Project 3:

Reinforcement Learning for Robot Obstacle Avoidance and Wall Following

---

Assigned: Tuesday, Oct. 26
Reminder: Mini-Assignment (already assigned) Due: Monday, Nov. 1, 22:00:00 (for everyone)

Undergrads: Generally complete, compilable code: due Mon., Nov. 15, 22:00:00
Completed software and paper: due Friday, Nov. 19, 22:00:00

Grads: Completed software: due Mon., Nov. 15, 22:00:00
Completed paper: due Friday, Nov. 19, 22:00:00

Overview

In this project, you will use the Sarsa(λ) online reinforcement learning algorithm to teach a robot to follow a wall and avoid running into obstacles. You will be using the Player/Stage simulator to simulate your robot, using an environment map that is provided to you. You'll be using a laser range scanner and odometry on the robot to perform the learning, where the robot moves using steering and velocity commands. You'll turn in your software that performs the learning, instructions for running your software, and a paper (for graduate students, 3-6 pages; for undergraduate students, sufficient documentation to illustrate your results) that describes your project and results.

A note on collaboration: For this homework, I highly encourage discussions amongst yourselves for the purposes of helping each other learn about Player/Stage, how to control a robot in Player/Stage, how to read sensor values, etc. This collaboration is solely for the purposes of helping each other get the simulation up and running. You may also discuss high-level concepts in helping each other understand the Sarsa(λ) algorithm. However, each person must individually make their own decisions specific to the learning aspects of this project. In particular, this means that individuals must make their own decisions on the representations of states, actions, and rewards to be used in this project, and must implement their own learning algorithm and prepare all the written materials to be turned in on their own.

Formulating the problem

In this project, we make the following informal definitions:

- *Following a wall*: means that at time *t*, the moving robot is at a desired distance d_w from the wall, and that this distance is maintained at time t+1. Note that a critical part of this informal definition is that the robot is moving. A robot that maintains a distance d_w from the wall at time t and t+1, but is not moving, does not fit the definition of following a wall. For the purposes of this project, we will consider a robot that covers longer distances *while following a wall* to be better than a robot than only follows a wall very slooooowly. (Note that we didn't just say we prefer a robot that covers longer distances. The robot should only be rewarded if it covers longer distances while also following the wall.) Note also that the robot can follow a wall from either side (right or left), except that once it is following a wall on one side (say, the right), it should continue to follow the wall on that side until the wall is lost. If you prefer, you may restrict the definition to be only right-side wall following or only left-side wall following (meaning that the

robot only follows walls on one side, and never on another), if this makes your robot learn faster or better. All these issues may affect your reward function.

  - For this project, we will set d_w to a value of 0.75m.

- *Avoiding obstacles*: means that the robot never gets closer than distance d_o to any object in the environment.
  - For this project, we will set d_o to a value of 0.25m.

As part of the project, you are responsible for formulating the remaining details of the learning problem. For example, you'll need to determine how you will represent the input state and the motor actions, and the best level of discretization of these values. You'll need to decide the reward function you'll use; presumably you'll have a positive reward for following a wall and a negative reward for running into an obstacle. As mentioned already, you may also have a component of the reward function that rewards moving longer distances along a wall, rather than have the robot creep along the wall. You can decide yourself where the starting position of the robot will be for each learning trial. You'll need to determine how many episodes are needed for your program to learn.

Some caveats:

- You are not allowed to use a grid-world representation of states. States should be a function of sensor values (and other measurements, as appropriate).
- You may define your reward function any way you like, including the use of scaled/graduated rewards as a robot moves toward or away from a wall or obstacle, if you find such rewards helpful.
- Your robot may have issues when going through a doorway, since it might not be possible to both keep a distance of d_o from an obstacle while also keeping a distance d_w from the wall. Ideally, the robot should still be able to enter rooms and follow the walls around the rooms, as well as the hallways. You should explore this issue to determine what your robot is able to learn. In this exploration, you may also study the effect of changing the values of d_w and d_o.

