Training the Q-learning Algorithm to Solve a Maze Problem

Axil Sudra and Abdirahman Khanog

1 Define a domain and task

Our chosen domain for this piece of work will be on an Agent solving a 10×10 maze/labyrinth. The word labyrinth has its origins in ancient times and as such is surrounded by an air of romance and mystery [1]. Generally the term *labyrinth* is used extensively in Egyptian and Cretian ancient writings and the term *maze* is used with regards to a puzzle-solving problem such as a *hedge-maze*[1]; both words can be used interchangeably.

A reinforcement learning solution to a problem involves the agent taking actions to arrive at a new state, this process is optimized by rewarding the agent. When deciding on an action to take, the agent must consider the next possible states and their associated rewards.

The agent in our reinforcement learning problem will have a delayed reward, that is the agent will take actions that either result in a reward of -1 or 0 and a final delayed reward of 100 at the goal-cell; in addition, the maze will have non-entry cells that the agent cannot enter but have no associated penalty/reward. The maze is more challenging as it has non-entry cells as well as -1/0 reward cells. Conventional programming methods tackle this problem by setting non-entry cells to a reward of 0 and routes on the shortest path with a higher reward, this becomes a problem of finding the shortest route by maximizing the acquired rewards along the way. The optimal solution to our problem does not involve finding the shortest route but avoiding occupied and -1 cells and hence maximizing a final delayed reward.

The maze is pseudo-randomly generated for each iteration and in some cases a maze that is impassable is generated, that is, a maze with no open cells that leads to a final reward. Hence, we needed a random seed to reproduce a solvable maze.

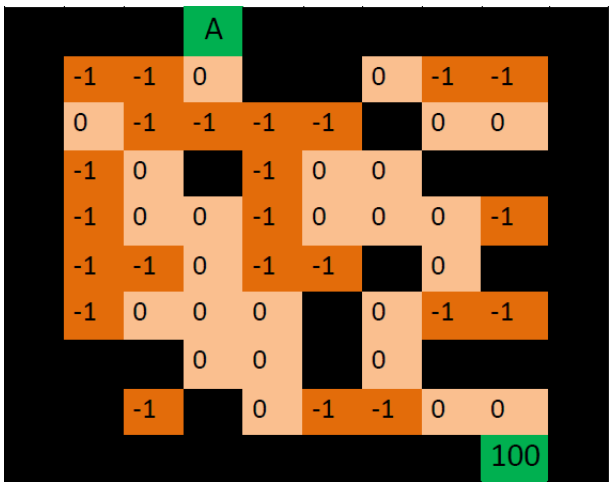

Figure 1: Maze

2 Define a state transition function

Figure 2 below shows 6 solutions to the maze problem. The agent which is marked with an 'A' occupies the entry point to the maze. The arrows only show direct paths to the exit of the maze; note that the

agent is allowed to move in all directions (left/right/back/forward). The reward is marked '100'; this is the final reward received at the exit of the maze.

