Lab 5 Notes

MV4025

August 23, 2019

C. Darken

Objectives

- Analyze and try to improve the starting Lab 5 codebase
- Train multiple systems that learn to attack the enemy
- Analyze the degree to which they learn (or fail to learn) to concentrate force before attacking
- If the force concentration is lacking, formulate a hypothesis as to the source of the problem, and also as to what should be done to fix it
- Time permitting, try to fix it!

Codebase

- The starting codebase is a good approximation of the state of the art for a running research project
  - More complicated, chaotic, and possibly buggy than the codes you have encountered in this class so far
- The goal of the project is to train an opposing force (red) to attack a human player (blue)
- The hope is eventually to provide
  - A sparring partner for the human player for CoA (Course of Action) analysis
  - Human behavior models for use in a range of DoD simulations

Outline

- Combat model
- Lab 5 scenario
- AI
    - Algorithm
    - Input (“state”)
    - Output (actions)
    - Reward functions
- How to run
    - Interactive training and testing
    - Server mode
    - Interactive testing only
- Experimental factors (parameters)
- Important source files

Combat Model

- Movement: constant speed towards specified point. Movement through other units is not permitted (they will block each other)
- Terrain: none (the terrain you see is ignored), i.e. we are fighting on a featureless plain
- Shooting:
  - When an entity first finds itself in range of one or more enemies, it takes the closest as its target
  - A shot is scheduled (uniform distribution on [0.2, 0.3] seconds)
  - Next AI update for the target, it can schedule return fire
  - Each shot has a 10% chance to kill
  - Units respawn (return to life at a random location) after one second

Lab 5 Scenario: Concentration of Force

- Two red forces, one blue
- Goal is to maximize blue kills minus a penalty for each red loss
- Everything below this line my analysis. The objective of this lab is predicated on it. Do you agree?
- Optimal behavior clearly depends on the size of the penalty
    - Small penalty: red should directly charge blue under all circumstances (“small” is believed to be less than one)
    - Very large penalty: red should avoid contact with blue (“large” is believed to be in excess of 1.6 or so)
    - Intermediate penalty: red should attack blue, but in a coordinated manner, i.e. both should start shooting at the same time (“intermediate” is believed to be the interval strictly greater than 1 up to 1.6 or so)

AI: Algorithm

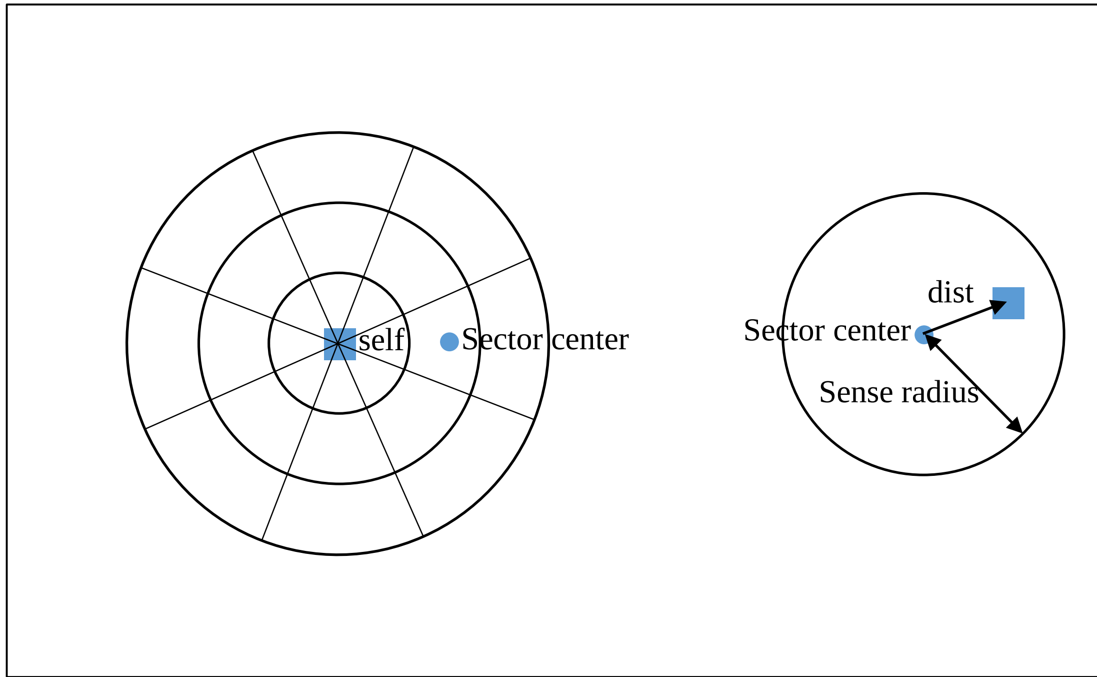
- The algorithm used to control the red forces is Q learning where the Q function is represented by a neural network, often called “neural Q learning” or “deep Q learning” (although our network will not have many layers, i.e. will not be very deep)
- *Q(a,s)* gives an estimate of the long term future discounted reward following taking action *a* in state *s*
- For this lab, *Q* is represented as a function that takes state *s* as an input and produces a vector of values, one per action

