CMPUT 607 W17: Applied Reinforcement Learning

Robot Module 3
"Horde and Pavlov"

Written aspects are to be handed in by email to pilarski@ualberta.ca by 11:59pm on the due date.

Learning Objectives

- You will learn how to deploy a large collection of general value functions (GVFs), i.e., *Horde*.
- You will learn how to measure the error of this collection while it is learning in real time.
- You will learn how to present learned predictions as inputs to simple control rules that change the behaviour of your robot in a pre-determined way (i.e., *Pavlovian control*).

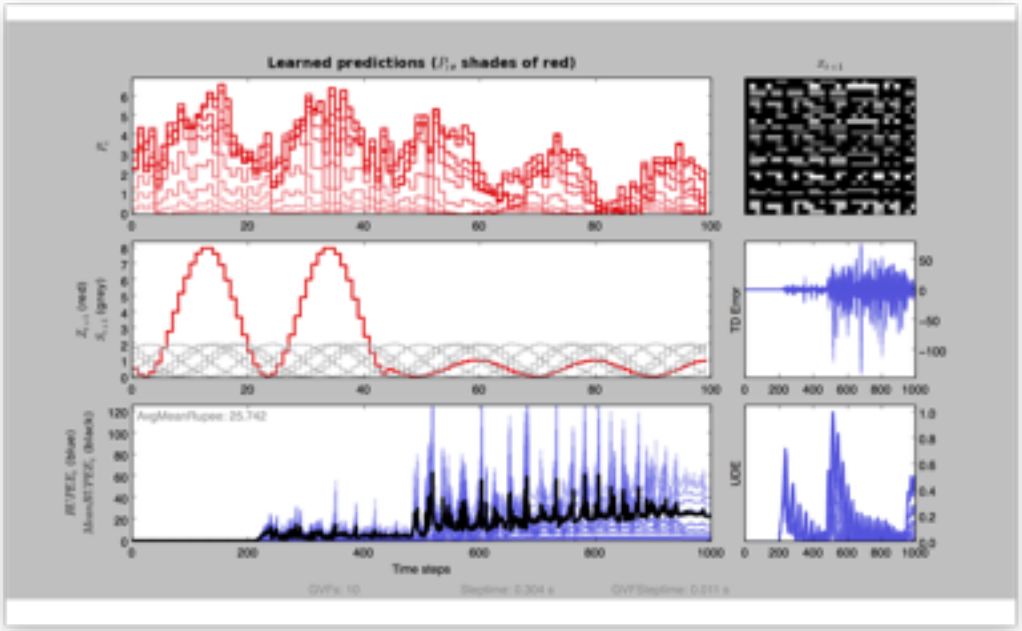
Performance Objectives

- You will have demonstrated in written and visual form that you are able to ***implement a collection of on-policy and off-policy GVF learners (Horde)*** to predict multiple aspects of your robot's data stream from a single behaviour policy.
- Your Horde must include ***at least 10 different GVFs***. Your Horde's question functions should also include at least three different policies (π1,π2,π3,…), and at least three different continuation functions (gamma or gamma_t).
- You will have demonstrated in written and visual form that you can ***compute and plot RUPEE and UDE measures*** for all GVFs in your collection; you will show that you can plot these error measures both in real time and offline.
- You will have demonstrated in written and visual form that you can ***implement Pavlovian control***; you will show this by demonstrating that your robot can respond in a fixed or pre-determined way to one or more of the learned predictions in your Horde.

