Deep reinforcement learning knowledge nuggets

Marko Guberina 19970821-1256

March 17, 2022

1 How to get to a good method

Start by trying to solve the full problem. Write out the most complete and most accurate solution. Then find some number of approximations of that method. Looking at these approximations and thinking about what you really want out of this, try to answer the question of "what is the most minimal, yet clear signal I want to extract". Answer that question by constructing the simplest, most computationally efficient thing you can think of that could work. If it works, that's your method. **Always leverage getting the neural network to generalize and do that in the most computationally efficient manner possible, so that you can then leverage infinite compute (in RL you also get data through compute) to get it to work.** Example: exploration bonuses. Ideally you want a Bayesian approach, but that's of course intractable. You construct some counting method which approximates the Bayesian approach. You realize you don't even care about conting, just somehow differentiating novel states. You use some loss you already have lying around and say that high loss indicates novel territory. You compute even more than you did before and you get a better result on benchmarks.

2 Testing and debugging

Always start with the simplest possible setting, i.e. the simplest possible environment. Test your thing on Carpole, Pong etc. Compare how your method fared against other methods — if it stinks, something is broken.

3 Testing non-RL elements of an idea

If you want to see wether something neural network, supervised learning or whatever else related works, first test in on super simple standard test datasets or cases. For example, if you want to know how an autoencoder would deal with a certain scenario, create a mock of that scenario on MNIST.