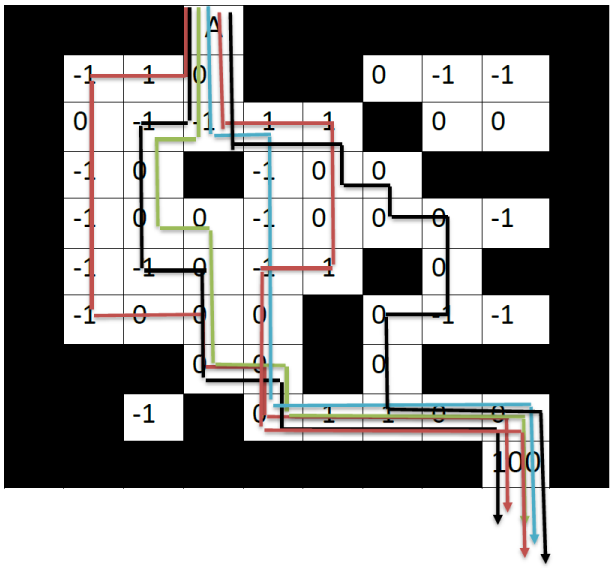

Figure 2: 6 direct paths to maze exit

Given a finite set of states S and a finite set of actions A, the agent observes the following at each discrete time step: $\forall s \in S$ and $\forall a \in A$. The state transition function is $S_{t+1} = \delta(S_t, a_t)$; the agent does not know δ but only knows S_{t+1} and the next state is determined by the agent's current state and action it takes. At the starting point the agent can only make a move/action down to a cell with 0-reward. Figure 3 gives the state transitions for our agent where columns represent each state. The game is terminated when the agent reaches the 100 reward at the maze's exit.

St1	St2	St3	ST4	ST5	ST6	ST7	ST8	ST9	ST10	ST11	ST12	ST13	ST14	ST15	St16
0	-1	-1	0	-1	-1	-1	-1	0	0	0	-1	-1	0	0	100
0	-1	-1	0	0	-1	0	0	0	0	0	-1	-1	0	0	100
0	-1	-1	0	0	0	0	0	0	0	0	-1	-1	0	0	100
0	-1	-1	-1	-1	-1	0	0	0	-1	-1	0	0	100		
0	-1	-1	-1	0	0	-1	-1	0	0	0	-1	-1	0	0	100
0	-1	-1	-1	0	0	0	0	0	-1	0	0	-1	0	0	100

Figure 3: state transition table

The environment of our agent is deterministic and static [2], that is our agent can predict the next state given the current state. Also Our environment is fixed and is therefore discrete [2]. The optimal solution would be to avoid -1 penalties and reach the exit point.

3 Define a reward function

The reward function is defined by $r_{t+1} = r(S_t, a_t)$; this is the reward obtained at the new state by making a move/action from the current state. The agent in this case can either obtain a reward of 0 or -1. The outer parameter of the maze is non-accessible and the environment becomes a 9-by-9 grid. The optimal route is achieved by maximizing a final reward. There are 6 direct routes that contain the optimal solution in figure 4. The reward matrix shows the cumulative reward at each state.
It is clear that route 3 gives the highest reward. This route is shown in Figure 2 by the green line.

	Route 1	Route 2	Route 3	Route 4	Route 5	Route 6
St1	0	0	0	0	0	0
St2	-1	-1	-1	-1	-1	-1
St3	-2	-2	-2	-2	-2	-2
ST4	-2	-2	-2	-3	-3	-3
ST5	-3	-2	-2	-4	-3	-3
ST6	-4	-3	-2	-5	-3	-3
ST7	-5	-3	-2	-5	-4	-3
ST8	-6	-3	-2	-5	-5	-3
ST9	-6	-3	-2	-5	-5	-3
ST10	-6	-3	-2	-6	-5	-4
ST11	-6	-3	-2	-7	-5	-4
ST12	-7	-4	-3	-7	-6	-4
ST13	-8	-5	-4	-7	-7	-5
ST14	-8	-5	-4	93	-7	-5
ST15	-8	-5	-4	N/A	-7	-5
St16	92	95	96	N/A	93	95

Figure 4: Cumulative reward matrix

4 Optimal Policy Selection

We have to find an optimal policy for the agent that incorporates a balanced exploration and exploitation parameter. The agent enters the maze environment not knowing anything and has to learn; its learning rate α is preset and controls the rate it learns at which ranges from 0 to 1. It is akin to setting the number of steps taken to reach the final exit cell. The agent must consider immediate and future rewards for an optimal solution, in our case the future reward obtained at the exit is very important. The learning policy is π, where an action $\alpha = \pi(s)$, this is an action taken giving a current state s under a chosen policy π.

