Report Title

Author Name1[*] and Author Name2[**]

[*]Dept. of Information Studies
[**]Dept. of Physics & Astronomy
University College London, WC1E 6BT
{*email1,email2*}*@ucl.ac.uk*

April 30, 2020

Abstract

The abstract text goes here. Here is a bit more of my abstract.

1 Introduction

Here is the text of your introduction.

Here is a simple equation

$$\alpha = \sqrt{\beta} \tag{1}$$

I can refer to my equation by saying look at Equation (1).

Here is a multi-line equation:

$$\begin{aligned} f(\mathbf{x}; \mathbf{w}) &= \sigma(\mathbf{w}^T \mathbf{x}) \\ &= \frac{1}{1 + e^{-\mathbf{w}^T \mathbf{x}}} \end{aligned} \tag{2}$$

I can point people to literature like this [3] (a conference paper) or this [2] (a journal paper), or even this [1, Sec. 4.1] (a book, with section indicated). See the file `report.bib` for how these bibliographic entries are defined.

It helps to have a convention to the font used for particular mathematical objects. A common convention is: for simple values use lower case latin characters s (this might include observed values in probabilistic work); for sets use caligraphic upper case $\mathcal{S}$; for vectors of values use bold latin $\mathbf{s}$; for matrices use upper case latin S; and for parameters (including parameters in probabilistic models) use lower case greek letters (scalar), σ, bold greek (vectors), $\boldsymbol{\sigma}$, and possibly upper-case greek for matrices of parameters Σ. However, it is up to you to develop your own style.

1.1 Subsection Heading

Write your subsection text here. Here I am referencing Section 1. Figures and tables can be inserted and referred to, e.g. Figure 1 and Table 1.

	Interested	Understand
Linear Regression	N	Y
Neural Networks	Y	N

Table 1: Some results as a table.

Figure 1: Simulation Results

1.2 Quoting and Emphasis

When you quote or emphasise things, you have a few options. I tend to put things in italics, e.g. Luke told me, *Put it in italics!*. If you prefer to put it in quote-marks then do so, but make sure you use left and right quotes properly, e.g. Luke told me, "Use left quotes to open, and right quotes to close!".

2 Background

Your background section text goes here...

3 Method

3.1 Game Mechanics

We have decided to follow the exact game mechanics in the actual game with no reduction and simplification in the rules. This means that we will always start with a random board with 2 tiles with values on each of the tiles "2" or "4". Then, after each valid move, a new tile with values "2" or "4" will be generated on a random empty tile. We loose when no valid moves are allowed.

3.2 Representation and Q-value Calculation

In our experiment, we have tried 2 types of representation, which are the simple representation and the relationship representation. They will be explained as follows. To calculate the q-value, we will use the linear approximation approach for simplicity.

3.2.1 Simple Representation

This is the simplest type of representation of our game, which is just simply a vector of 16 elements with each element representing the values of the individual tiles. For example, if we look at figure 2, the state will simply be:

$$\phi = (0, 0, 2, 4, 0, 0, 4, 8, 0, 2, 16, 32, 0, 2, 2, 16)^T$$

To calculate the q-value, we will use our state vector acting on the weight vector representing each action, which can be written as:

$$q_a = \phi^T w_a$$

where q_a is the q value of that action and w_a is the weight of that action.

Figure 2: An example game board. This image is obtained from Wikipedia.

3.2.2 Relationship Representation

Reading Barto's "Reinforcement Learning" book, I have learnt that, to improve my learning, my state vector needs to have information involving interaction between the tiles. In addition to that, I have also decided to use the book's convention where the action dependency will be put in the state instead of the weight vector. This means our Linear Approximation equation can be re-written as:

$$q_a = \phi(a)^T w$$

When we play the game, whenever we consider a horizontal move, we always compare the tiles horizontally. This means the interaction considered will be horizontal. We will represent the interaction by the difference between 2 tiles. Our state vector for move left will then be written as:

$$\phi(L) = (\text{values of all 16 tiles}, \text{horizontal differences between each tile of each row})^T$$

To distinguish move "L" and "R" the horizontal differences in $\phi(R)$ will have opposite sign to that in $\phi(L)$. $\phi(U)$ and $\phi(D)$ can be calculated with similar method except we will consider vertical differences instead.