What to Do

- This module consists of the following three main practical activities:

    - **Implement Horde**
        - *Extend your previous experiment to include ten or more GVFs*. If you like, you can keep the same behaviour policy, or change it and your robot as you see fit.
        - For this activity, you will likely wish to *create a new data structure* or set of data structures to contain your Horde of GVFs. This data structure should allow you to manage updating and reading from your GVFs. For example, you may want to create an object that stores a list of GVFs with their weights and their parameters (step sizes, question functions, etc.), such that you can quickly call an update method on each (or all) GVFs, and return an array with all their predictions, all their error measures, etc. For example, you might have methods like "pred_array = Horde.update(x_t, x_tp1, actions, z_array, gamma_array)" and "pred_array = Horde.get_predictions()".
        - *Extend your online and offline plotting approaches* such that you can plot all of these new predictions, the signal space of your learners, and the cumulants delivered to your learners. You may structure these plots in any way that you feel most clearly shows the predictions and their relation to the cumulants (e.g., one plot per GVF, or conversely, one plot for all the predictions and another for all the cumulants.) For easy debugging, and as shown below, you may also wish to plot the x_t values delivered to your GVFs. Some additional examples are provided in the dropbox, and can also be seen in the related readings.



- **Implement RUPEE and UDE within your learners**
  - *Implement the algorithms for RUPEE and UDE*. Look in White 2015 to determine the implementation details and recommended parameter values for these algorithms.
  - Verify that these algorithms are working correctly when you run your Horde. To do this, implement routines to plot both RUPEE and UDE values for all GVFs in your Horde, with *plotting functions that work both in real time and offline*.
  - *Surprise your robot*. Have it doing something easily learnable and then, after some time, change some aspect of its operation so that you expect large TD Errors for one or more of your GVFs. *Note what happens to RUPEE and UDE values when this event occurs*. A simple example is just doubling the signal used as the cumulant for some of your GVFs.

- **Implement Pavlovian control**
  - *Pick one (or more) of your predictions as the inputs to your Pavlovian control routine*. For quick success, these should be easily learnable and clear (not noisy) predictions.
  - *Decide on a hand-coded behaviour* for your robot to perform in response the the magnitude or change in magnitude of this prediction.
  - *Implement your control policy*. Think of it like "robot reflexes": in the simplest case, if your robot predicts something will happen, it has a hand-engineered policy that takes over some or all of its behaviour to perform an appropriate and timely response.

  - ***Examples****:*
    - A *set point*, as used below, refers to a specified value or range of values for one or more of your actuators (i.e., the bounds of a servo's range of motion might be called set points, where the first set point is servo.read_encoder() < 200 and second is servo.read_encoder() > 900.)
    - ***Example 1:*** A prediction of load or position exceeds a given threshold while your robot is moving; your robot switches direction.
    - The predicted time until reaching a positional set point is less than a fixed threshold; your robot twitches one of its joints to signal this event.
    - ***Example 2:*** The motion of one actuator is a function of learned predictions, e.g., it is set to the next predicted angular set point the other actuator will pass (via gamma_t), or the future position of the other actuator (via fixed gamma).
    - ***Example 3:*** Following a random policy, your robot's prediction of the number of steps to a angular set point is greater than a certain value if the robot were to choose to move directly to that set point (off-policy question); robot moves toward the set point until this is no longer true.

What to Submit

- *Three things* need to be submitted to meet the evaluation requirements of this module:

  - **A written report in PDF format.**
    - This report will demonstrate that you have completed all performance objectives. Please be sure to look over the evaluation criteria below to guide your write-up.
    - In terms of style and length, please consider this report to be equivalent to a short conference or workshop paper, or an industry white paper.
    - Start your report with a short abstract describing its contents.
    - Please place the focus on the performance objectives, e.g., by making one section for each objective. The document structure can be straightforward and matter of fact. You do NOT have to craft a glorious narrative.
    - You are NOT being marked on the beauty or polished nature of your figures. If you draw a clear, appropriate figure on a whiteboard and take a photo, that is preferable to spending far more time making a Adobe illustrator graphic that communicates the same thing. You can always polish your figures later for your final report, if needed.
    - Data plots should be well labeled and chosen so as to clearly communicate the relevant ideas as they relate to your performance objectives.
    - When listing equipment or experimental apparatus, where appropriate, be sure to include the supplier / manufacturer as per standard practice.
    - You may use any standard conference or journal paper template you feel is most appropriate to your field of work. Examples include ICML, AAMAS, NIPS, IEEE Conference Style, etc.; you may use Word, LaTeX, Pages, or any other template that fits with your own thesis working environment.
    - Please do not exceed 8 printed letter-size pages. There is no minimum length. As you will want to include a number of figures and images, you will find this naturally limits the amount of writing you need to do to a reasonable level.