As always, there is not just one single way to achieve this robot learning. Explore various design possibilities and find the one that you believe works best.

Player/Stage Simulator

You have already been given a tutorial on Player/Stage. A Player/Stage Getting Started Guide is available on the course website, here:

http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall10/Handouts/PlayerStageGettingStarted.html.

Full documentation on Player/Stage is available online, at http://playerstage.sourceforge.net/doc/doc.html.

A shared version of Player/Stage is available for you on our UTK CS Linux machines (i.e., cetus and hydra machines), so you should not try to install Player/Stage yourself (takes up too much disk space, and labstaff won't be happy!).

By following the Player/Stage Getting Started Guide, you'll find example code for getting started. The laserobstacleavoid.cc code shows you the basic way to interface to your robot, read sensor data, and make the robot move.

Any questions you have on Player/Stage should be directed to the TA (Richard Edwards, redwar15@eecs.utk.edu).

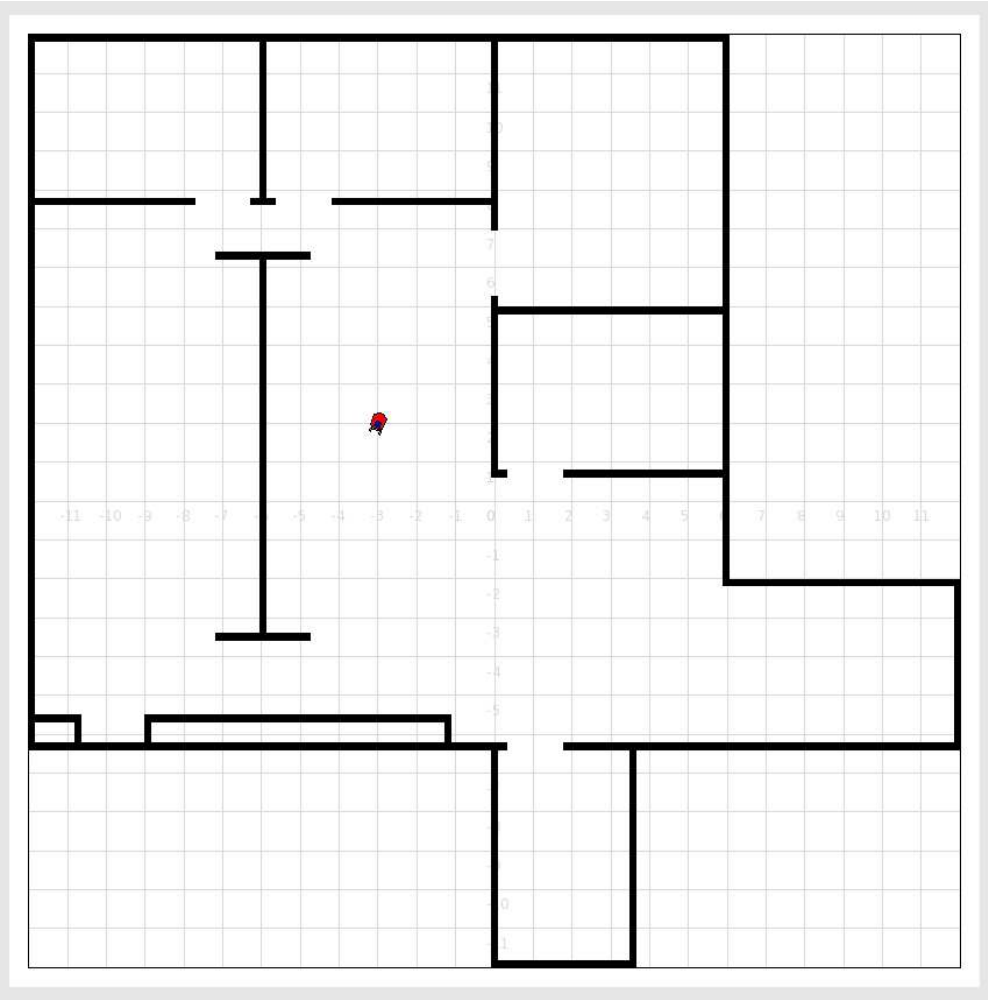
Configuration and World files for Project #3

I have prepared the configuration and world files you should use for this project. They are available here:

Project-3.cfg: http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall10/Projects/Project-3.cfg

Project-3.world: http://web.eecs.utk.edu/~parker/Courses/CS425-528-fall10/Projects/Project-3.world.