3.3 Actions

As explained before, there are 4 available actions for the agent to choose from. They are "Left", "Right", "Up" and "Down". Our agent will choose the action by calculating the q-value of each action. Since there are lots of possible traces in this game, we choose the action using the epsilon-greedy policy to allow exploration by the agent. The epsilon-greedy policy is:

$$\pi(s,a) = \begin{cases} 1 - \epsilon + \frac{\epsilon}{|A|}, & a = argmax_{a'}(q(s,a')) \\ \frac{\epsilon}{|A|} & \text{otherwise} \end{cases} \tag{3}$$

where $|A|$ is the number of available actions the agent has. In our case, $|A|$ will be most likely to be 4. As mentioned above, there are a huge possibility of traces in this game. To ensure we don't waste time exploring invalid moves, we have decided to restrict our agent to only choose from valid moves. This means if "Left" is not a valid move, we will only have 3 actions available for the agent to choose from and $|A| = 3$ in this case instead assuming the other 3 moves are still valid.

From multiple sources, I have also seen that instead of using a pure epsilon-greedy policy, people tend to use a decaying epsilon-greedy policy. This allows exploration at the beginning but it decreases the freedom of exploration as we get to the later episodes of the learning process. We have implemented this by a rather naive step-wise decaying scheme.

3.4 Reward Schemes

Multiple reward schemes has been attempted and they both have their merit and drawbacks. We started off with a simple reward scheme where it will reward the value of the greatest value on the board after a move.

For example, if after a move we obtain a board state shown in figure 2, we will get 32 points.

Another reward scheme we have attempted will be the merge-reward scheme, where the agent will be rewarded whenever we merge two tiles into one. The reward will be the value of the new tile. For example, if the initial board is that shown in figure 2 and we move right, the two "2"s in the bottom role will merge and give a new tile "4". This will give us 4 points.

3.5 Learning

We have mainly used the q-learning equation when updating the weight vector, which is:

$$\begin{aligned} \Delta w_a &= \alpha(R_{t+1} + \gamma \text{max}_a(S_{t+1}(a), w_{t+1}) - q(S_t; w_t)\nabla_w q(S_t; w_t)) \\ &= \alpha(R_{t+1} + \gamma \text{max}_a(S_{t+1}(a), w_{t+1}) - q(S_t; w_t)S_t)) \end{aligned} \tag{4}$$

where we have taken $\alpha = 0.0000001$, $\gamma = 0.9$.

We used the decaying-epsilon-greedy policy as explained above with $\epsilon = 0.1$ initially. Finally, we have decided to play the game with at least 1000 episodes.

4 Results

As mentioned above, we tried 2 types of representation and have chosen to do each learning session in increments of 1,000 episodes starting from 1,000 episodes to 10,000 episodes. This was to track the progress of the algorithms and to see if there were any learning trends. The results for them will be shown below.

4.1 Simple Representation

The results from 10,000 episodes of the simple representation is shown below.