The optimal policy is found by finding π^* such that the return, that is the long term performance $\mathrm{R}^{\pi} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...$, is maximized. An ϵ parameter is set which is the greediest policy and represents the agent's exploration. The agent at the beginning will choose actions at random from the set of all possible actions with probability ϵ in order to explore or the agent will exploit from what it knows with probability 1-ϵ, which is referred to as greedy action as agent will exploit actions with the highest rewards. Note that $0 \leq \epsilon \geq 1$; for ϵ close to 1 the agent will explore and for ϵ close to 0 the agent will exploit.

As the agent moves through the maze and acquires more knowledge of the environment and receives more rewards, exploration should decrease. By default after every step, the ϵ value decreases as it is multiplied by 0.99999 if $\epsilon \geq 0.5$ or it is multiplied by 0.9999 if $\epsilon < 0.5$. This ensures that ϵ decreases gradually.

The Q-learning agent's world consists of two matrices: the R-matrix and the Q-matrix. The agent can only operate in the R-matrix environment which represents states, actions and their associated rewards. The Agent only knows immediate actions and their associated rewards. The Q-learning algorithm is an on-line learning algorithm as it processes inputs in a serial fashion, that is the agent in the maze receives reward/penalty and enters a state one at a time. Q-learning is also an off-policy method because it learns about a greedy policy while following a policy involving explanatory actions.

5 Graph and R-matrix

The R-matrix displays all valid actions, states and rewards which makes the matrix a representation of both reward and state-transition functions. The reinforcement learning problem is defined by the underlying Markov Decision Process. The matrix in Figure 5 represents the state/action diagram and the

instant reward values. Blank cells are non-entry cells which correspond to hedges in our maze problem. The agent does not know about the reward-matrix and the optimal route to take, by finding optimal parameters the agent learns its environment and hence solves the maze.

In figure 5, the goal-cell can only be accessed via an action from state 10 to state 9 and also the outer perimeter is non-accessible, that is it has hedges. Figure 6 is a graphical representation of figure 5, the coloured arrows represent actions and the numbered circles represent states. For state-action-state such as; $2 \rightarrow 2$ is represented by a circle with black-shaded outline, this means there is no hedge at that point. A bidirectional arrow means that state-action-state, such as; $2 \rightarrow 3$ and $3 \rightarrow 2$ exist whereas uni-direction arrow, such as; $7 \rightarrow 3$ means that only one direction exists. A green-shaded lines represents cell values of 0 and red-shaded lines represent cell values of -1 where line is bi-directional then same colours as above have been used and for cases such as; $2 \rightarrow 3$ with cell value -1 and $3 \rightarrow 2$ with value 0 then colour used is purple, the order of 0 and -1 is not relevant.

Actions

States	1	2	3	4	5	6	7	8	9	10
1										
2		-1	-1	0			0	-1	-1	
3		0	-1	-1		-1		0	0	
4		-1	-1			0	0			
5		-1	0	0		0	0	0	-1	
6		-1	0	0		-1		0		
7		-1	-1	0			0	-1	-1	
8				0			0			
9			-1		0	-1	-1	0	0	
10									100	

