Practical: Evaluating Evaluative Feedback and Describing Markov Decision Processes

MSc. Interactive Entertainment Technology
CS7032: AI & Agents
Liliana Mamani Sanchez (mamanisl@cs.tcd.ie)

November 24, 2015

1 Goals

To replicate the experiments with the 10-armed testbed [SB98] presented in class. To investigate different settings of the exploration vs exploitation dilemma. And to model a simple game as a Markov Decision Process (MDP). The MDP modelling part is unrelated to the R code given.

2 Some code

In the course web site you will find some code for this practical\(^1\). This file contains functions that create an n-armed bandit task (makeNArmedBanditTask) and run the $\epsilon$-greedy experiment described in the notes (runExperiment1).

The code is written in R, a programming environment for data analysis and graphics which is freely available from the R project website [VS04]. R has an Algol-like syntax which should make it relatively easy to learn the basics. The code provided does not use any of the advanced features of the language, so it should be straightforward to extend and modify it. A short introduction to R is available through the course website. Only question 2 requires some coding. However, you may use a programming language of your choice, or simply answer question 2 using pseudo-code.

The R code provided implements the basics of evaluative feedback as described in our lecture notes:

- function `makeNArmedBanditTask(n,s)` creates a matrix of \( s \) by \( n \) randomly sampled from normal distributions which represents \( s \) plays on \( n \) machines (“arms”)

- function `evalLearn(dset, e, tmax, q0)` “learns” to evaluate feedback for \( dset \) (e.g. a 10-armed dataset generated as described above) by iterating up to \( tmax \) times, with greed factor \( e \).

- function `selectAction(avgr, e)` selects an action that gives maximal reward or an exploratory action, depending on the “greed” parameter \( e \) and function `updateEstimate(old, target, step)` updates the average rewards efficiently.

---

\(^1\)This code was a contribution of Dr. Saturnino Luz
In addition, a sample function \texttt{runExperiment1(...)} is provided which uses the above described code to run the first evaluative feedback simulation described in the notes and plot the results.

3 Exercises

Do the following exercises, to be handed in next Tuesday:

1. run the initial evaluative feedback experiment for different settings of the “greed” parameter ($\epsilon$), number of runs etc. Plot and compare the results. NB: \texttt{runExperiment1()} does all that is needed, all you have to do is pass different parameters.

2. Extend or modify the R code given (or re-implement it in your language of choice) so as to run at least one of the other settings described in the notes: softmax action selection, optimistic initial estimates, reinforcement comparison, etc. Alternatively (if you are not comfortable enough with R) you can simply describe the algorithm in pseudo-code.

3. Describe the robocode game as a Markov Decision process along the lines seen in the last lectures. In particular, outline a scheme for describing:
   - the battle environment as a set of states (note that you will typically need to simplify the state to a few variables, as we illustrated, for instance, in our simplification of the draughts game).
   - actions and
   - a reward structure (for a robocode learner)

References
