CS4341

Assignment #3: Machine learning

Due date: February 21, 2022 @ 1:59 pm

This assignment will familiarize you with applying machine learning to a task, namely the navigation world from assignment #1. Your job is to learn a heuristic function to guide your A* search and to evaluate how well the learned function performs. There isn't much programming for this assignment. You will mostly be adding some record keeping to already-existing code, doing offline analyses, and then tweaking your A* to use a new heuristic function.

The task

You will use the same exact puzzle from assignment #1: the environment, actions, and movement costs are all identical to that assignment. You will apply A* search to allow your robot to get to the goal. You may reuse your code from assignment #1. If no one in your group had a satisfactory solution, you may (with permission) use a classmate's code from assignment #1 **as long as you attribute it.**

The key difference for assignment #3 is you will *learn* a heuristic function rather than make one up. You are responsible for constructing features, training a model, and then testing that model with your robot.

Your learning agent has access to:

- The value of the squares immediately adjacent to the robot (8 squares)
- The value of the squares around the goal (8 squares)
- The robot's location (x,y) and facing direction
- The goal location (x,y)

You can use any of the above as features for your learner, and are free to compute features from the above. For example, you may want to compute the horizontal and vertical distance to the goal. You will use your features to train a model to guide your A* search. The dependent (y-value) you should predict is the future path cost to the goal from this location (see next section for advice on how to do this).

Once you have trained your model, you will integrate it into your code and use it as a heuristic to guide your A* search. Your learned heuristic need not be admissible (and probably won't be).

Suggested approach to train a model

You are free to use any learning infrastructure you want. The following are only suggestions.

Here is how I would approach this assignment:

1. Start with a working solution for assignment #1 and use Heuristic #5 for the following steps (do not use heuristic #6).
2. Whenever A* search terminates, for each square on the path you should record:
    a. The **actual** path cost to the goal *from this node.* For the start state, this value will be the total path cost; the value will decrease for each move closer to the goal the robot gets until the last action should have a path cost to the goal of 1 (since the goal always has a move cost of 1).
    b. The values of your features
    c. Write the feature values and path cost to a .CSV file
    d. See https://docs.google.com/spreadsheets/d/161Bynt6pEilFWOlfDC924vTf3J3wCk6eO8MjsJzPSD8/edit?usp=sharing for an example of this process
3. Find a map size that your solver takes about 10 seconds to solve, and have it solve several hundred boards to collect data. Two hours should be long enough to get sufficient data. At the end of this process you should have a large collection of data: for each of the several hundred maps solved, you should have one line in the file for each action taken by the agent. Each line will have your feature values, and the value returned by A*.
4. Read the .CSV file into Weka and use linear regression. Linear regression will be easy to reintegrate into your code later.
5. Train a model using Weka and take note of the model fit statistics.
6. Use the coefficients in the learned model to code a new heuristic for your A* learner.

The above uses things you already know. You are free to take other approaches, such as using different software, training a neural network, or using online learning to update the parameter values.

Running your code

Your code should execute in the same way as for assignment #1: it should take two command line arguments containing the file name and which heuristic to use. Your machine-learned heuristic should be run when your program is given a “7”.

Analysis and writeup

First, explain your learned model. Explain your features (an example may help to illustrate them) and model type (e.g., linear regression, neural network). What software did you use to

train it? What are the model's coefficients? What is its degree of model fit (if you're using Weka you can use correlation; if you're using software that doesn't report that feel free to use RMSE)?

Now collect some experimental data. You will compare your learned heuristic (Heuristic #7) to Heuristic #5 and Heuristic #6 from assignment #1. You should use those heuristics' performance on the code base you used to train your learner. Do not use teammate A's codebase and teammate B's analysis from assignment #1: the results may not make sense. **Generate new data for this assignment**.

Use the same puzzle size you used to train the data[1] (about 10 seconds per board). Run 10 boards in total. Run each board with all 3 heuristics and record the number of nodes expanded and the solution cost. Compare your 3 heuristics in terms of branching factor, and mean solution cost. How does your learned heuristic compare to H5 and H6?

[1] Serious methodologists will raise a concern that we're testing our learned heuristic under the same exact conditions in which we trained it. We really should show that what we learned *generalizes* to new situations (e.g., larger or smaller grid worlds, gridworlds with a different range of values, …). That said, this is intro to AI, so we'll let it pass. Of course, methodological purists already shed a tear when we violated the IID assumption by writing multiple lines to the .CSV file for each trial. Do whatever statistical penance you feel is necessary.

IF I RUN THIS AGAIN SHOULD ONLY HAVE STUDENTS SAVE THE INITIAL COST TO GOAL FOR EACH TRIAL. HAVING FEWER DATA IS BETTER THAN VIOLATING IID ASSUMPTION, AS VASTLY INFLATES R2 VALUES