Figure 5: Reward matrix

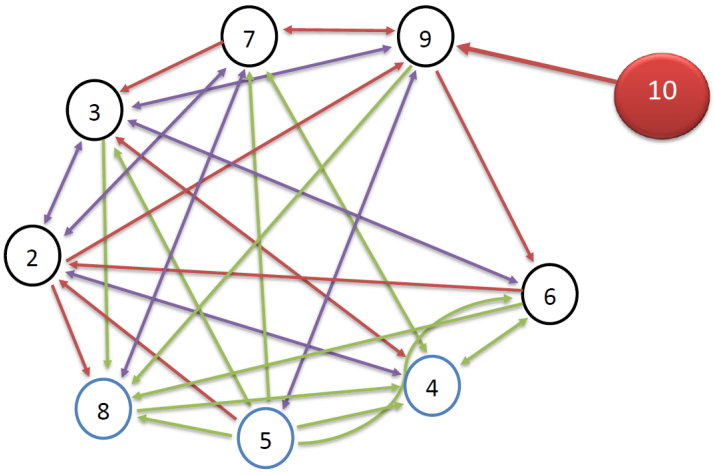

Figure 6: Maze graph with valid transitions and states

6 Q-learning

Given a strategy wandering in a Markov Decision Process, Q-learning which is an off-policy method would run perfectly on top of this strategy [3]. The Q-matrix is the brain of the Q-agent. The agent's goal is to estimate the relative merits of actions which lead to the best long term outcome; this process is measured and quantified by a final reward. In the case of our maze problem, the agent must find the route that maximizes a final reward. Rows in a Q-matrix represent actions and columns represent states. The agent enters the maze not knowing anything, i.e. zero Q-matrix. As the agent increasingly experiences the environment, the Q-matrix in figure 7 will be updated via the transition rule, this transition rule is a very simple formula; $Q_{new}(s,a) = Q_{old}(s,a) + \alpha[(r(s,a) + \gamma \max Q(s_{next}, a_{all}) - Q_{old}(s,a)]$, the term in the square bracket is the error term.

In [3], a MDP is defined as; 4-tuple (S,U,P,R). S is the set of the states, U is the set of the actions and $U(i)$ is the set of actions available at state i. $P_{i,j}^{M}(a)$ is the probability of transitioning from state i to state j under action a $\in$ U(i) in state i. $R_M(s,a)$ is the reward received for state s and action a.

The agent is trained to eventually exploit, this is made possible by the inbuilt decaying factor applied to ϵ. Also, initially the agent will explore and thus populate the Q-matrix with valuable information which in turn will cause the agent to begin exploiting more and more as learning grows. For an optimal policy to be carried out, the Q-learning algorithm has to have a corresponding set of values that the agent can follow.

6.1 Q-learning Parameters

Learning Rate (α): As the agent begins exploring the environment, it makes estimates of the values for each state and this estimate is updated once the agent visits that state again[4]. This cycle continues and gives two values for a state, i.e. Q_{old} and Q_{new}, and the difference between old and new is the error term in brackets, that is; (r(s,a)+γmax Q(s_{next},a_{all})-Q_{old}(s,a)][4]. We will call this error term, δ. Each state's value is updated by $Q_{new} = \alpha\delta$ and hence α; this term becomes important as it controls the magnitude to which the agent takes the error estimates into account[5]. It is akin to step sizes and tells us the extent to which new information should override old information. α can only takes values in $[0, 1]$. For $\alpha = 1$, Q_{old} is replaced by Q_{new} and the agent will explore, for $\alpha = 0$ the agent will exclusively exploit[5].

Discount Factor (γ): The discount factor, γ controls the importance put on future rewards. For $\gamma = 0$, the agent puts no value on future rewards and only considers immediate rewards; this is termed 'myopic' [5]. For $\gamma = 1$, the agent will value delayed rewards highly [5].

Parameter	Starting Values
α	1
γ	1
ε	0.8
π	Greedy-Policy

Figure 7: Initial Q-learning Parameters

7 Learning Episode

We begin a learning episode by checking the R-matrix for available actions, depending on chosen policy, we select an action. Then we observe the state and reward as a consequence of action chosen. The Q-matrix is checked for $Q_{old}(s,a)$ and $Q(s^{'}, a^{'})$, the latter is an estimate. The Q-learning formula is used

to calculate the new $Q(s,a)$ and Q-matrix is updated with this value. The agent will learn without any help, it Will go from state to state until completion. Each completion or in our case exit to maze is called an episode/training-session. The optimal route will be shown on the Q-matrix and for state-diagram and associated rewards we will refer to the R-matrix.

