the results are so similar. IQL under performs both of these.

**Unanswered Questions and Objectives:**

* **Scaling up**
  + How does the system react to 10 cartpoles? 20?
  + How should we adjust the parameters as we grow? The batch size?
  + Add Access point example for easier scale up
* **Parameter Sharing**
  + Can the submixers share parameters if the inputs are symmetric and homogenous?
  + Should I implement QMIX without parameter sharing for agents?
* **Cooperation**
  + How does the spring and initial distances affect the results of these algorithms? Do larger initial distances require for higher coordination?
  + How does the neighbourhood size affect this?
  + What is the affect of the observation radius?
* **Other**
  + Should I check more up to date algorithms?

**Parameter Sharing Pitfalls**

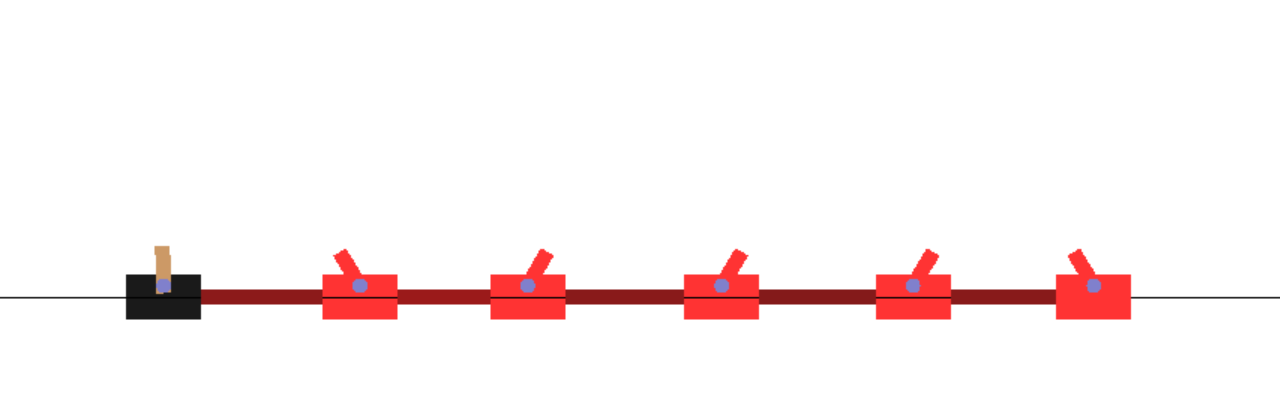
As I know, the code for the regular QMIX assumes homogeneous agents, and therefore incorporates parameter sharing for the agent modules. This means that the same network is used for every cartpole agent.

This makes learning harder, as the learning process might not converge properly! For example in the following test, the agent found an optimal strategy for the leftmost cartpole, which almost always reached the optimal solution, while more or less giving up on the rest. This is why the total reward was more or less equal to the max steps per episode limit. I think it somewhat understood that making the network invariant to the original starting position was too great a jump (potential mountation) and therefore it stuck there for a while. I think that action selection here needs to be chosen carefully, as one that still allows for exploitation after a certain amount of time.

I also think that it should be able to overcome this, even with parameter sharing. We as humans don’t get blindsided by seeing the position of a cartpole. Maybe we can even remove x from the observation to help training.

Conclusions: I am modifying the get state to only receive the relative position to the starting point. Observations will get the relative position as well.

Maybe I should implement QMIX without parameter sharing? I’ll add this to the major questions.



**More problems in parameter sharing:**

Found another big problem: The observation changes between agents, especially those at the edges. If the obs radius is larger than 0, we have a problem because padding with 0 might cause weird results. Also I didn’t really consider the padding direction, and edges looked like

* Left edge: [this cartpole – second cartpole - padding]
* Right edge: [second to last cartpole – this cartpole - padding]

So it’s really tough to learn anything in this scenario as well, since the cartpole isn’t even the middle coordinate.

Conclusions: be very cautious with radius > 0, and now I fixed the issue with the padding so that this cartpole is always in the center.