Oliver Krengel
10-703: Deep Reinforcement Learning
Prepared for Ed and Henry

Steps to generate methods section and learn the codebase

Set up:

- Download the source code from Github: – midway_checkpoint branch
- Navigate to risk folder
- Source tensorflow environment
- Open attack_train_test.py

Take a look at lines 42 to 99
Training and testing are set up by tuning these parameters

In particular: look at 53-60 and 64-70:

- model_instance determines which file to load
- ‘0’ starts a new instance
- When training is complete, the console will print which model instance it has been saved to
- Put in that model instance to start from most recently saved model
- Leave checkpoint_index at -1 to load most recent model
- Don’t touch perform_update as this determines whether training or testing

Look at lines 78 through 83:

- These determine which strategies will be played against each other
- All the strategies I’ve written are here, and in the q_funcs/attack folder if you want to look at the source code
- Opponent cannot be LinearAttackNet, but otherwise all strategies are good for agent and opponent

Start testing:

- Set the agent and the opponent to random, now run: # You sourced tf, right?
  - python3 ./attack_train_test.py –train 0
- This will run 1000 games of the two players against each other
- Results should be 500/500 +/- 50
- Now set the opponent to MaxSuccess
- Run it again
- Results should be far worse for random

For the methods section:

- Make a table of the game results for every matchup between MaxSuccess, ArmyDifference, and Random
- Since random is the same as a randomly initialized network this is going to be helpful

Start Training:

- Set the agent to LinearAttackNet and the opponent to ArmyDifference
- Perform a test with a newly initialized network i.e. model_instance=’0’
- Record the results and set the model_instance to whatever is printed
- Take a look at the hyperparameters:
    - Text me after a while if you’re having a hard time with these, but I don’t want to tell you exactly what to use
    - I’ve gotten decent results but I’m sure they could be better
    - Some configurations make learning impossible
    - Note: 1000 iterations on my computer takes roughly 30 seconds
- Run:
    - python3 ./attack_train_test.py –train 1
- Record the results
- Run
    - python3 ./attack_train_test.py –train 0
- Record the results
- Now switch opponent to random
- Run the test again
    - How’s your win status looking?

For the methods section:

- Train 1 network against each of ArmyDifference, MaxSuccess, and Random
- Record the results of all trained networks against the 3 possible opponents

Some more stuff, more important for moving forward:

- Take a look at lines 93 and 94:
    - I have the starting territory fixed, toggle these lines to make it random
    - Do the same on lines 376/377 as well
    - We will need to be able to train the network to deal with actions for all states, so try random state switching and see how it trains in comparison
- If you want to make a new, deeper network
    - Navigate to the attack folder
    - Copy linear_attack_net
    - In the file:
        - Change the class name to indicate the new network
        - You can add layers right around line 92
    - Make a new directory in the attack file (NAME MUST BE EXACT)
        - mkdir “<module_name>_logs”
        - create a new file called 0.instance and make its contents 0
            - vim 0.instance
            - insert(i) 0
            - :wq
    - These steps will allow you to use the .instance branching I have implemented to save every network with a unique ID
- Try writing the code to randomly choose one of the three opponents every loop
- I wouldn’t try to go through the main loop in the test, it could easily be pared down and will be