7.1 Episode 1

We will present two fully worked examples of the Q-learning Algorithm. Our agent will move around the R-matrix in figure 5. The blank cells are non-entry cells and at the start agent can only move in one direction, namely; [0,3] → [1,3] which carries a 0 reward. Also the matrix is modelled as 0-9 rows by 0-9 columns, outer perimeter has hedges except the exit-cell which has final reward. figure 7 has our initial parameter values.

We will use the Q-learning Algorithm; $Q_{new}(s,a) = Q_{old}(s,a) + \alpha[(r(s,a) + \gamma \max Q(s_{next}, a_{all}) - Q_{old}(s,a)]$. As the agent knows nothing about environment, it will have to explore. As mentioned above, agent will move from [0,3] → [1,3].
$Q_{new}(1,3) = Q_{old}(1,3) + 1[(r(1,3) + \max Q(2,2), Q(2,3), Q(8,5), Q(8,6) - Q_{old}(1,3)]$
$Q_{new}(1,3)$ = 0 + 100 + [-1+-1+-1+1] - 0 = 96. This tells use from position [1,3] the highest reward obtainable is 96 for $\alpha = 1$ and $\gamma = 1$, this route corresponds to green line in Figure 2. Given a starting point in the maze, i.e. Q[0,3] → Q[1,3], the algorithm calculates with greedy policy the highest reward obtainable from [0,3] → [1,3] → [9,8]. The full route is shown on figure 2 and Q-matrix, for the calculation above, cells with 0 reward not included in Q-formula.

Actions

States

	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	96	96	0	0	0	0	0	0
4	0	0	96	0	0	0	0	0	0	0
5	0	0	96	96	0	0	0	0	0	0
6	0	0	0	96	0	0	0	0	0	0
7	0	0	0	96	0	0	0	0	0	0
8	0	0	0	96	96	0	0	0	0	0
9	0	0	0	0	96	96	96	96	96	0
10	0	0	0	0	0	0	0	0	0	0

Figure 8: Q-matrix

7.2 Episode 2

If we choose a different point on the maze, it is also evident that if agent is in position Q[2,3] then highest reward is only obtained by following above path, i.e. green line in figure 2 which greedy policy successfully does.
$Q_{new}(2,4) = Q_{old}(2,4) + 1[(r(2,4) + \max Q(2,5), Q(7,7), Q(8,6) - Q_{old}(2,4)]$
$Q_{new}(2,4)$ = -1 + 100 + [-1+-1+-1] - 1 = 95.this is the route from [2,3] → [2,4] → [9,8]. Figure 9 is the updated Q-matrix, the route with reward 95 represent route in black-line in figure 2.

8 performance vs. episodes

In this section, We will test the performance of Q-learning algorithm by changing 4 parameters; α, γ and ϵ, we will also see the effect epoch number has on algorithm. I will observe the effect each parameter has

Actions

States	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0
3	0	0	96	95,96	95	95	0	0	0	0
4	0	0	96	0	0	95	0	0	0	0
5	0	0	96	96	0	95	95	95	0	0
6	0	0	0	96	0	0	0	95	0	0
7	0	0	0	96	0	0	95	95	0	0
8	0	0	0	96	96	0	95	0	0	0
9	0	0	0	0	96	96	95,96	95,96	95,96	0
10	0	0	0	0	0	0	0	0	0	0

Figure 9: Updated Q-matrix

on the overall performance of model whilst keeping the remaining parameters constant. I will gradually increase each parameter in turn and once model cannot be improved, I will make note of this and move on to the next parameter. Epoch will be set to 500 then a 100.

8.1 Test-Value α

We will test three values, 0.5, 0.7 and 0.9. All other values kept at 0.5. For $\alpha = 0.5$, the agent does not converge but wanders endlessly. Also it was common to observe an agent with -60,000 accumulated rewards. Since the maze has -1 and 0 cells, -1 rewards are only possible roughly 1/2 the times, this means agent with -60,000 has made 120,000 pointless moves.