  - **An archive of the code you have written and used to complete the module.**
    - Please send an archive (e.g., .zip, .tar.gz) or a Git repository link that includes all code you have used to complete this module.
    - Please include all code you have written yourself (your experimental wrappers, any control code, etc.) and any publicly available or shared modules you have downloaded to make it work—if these are very large in size, please just provide a text file with a list of modules used and the location of the website / source you used to download them.
    - Please explicitly label the code you have written yourself, and the code you have downloaded form other sources (e.g., via comments in the first few lines of the file and/or in a separate text file like "authorship.txt" within the archive).

  - **A short video of your robot moving alongside a real-time view into your Horde.**
    - Video can be in any standard format, e.g., .mp4, .mov; *sending a link is fine*.
    - Video does NOT need to be production quality. It can be as simple as a video taken with a smart-phone of your robot moving next to your computer monitor, with learned predictions scrolling on-screen in (near) real time as the robot moves, alongside your measures for RUPEE and UDE, during Pavlovian control.

Evaluation Criteria

- Robot modules make up 20% of your final course mark; each of the five modules is therefore worth 4% of your final mark. Marks between 0-4% for this module will be assigned as follows:

  - **All performance objectives present with a level clarity, detail, and rigour that would be considered acceptable for presentation to other academics or industry (4%)**

    - ***Clarity***: Could a skilled but non-expert (outside) reader follow what you have done and why? Is what you learned and what you found presented in an understandable way for an outside reader? Are all visual elements well labelled and readable?

    - ***Detail***: Could an outside reader repeat what you have done and the way that you did it by reading your text? Does an outside reader have enough information about the equipment, sources, and code you used to replicate your work, should they have the required resources?

    - ***Rigour***: Does an outside reader have enough evidence to believe that what you write and claim is correct? Is your description supported by convincing data?

  - **All performance objectives present but one or more performance objectives lacking in detail, clarity, and/or rigour (3.5%)**

    - *At least one* performance objective with acceptable clarity, detail, and rigour.

  - **Missing one performance objective** [or] ***all* performance objectives present but *all* are lacking clarity, detail, and/or rigour. (3%)**

  - **Missing more than one performance objective (2%)**

  - **Most or all performance objectives missing / largely incomplete (1%)**

    - Evidence that an attempt was made to complete the module (e.g., new code to address the module objectives, video of robot, summary of results.)

  - **Did not complete and submit the assignment (0%)**

Collaboration Guidelines

Every student is expected to submit an individual report and conduct their own robotics and RL experimentation. **This module is not a group project.** This module is not about testing your ability to write code, or work on your robot for robotics sake alone. I am not evaluating you on those aspects of your module. With this in mind:

- ***It is okay to work together on robot infrastructure***, scripts, routines that effect the non-RL parts of the robotics. Sharing a routine to send actions to an actuator, reprogram its fields, communicate via ROS, ROS infrastructure, and other things of a similar nature are totally fine and encouraged to make the course really about how reinforcement learning interacts with the robot and not just about fighting with the robot to get it to work.

- ***I expect you to write all your own RL code and the code that runs your experiments.*** While you can and should share approaches with each other, I will be looking specifically at how you each conduct your empirical RL research. This is not just so I can see your understanding, but also so that I can give personalized feedback to help you grow as an applied RL researcher.

- Writing machine learning code is in fact one of the most important things for understanding how principle meets practice, and truly getting to know your algorithms. ***Cut and pasting another student's learning code and the experimental wrappers to run it is not oka***y, defeats the purpose of the course, and will be considered academic misconduct. (As a reminder, and not surprisingly, copying other things like data, written assignments, etc. is also not okay.)