Below is what your simulation environment would look like if you were to start up player using Project-3.cfg (which implicitly uses Project-3.world):



What your submitted code should do

The final code that you submit should have 2 options – one option is the reinforcement learning algorithm in learning mode, and the second option is to test the best learned policy for an arbitrary robot starting position. Your code should allow the user to specify whether the code should be run in the training mode or the testing mode. This means that your project submission should include the best policy you found during your own training, so that you can use those during testing mode. It is up to you to design the user interface to allow the user to specify which mode the code should be in. Be sure to provide documentation in your "README" file on how the user should specify the inputs. Or, make your user interface so easy to use that the user just has to follow your program's online instructions.

CS425 – Undergraduate Instructions

Undergraduate Student Paper Guidelines

As part of your project, you must prepare a single 1-3 page document (in pdf format, using Latex and 2-column IEEE style) that documents the reinforcement learning approach you used (including your definitions of states, actions, and rewards) and the values of any parameters (such as the discount rate) that you used in your system. This document must also include results of your reinforcement learning solution, including snapshots of your robot during learning, and subsequent to learning. (Be sure to turn on "robot trace" for these snapshots). To the extent you desire, you are welcome to add additional descriptions along the lines of the graduate student paper. However, this is not required.

The documentation you turn in must be in pdf format in a single file. Name your paper *yourlastname*-Project3-Paper.pdf.

Undergraduate Grading

Your grade will be based primarily on the quality of your project implementation and the documentation of your findings in the writeup. You should have a "working" software implementation, meaning that the learning algorithm is implemented in software, it runs without crashing, and performs the learning task. Your results description should be sufficient for the reader to tell that the robot has in fact learned.

Note that, as with the previous projects, you first must turn in a "generally complete" version of your software, prior to the final version of your software. "Generally complete" means that your code:

- Compiles
- Implements the Sarsa(λ) reinforcement learning approach, including the robot interface to control the robot's motion and sense the robot state
- Runs without crashing

This code (for the first deadline) does not have to demonstrate good learning performance, however. It just needs to be at the stage that you can tweak the design of the RL algorithm and the parameters of the system to get it to learn properly. However, it does need to have all the implementation completed for the main learning procedures.

For the final submission, you will turn in working software and the paper, as described earlier.

Your grade will be based on:

- 15%: "Generally complete" software (i.e., software with the above characteristics). This software does not have to learn well. But, it has to have all the pieces in place for you to begin tweaking the parameters to achieve good learning performance.
- 65%: The quality of your final working code implementation, and software documentation. You should have a "working" software implementation, meaning that the learning algorithm is implemented in software, it runs without crashing, performs learning as appropriate for this project, and learns reasonably well. Your final code should have the training mode and testing mode, as outlined earlier in this document.
- 20%: Your paper, generated in Latex using IEEE styling, and submitted in pdf format – Figures and graphs should be clear and readable, with axes labeled and captions that describe what each figure/graph illustrates. The content should include all the results and discussion mentioned previously in this document.

Undergraduates: Turning in your project

You should submit your project using the class submit script, **425_submit,** by the deadlines. ***Note that you will use this submit script twice, since you have two (2) deadlines for this project.***

Submission #1:

1. This submission is of the "generally complete code". It should include all the programs, include files, makefiles etc., needed to compile and run your project on the CS linux machines (cetus or hydra), including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project2-prelim.cc (or whatever extension is appropriate for the programming language you're using).