For $\alpha = 0.7$, agent has improved in avoiding -1 cells but has not yet converged and is still wandering, this is due to the reward being delayed and γ is too low for agent to head in the direction of exit-cell. Finally, we will check if an $\alpha = 0.9$ will make a difference; This has made no difference, except that the step sizes have grown and agent acquires larger negative rewards in less time.

We will check if epoch-size has a positive effect on convergence, i will increase from 500 to a 1000. The only improvement is agent avoid -1 cells but still no convergence. This improvement in increasing the reward is due to the higher iteration that causes agent to learn somewhat about what to avoid.

8.2 Test-Value γ

At $\gamma = 0.7$ and $\alpha = \epsilon = 0.5$, algorithm converged immediately and the agent took the route with highest reward. The increase in epoch from 500 to 1000 had no impact whatsoever. Increasing γ to 0.9 had a detrimental effect as the agent took the worst route available with the least reward. This can be due to the optimization problem when finding policy π[14], the optimization problem has an infinite time horizon and the purpose is to maximize the discounted reward, this discount refers to the $\beta \leq 1$, which is said not to converge if this condition is broken but in our case 0.9 is close to 1 and this causes the agent to find non-optimal route.

8.3 Test-Value ϵ

At $\epsilon = 0.7$ and $\alpha = \gamma = 0.5$ algorithm does not converge. At $\epsilon = 0.9$, nothing changes except an increase in exploration that results in wandering. Increasing epoch size has no effect.

8.4 Optimal Parameters

: The solution to our maze problem is solved by finding a balance between the parameters; α, γ and ϵ and by setting the number of full cycles agent goes through the maze. The optimal parameters are; $\alpha = 0.6$, $\gamma = 0.7$, and $\epsilon = 0.8$ with epoch ≥ 8.

9 Evaluation of Results

9.1 Quantitative Analysis

Searching for the optimal parameters of the Q-learning algorithm in our maze problem, we identified that given a non-optimal gamma (γ) and to a lesser extent a non-optimal epsilon (ϵ), a high value of alpha (α) increases the error terms in the Q-learning algorithm. Given the optimal parameter gamma (γ) and to a lesser extent alpha (α) and epsilon (ϵ) the number of epochs can be significantly reduced whilst still converging. On the other, with a non-pareto-optimal combination of parameters, we found that the agent wanders infinitely.

9.2 Qualitative Analysis

We observed that the training of the agent in the maze is very important, as with a low number of epochs and non-optimal parameters the agent gets stuck. Modelling a maze problem by placing all the rewards at the exit, in our opinion it substantially reduces the learning of the agent.

10 Q-learning and Associative Learning Theory

10.1 Parallelisms Between the Q-learning algorithm and Error Correction Models in Psychology

Past error correction models developed in the field of Associative Learning (AL) have had a significant influence on the structure of the Q-learning algorithm [6]. One of the most notable developments in AL that led to improvements in error correction models was Pavlov's classical conditioning (also known as pavlovian conditioning) experiment [7] in psychological studies; this involved a learning process, although no decision process. The experiment exposed dogs to the paired stimuli of a bell (conditioned stimulus) and food (unconditioned stimulus); following some training, it was observed that the dogs would salivate to the ringing of the bell even if it was present without food. This is known as a conditioned response whilst salivation to only food is unconditioned response. The conditioned response from the dogs suggested how they learned to predict reward through experience. Based on these observations, it has been established that classical conditioning is based on a comparison between what reward the dogs experience on a particular trial, and what reward they had expected on the basis of its previous learning [8]; the difference in comparison is called the prediction error.

