TÉCNICO LISBOA

Inteligência Artificial para Jogos

2024-2025

3rd IAJ Project – Reinforcement Learning in a Game

This third project about implementing Learning algorithms integrated with a game environment. For that you will use the same base of the previous project, Sir Uthgard adventure, but this project is less guided and more open-ended. You will have more liberty with what Reinforcement Learning Algorithm and implementations to use and what modifications to the base game are necessary for the RL algorithm to work. The only condition is that you are not allowed to use 3rd party packages. You must code and (if you copy and adapt) understand everything yourselves. So, for instance, for the Neural Network, you must implement the necessary (parameter update) algorithms.

For this second project, you will need to write a report, explaining and justifying the decisions made by your group in terms of implementation, describing performance tests (**including Episode Reward evolution plots**) and corresponding analysis. **Explain the rationale and describe the architecture, parameters and heuristics used**.

You should submit a zip file (named P2 Group XX, where XX is your group number) with both Unity source code and with the report (a pdf/doc file) via Fenix[1], **until 23:59 of Nov 2nd**. I will accept submissions after the deadline, but with a 0.5 value penalty for each hour after the deadline.

Important Note: More than the code written in the project, it is important that you master the different algorithms and learn to work with them. In the project discussion you will be asked to discuss details on the algorithms, their implementations, and/or exemplify on the board or paper how they work.

[1] To reduce the Project's size, before creating the zip delete the Library folder from the project!

Level 0 – Preparing the Game:

- Download Project 3 from Fenix; there are a few additions to the Character Control interface and a couple of extra Properties you will need. For instance, the AutonomousCharacter has now a property Reward, to receive Rewards from the environment (in our case, the GameManager); As an alternative, you can use your own Project 2 and make the adaptations yourselves.
- If using the new project, add the new hero actions, and monster State Machines you developed for Project 2.
- Because now the game will have to run automatically many times, you will have to make changes and move the re- initialization code from the Awake() and Start() Methods to other Methods that are called when the character either wins or dies (a basic version of this is already done in the new project, but you can change and improve it).
- To speed up runs you can go to Services->General Settings->Time and change the Time Scale. But be careful that the value does not mess up other things, like the physics…

Level 1 – Implementing Tabular Q-Learning: The Table (3 points)

- You have inside DecisionMaking a new empty folder named "RL" for the new classes. With the game prepared, you must now implement all necessary classes and methods for tabular Q-Learning. As with the other decision making algorithms, it should implement a ChooseAction() method to be called by the AutonomousCharacter. But it should also have a way to learn from the results of its actions. There must be a class for storing the learned parameters, in this case, a table.
- You might want to create more abstract TQLStates (what Sir Uthgard observes) and/or TQLActions to represent the state and actions for the RL Algorithm. For instance, it might be good to the reduce all the possible HP values (1 to 40) Sir Uthgard can have to just 3 or 4 possibilities (e.g. VeryLow, Low, OK). If you do this, you have to implement conversion between States and RLstates, and from RLActions to the actual game actions.
- For RLState, you must also think what parts of the game state are relevant, and what parts are less relevant and might be ignored. Do not forget to justify your choices in the report.
- The table dimensions will be the number of different states you consider times the number of actions. Is this number reasonable? How many different pairs state-action does Sir Uthgard see in a typical Episode (run)? How many episodes do you think you will need to train? Justify your numbers in the Report.

Level 2 – Implementing Tabular Q-Learning: The Algorithm (2 points)

- You should implement now the algorithm that chooses the actions using the Q-Table and then updates the table with the received rewards using the Q-Learning formula… do some tests and check it is working.
- The Canvas has lots of places you can use to Print out feedback. Use them!

Level 3 – Implementing Tabular Q-Learning: Training and Results (5 points)

- You should think also about the reward. While theoretically it could be enough to give a reward at game's end (e.g -100 for losing and +100 for winning), this might make life hard for Sir Uthgard. So, you might try to give some small partial rewards for actions that contribute to winning, like picking up chests and levelling up. Rewards should be given by the GameManager when it ends processing each action (see example in SwordAttack)… **Note that rewards shouldn´t be over-engineered**! The idea of RL is that the system learns by itself from painful experience…
- To train the Model, select the Dungeon **without** the extra formation orcs, a non-stochastic world, and choose from all the Lab implemented actions, which ones you will allow Sir Uthgard to use.
- Let it run a few hundred times and see if learned something. You should implement a way to store on disk the learned parameters, so that you can load a trained "brain" and continue training or test its performance, and not start over every time.
- **Do not forget to log the performance of the algorithm as training goes on for the report, both the reward evolution, and other KPIs (Gold accumulated, time survived, etc).**
- If Sir Uthgard is having problems learning you can make it easier for sir Uthgard (Sleeping Enemies), and reduce the initial actions available, and then add complexity…

Level 4 – Implement a Neural Network inside Unity (3 Points)

- Implement a generic Neural Network class, capable of supporting a simple Feed Forward Neural Network, but with any number of layers, any number of neurons per layer, and supporting different activation functions.
- The class should include methods for initialization, the Forward pass, and for adjusting the weights by backpropagation, both for supervised Learning, as for policy learning.

Level 5 – Implement and test a Neural Network – based RL algorithm (5 Points)

- You can implement any algorithm of your choice. For instance implement the classical Policy Gradient algorithm REINFORCE, using a Neural Network to represent the Policy. Another option is to implement a Deep Q-Learning algorithm, where the table is substituted by the Neural Network.
- Note that you have to decide again on the state representation to feed the NN for the new algorithm. What are the advantages / disadvantages viz the Tabular representation you used before? Discuss this in the Report.

Bonus Level – Implement a 3rd RL algorithm or a variant of the above ones:

- Implement another RL algorithm, or a variant of the first one, and compare their performance.

TÉCNICO LISBOA

Project Report (2 Points)

When submitting the project, you should also submit a report. The report can explain the algorithms used, but should mainly focus on describing the several steps/stages of your project. You should describe your reward function/observations/actions or changes to the game. You should also describe the several iterations of the scenario, including information about the training process that you used to inform your decisions, and describe the changes you made to hyperparameters (and why), how many games were necessary, etc…. You should also specify the final version of the algorithms used with all hyperparameter values. There is no page limit in the size of the report, but please be reasonable. Note that the 2 points of the report include only the overall quality and completeness. The report also contributes to the grades of the other parts of the project, so please give it the consideration it deserves.

Submission details

You should email a zip file with the report and with the Unity Project containing your work. It should include both the trained "brain"s and the option to train again. The submission should be done via fenix), until 23:59 of Saturday November 2nd.

I will accept submissions after the deadline, but with a 0.5 value penalty for each hour after the deadline.

Note: Project Discussions will happen the following week, on Thursday November 7th. If the group has an impediment that day, please let me know beforehand.