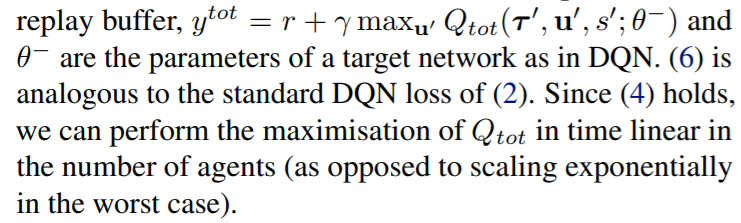
**Status Update – Week 6**

**Research Question**: Under what assumptions can we use a combination of global and local rewards in order to speed up training whilst ensuring cooperation? Specifically, can this be done under locality assumptions?

For example, in the multi cart pole scenario, it is obvious that the n’th cartpole has no effect on the first cartpole, and therefore it should not learn from it. This is clear from observation of the local rewards.

**Individual Maximization**

I have almost finished coding, but suddenly came across something that seemed (at first) a bit problematic: as we have stated before, as part of each individual loss we wish to compute . In the general case of QMIX:



This fact about exponential scaling is very interesting and can be applied to our algorithm as well. Notice that each is also monotonic:

Where the last transition is due to the monotonicity, we are enforcing on each of the submixers. So we can conclude that the maximization of the sum is just a linear maximization on each utility function, just like in the regular QMIX.

**Consequences of Linear Maximization**

This is weird, since I think that it has some implications regarding the entire algorithm. Last time we talked about:

But now we realize that for both of these, the maximization of the utility functions will give us the desired output. Let’s quickly prove this:

Define the following formalities:

Where are our submixers (from the locality paper), and are the individual utility functions. As we stated before, both and are increasingly monotonic functions thanks to the non-negative weights introduced in QMIX. with respect to their utility functions , It is enough to maximize each individually and then feed them forward. Let’s mark the maximizing individual function for as , and the joint action composed of them as

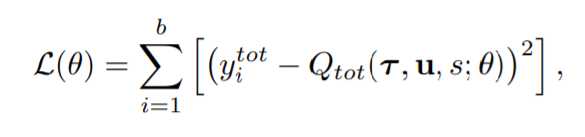
Then we obviously get:

This means that under **the QMIX assumption of monotonicity** the here is in fact 0! This is great since it means that our original algorithm was correct in terms of loss. The effect of l from the previous status only has **positive speeding up consequences, no correctness problems**.

This also means that if we can observe individual rewards and can make the QMIX assumption, we should definitely choose l=1 for the most direct credit assignement.

**Order of Squaring**

Small issue of squaring the loss. In the QMIX article, the loss is squared in a central manner:



We can implement this easily, however we have a choice where we wish to square:

Option 1: Squaring as usual -

Option 2 – Squaring each

The problem with option 1 is that it is exactly the same like QMIX. When the loss is written this way, it can be rearranged to create the original loss.

So what did we actually show before? We showed that the original loss is rearrangable so that we get the new loss (disregarding squares). This isn’t good, because it means that the loss we designed has no improvements regarding the old loss. The local rewards aren’t used here properly. Option 2 is a bit more like it.