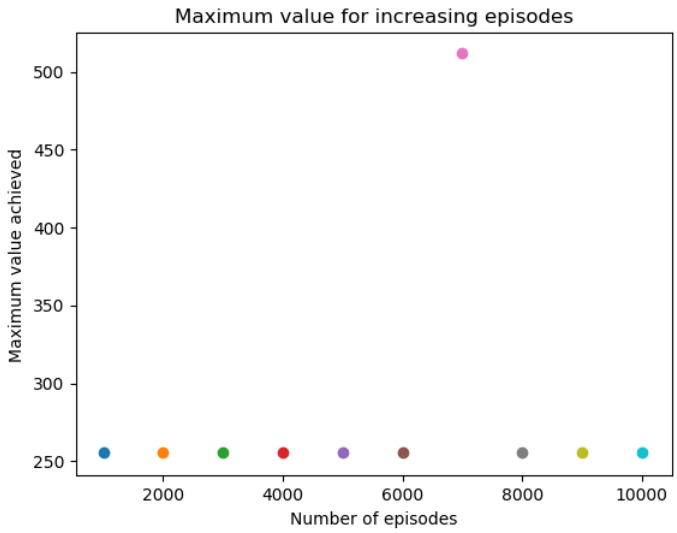

Figure 3: Results from simple representation with increasing episodes

From the results shown in figure 3, it can clearly be seen that this representation is not very effective at making our agent learn how to play 2048. However, the agent was able to reach the 256 tile quickly and consistently, which could mean that the agent was either able to visit the earlier states more often, therefore getting more accurate q-values for certain states and actions, or simply chose lucky actions. Furthermore, reaching the 512 tile only once out of 10 increasing episodes shows that the tile was reached with a high level of luck rather than knowledge of the action. We have two main reasons to believe why this representation was not capable of teaching the agent 2048. The first one being that the representation did not have enough

information on how the tiles relate to each other, thus not allowing the agent to choose actions based on the certain factors of the board. The second one being that there are too many combination of states and that the agent was not able to visit all the states in the amount of episodes provided.

4.2 Relationship Representation

The results from 10,000 episodes of the relationship representation is shown below.

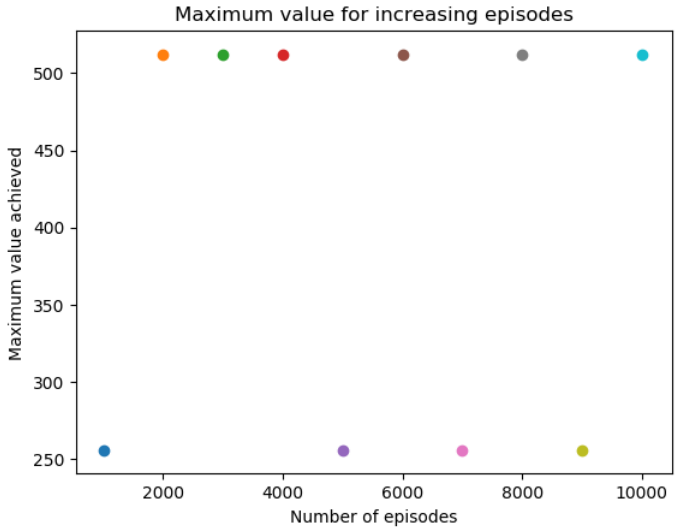

Figure 4: Results from relationship representation with increasing episodes

The results from figure 3 show that this representation is evidently more effective in learning 2048 compared to the previous representation, given the fact that it was able to reach the 512 tile with less training episodes and more consistently. However, it can also be seen that the maximum value achieved by this representation is the same as the previous representation, demonstrating that this representation was also not able to effectively teach the agent 2048. We believe that adding additional information such as interaction between tiles in the state vector has improved both the learning rate and consistency of the agent. However, the problem mentioned in the previous experiment still remains. This representation is still not able to account for the fact that there will be an impossibly large amount of unique states for the agents to visit, implying that the agent may not be able to visit every state in a feasible amount of time. On the other hand, it could be argued that there was not enough training time for the agent to completely learn 2048. Perhaps the agent would've able to learn 2048 if a larger amount of episodes were used to train the agent?

5 Discussion

Your discussion section text goes here...

Declaration

This document represents the group report submission from the named authors for the project assignment of module: Foundations of Machine Learning and Data Science (INST0060), 2018-19. In submitting this document, the authors certify that all submissions for this project are a fair representation of their own work and satisfy the UCL regulations on plagiarism.

References

[1] Christopher M. Bishop. *Pattern Recognition And Machine Learning.* Springer-Verlag New York, 2006.

[2] Tom Fawcett. An Introduction to ROC Analysis. *Pattern Recognition Letters*, 27:861–874, 2006.

[3] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems (NIPS)*, pages 3111–3119, 2013.