A significant error correction model developed from the classical conditioning experiment conducted by Pavlov [7] is the Rescorla-Wagner model. It is used to explore how animals learn stimulus-reward relationships in a trial with multi-conditional stimuli (i.e. a light and bell) and single unconditional stimulus [9]. The Rescorla-Wagner model also demonstrates the concept of 'blocking', which is essentially when learning from one stimuli reduces (or gets blocked) as a result of new learning from another stimuli [10]. The model attempts to measure the associative strength, denoted by V, between each conditioned stimuli and the unconditioned stimulus. Mathematically, the error correcting model is described as $V_{new}(CS_i) = V_{old}(CS_i) + \eta[\lambda_{US} - \sum_i V_{old}(CS_i)]$, where CS_i is the ith conditional stimuli in the trial and US is the unconditional stimulus. [11]. The Rescorla-Wagner model is sophisticated in the way that it describes the associative strength of conditional stimuli through a simple error correcting rule (V), however the conditioning trial of the model is a discrete temporal object and it does not account for second order conditioning [11].

The Rescorla-Wagner model's problems are addressed by the Temporal Difference Learning (TD) rule presented by Barto and Sutton [12]; TD learning essentially attempts to estimate the value of 'states' (in time) in terms of the future rewards or punishments that they predict [11]. The TD learning rule is given by $V_{new}(S_{i,t}) = V_{old}(S_i, t) + \eta[r_t + \gamma \sum_{S_k @ t+1} V_{old}(S_{k,t+1}) - \sum_{S_j @ t} V_{old}(S_{j,t})]$, where $S_{i,t}$ denotes state i at

time t [11]. In comparison to the Rescorla-Wagner model, the TD rule calculates the associative strength of the stimuli to predict some immediate reward at time t and also the future (predicted) accumulative reward resulting from a specific state.

The Rescorla-Wagner model and the Temporal Difference Learning (TD) rule (which is an extension of the Rescorla-Wagner model) are considered as significant error correction models in psychology. The Q-learning algorithm is considered a 'off policy' algorithm for TD learning as stated by Watkins and Dayan [6]; this means that the the agent can learn different policies for behaviour and value estimation through actions that can be completed from a given state. Both Q-learning and error correction models in psychology aim to learn associations either through stimuli (in the case of error correction models in psychology) or environment (in the case of Q-learning). Predictions are also learned by animals (and humans) through stimuli experience in error-driven learning paradigms such as the Rescorla-Wagner model which calculates the error between the predictions and actual values; Q-learning has a similar mechanism of predicting outcomes (rewards) for specific states and actions that can be taken at a specific point in time.

10.2 Proposal of Implementing Error Correction Models as Reinforcement Learning Architectures

Reinforcement learning involves an agent learning an environment (i.e. states) that corresponds to appropriate actions in respect to that environment [13]; this is essentially known as a policy. The policy of a learning algorithm outlines the actions the agent can (and will) take in various states; this is usually adjusted through a reward function. As outline by Sutton [13] there are many available reinforcement learning architectures (e.g. policy-only, Q-learning, etc.) that can be implemented depending on the type of problem to be solved. In this subsection, we propose a learning architecture that uses error correction model methodology to calculate prediction errors (i.e. maximize the accumulative rewards, also known as the return) from specific states and actions that can be taken; more specifically we will focus on formalizing Pavlov's classical conditioning experiment with the use of this learning architecture.

As explained above, Pavlov's classical conditioning experiment investigated the associations between conditioned stimuli and unconditioned stimulus. From Pavlov's experiment, it was observed that dogs salivate to conditional stimuli (bell) and not the actual unconditional stimulus (food). A possible solution to determining whether unconditional stimulus will follow the presence of the conditional stimuli could be the use of the Q-learning algorithm (with a integrated error correction model in the return predictor function) and a probabilistic K-means (unsupervised) algorithm.

In addition to the Q-learning algorithm's 'off-policy' nature and return predictor, which predicts the accumulative rewards for future states and actions, a further recursive error correction function could be implemented to assess the errors resulting from the combination of states and actions accessible from the current state and action; the aim of this function would be to minimize the prediction error. The various metrics produced by this modified Q-learning algorithm could be used as input to the K-means algorithm to cluster predictions, outlining similarities and solutions that are considered the most optimal.