AI: Algorithm Details

- AI takes an action and possibly does one learning cycle every 0.1 sec
- Learning is based on experiences, which are stored in an experience buffer with random replacement
- A single experience consists of an initial state, an action that was taken in the initial state, the reward that was obtained (could be zero), and a resulting state
- Note that the code for pairing initial and final states is not completely straightforward. Is it correct?
- Learning is done by mini-batch gradient descent, where the batch is randomly selected from the experience buffer

AI: Input (State)

- The default input to the AI is a set of 48 numbers representing how much friendly resp. enemy force is present in each of 24 (see Mervyn's diagram below)
- For each force and for each sector center, the distance to the center is calculated (D), and then max(0,1-D/10)
- Exception: if the above is zero for all sectors, one is added to the sector corresponding to the outermost range ring sector closest to the entity



AI: Output (Actions)

- The nine possible actions are
  - To move in one of the cardinal or diagonal directions (eight directions)
  - To do nothing

AI: Reward

- Rewards are given to individual units
- +1 is added every time they kill a blue unit
- loss_factor is subtracted every time they die
- I recently observed that the death penalty was not getting incorporated into AI experiences. I believe the problem is fixed, but have not carefully verified it yet.

How to Run: Interactive Training and Testing

- Set parameters in ExperimentControl.cs (or create a todo.txt as for batch runs)
- Set the time scale to 100 (or to 1 if you want to observe the training). Time scale is set in File->Build Settings->Player Settings->Time->Time Scale
- Run until training is complete and system pauses
- Set time scale to 1 if you want to observe the testing
- Unpause and allow testing to complete

How to Run: Server Mode

- If running Windows, you will need a Unix-style shell. I use “Git Bash” which is part of the Git for Windows installation
- Do a server build for your target platform (File->Build Settings)
- Copy files from Tools into your build directory
- Edit Tools/replication-generator.py to specify experiment design
- Run it and store results in a file called todo.txt
- Edit run.sh to use the proper executable name for your build, e.g. “CoA\ AI.exe” (with the space, without the quotes) on Windows
- Edit and execute run_all.sh to start whatever number of simultaneous processes your machine can handle (might be just one)
- Numeric performance results appear in result.txt. For a run with seed xyz, the trained neural network is stored as brain-xyz.bin

How to Run: Interactive Testing Only

- This procedure is useful if you want to judge the performance of a particular neural net by eye and possibly capture to a video
- In ExperimentControl.cs, change the curriculum to CurricName.Lab5Test.
- Move the brain-xyz.bin file containing the neural net you want to observe to your top level (CoA_AI) directory and rename it brain.bin
- Start the application in the editor

Deliverables

- Written report submitted via Sakai. Should be more detailed enough to convey and support your main points.
- Debrief your work in our last lab session of Friday, Sep 13. If we run out of time, I will ask you to present on Tuesday, Sep 17 instead.

Experimental Parameters

- discount_factor: positive number less than one
- type_hidden_units: relu, tanh, or sigmoid
- test_duration: in seconds
- learning_rate: small positive number
- reward_timeout: scenario resets if this many seconds elapses without a kill
- mobility_model : leave as NoGradePenalty for Lab 5
- seed: initializes random number generator
- load_brain: start by loading neural net from file, as opposed to random weights
- num_hidden_units: if zero, then there is no hidden layer
- train_duration: in seconds
- respawnWidth: size of box in which units are placed after death
- loss_factor: amount of penalty per friendly loss, zero or greater
- ranged_state: form of the input state to the neural net, leave as True for Lab 5

Important Source Files

- ExperimentControl.cs: sets up the experimental parameters and curriculum
- Curriculum.cs: controls which scenes will be used for training/testing
- Entity.cs: the combat model, state creation, reward generation
- Brain.cs: experiences, the experience buffer, neural net training

Suggestions

- You could consider…
  - Using a larger discount factor (I often use 0.99) and loss factor (I use values greater than 1 and less than 1.6) when you are trying to see force concentration
  - Deeper neural networks by altering the network creation code and allowing the user to specify the number of units in each layer
  - Creating a visualization experiences (original state, action, reward, resulting state tuples) inside of the codebase or by exporting the data to an external tool. Do they make sense? Are they sufficient to learn force concentration?
  - Capturing video of particularly good or bad runs for the debrief session
  - Making your own scenarios that might provide clearer results

Major Bugs

- Please report any bugs that clearly make the codebase unfit for the stated purpose of the lab immediately!