Submission #2:

This submission should be in 2 parts:

1. Paper (in pdf, 2-column IEEE format, generated using Latex, named *yourlastname*-Project2-Paper.pdf)

2. All the programs, data files, include files, makefiles etc., needed to compile and run your project on the CS linux machines, including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project2.cc (or whatever extension is appropriate for the programming language you're using). To use the submit script, place all your files in a single directory. Remove any files that are not necessary for your project. Then enter the command line "425_submit", and answer the simple questions you're asked by the script. (This script only works on cetus and hydra.) This script will make a tarball of your files and email it to the instructor, Lynne Parker, and the TA, Richard Edwards. When we unpack your files, we should be able to read your README file and follow the instructions to compile and run your code. We'll be in a bad mood when grading your work if the instructions you provide do not work. So please test your instructions before you send your files to us.

CS528 – Graduate Instructions

Graduate Student Paper Guidelines

As part of your project, you must prepare a paper (3-6 pages) describing your project. Your paper must be formatted using Latex and the standard 2-column IEEE conference paper format, the same as in previous projects.

The paper you turn in must be in pdf format. Name your paper *yourlastname*-Project3-Paper.pdf. Your paper must include the following:

- An abstract of 200 to 300 words summarizing your findings.
- An introduction describing the robot learning task and your formulation of the problem, including the definition of the states, motor actions, and rewards.
- A detailed description of your experiments, with enough information that would enable someone to recreate your experiments.
- An explanation of the results. Use figures, graphs, and tables where appropriate. Your results should make it clear that the robot has in fact learned. This means you'll need to give some pre-learning results with which to compare your learning approach. Ideally, results are quantitative, not qualitative. This means that you have some concrete measure by which to evaluate your results. These results must include at least 2 snapshots of your robot performing the task – one during the learning, and one subsequent to learning (i.e., with the robot executing the learned task).
- A discussion of the significance of the results.

The reader of your paper should be able to understand what you have done and what your software does without looking at the code itself.

Graduate Grading

Graduate students will be graded more strictly on the quality of the research and paper presentation. I expect a more thorough analysis of your results, and a working learning system. Your analysis should include a discussion (and perhaps results) on two or more of the following points (in addition to the points previously noted above):

- Rate of convergence of learning
- Effect of the layout of the learning environment (if you want to try the learning in different environments; several test environments are available to you as examples)
- Effect of different state and motor output representations (e.g., using more or less resolution) on the quality and speed of learning
- Effect of different reward assignments on the quality and speed of learning
- Future work that you believe would improve the learning
- Any other insightful observations or experiments you'd like to make

You should not address these points in a bullet-type fashion, but instead work the answers into your paper in a discussion-style format. The paper should have the "look and feel" of a technical conference paper, with logical flow, good grammar, sound arguments, illustrative figures, etc. As always, graduate students are expected to format their paper in standard IEEE conference format.

Please do not exceed 6 pages for your paper (and don't go smaller than 10 point font).

Turning in your project

You should submit your project using the class submit script, **528_submit,** by the deadlines. ***Note that you will use this submit script twice, since you have two (2) deadlines for this project.***

Submission #1:
This submission is of your final code, and should include all the programs, include files, makefiles etc., needed to compile and run your project on the CS linux machines (cetus or hydra), including a README file that gives instructions on how to compile and run your code. Your main project code file should be called *yourlastname*-Project2.cc (or whatever extension is appropriate for the programming language you're using).

Submission #2:
This submission is of your paper (in pdf format, named *yourlastname*-Project2-Paper.pdf). To use the submit script, place all your files in a single directory. Remove any files that are not necessary for your project. Then enter the command line "528_submit", and answer the simple questions you're asked by the script. (This script only works on cetus and hydra.) This script will make a tarball of your files and email it to the instructor, Lynne Parker, and to the TA, Richard Edwards. When we unpack your files, we should be able to read your README file and follow the instructions to compile and run your code. We'll be in a bad mood when grading your work if the instructions you provide do not work. So please test your instructions before you send your files to us.