In relation to Pavlov's classical conditioning experiment, through the use of the modified Q-learning algorithm mentioned above, each conditional stimuli could be represented as a state and each action could either lead to predicting the unconditional stimulus, another conditioned stimuli or nothing at all. As a result of the modified Q-learning algorithm's architecture, the predicted reward (i.e. unconditional stimulus) and the errors (from the incorporated recursive error correction function) for the combination of states and actions accessible from the current state and action can be obtained. These results would then be mapped into the K-means clustering algorithm to determine if the agent can make certain decisions that lead to an optimal solution (i.e. predict whether it the conditional stimuli present are going to lead to the unconditional stimulus).

References

[1] Matthews, W.H. (1922) *Mazes and Labyrinths: A General Account of Their History and Developments*. London, England: Longmans, Green and Co.

[2] *Artificial Intelligence/AI Agents and their Environments* (2019) Available at: https://en.wikibooks.org/wiki/Artificial_Intelligence/AI_Agents_and_their_Environments (Accessed: March 2019).

[3] Even-Dar, E. and Mansour, Y. (2003) 'Learning Rates for Q-learning', *Journal of Machine Learning Research*, 5(1), pp. 1-25.

[4] Sutton, R.S. and Barto, A.G.2017, "**Reinforcement learning** An introduction, n (Second edition, in progress). Cambridge: MIT press: webdocs.cs.ualberta.ca/ sutton/book/the-book.html

[5] *Q-learning* (2019) Available at: https://en.wikipedia.org/wiki/Q-learning (Accessed: April 2019).

[6] Watkins, C.J.C.H. and Dayan, P. (1992) 'Q-learning', *Machine Learning*, 8(3), pp. 279-292.

[7] Pavlov, I.P. (1927) *Conditional Reflexes: An Investigation of the Physiological Activity of the Cerebral Cortex*. Oxford, England: Oxford University Press.

[8] Daw, N.D. and Tobler, P.N. (2013) 'Chapter 15 - Value Learning through Reinforcement: The Basics of Dopamine and Reinforcement Learning' in Glimcher, P.W. and Fehr, E. (ed.) *Neuroeconomics (Second Edition)*. London, England: Academic Press, pp. 283-298.

[9] Rescorla, R.A. and Wagner, A.R. (1972) 'A Theory of Pavlovian Conditioning: Variations in the Effectiveness of Reinforcement and Nonreinforcement' in Black, A.H. and Prokasy, W.F. (ed.) *Classical Conditioning II: Current Reseach and Theory*. New York, USA: Appleton Century Crofts, pp. 64-99.

[10] Kamin, L.J. (1969) 'Predictability, Surprise, Attention and Conditioning' in Campbell, B.A. and Church, R.M. (ed.) *Punishment and Adverse Behaviour*. New York, USA: Appleton Century Crofts, pp. 279-296.

[11] Niv, Y. (2009) 'Reinforcement Learning in the Brain', *Journal of Mathematical Psychology*, 53(3), pp. 139-154.

[12] Barto, A. G., Sutton, R. S. and Watkins, C. J. C. H. (1990) 'Learning and Sequential Decision Making' in Gabriel, M. and Moore, J. (ed.) *Learning and Computational Neuroscience: Foundations of Adaptive Networks*. Cambridge, MA, USA: MIT Press, pp. 539-602.

[13] Sutton, R.S. (1992) 'Reinforcement Learning Architectures', *Proceedings of the International Symposium on Neural Information Processing*.

[14] *Understanding the role of the discount factor in reinforcement learning* (2019) Available at: https://stats.stackexchange.com/questions/221402/understanding-the-role-of-the-discount-factor-in-reinforcement-learning (Accessed: April 2019).