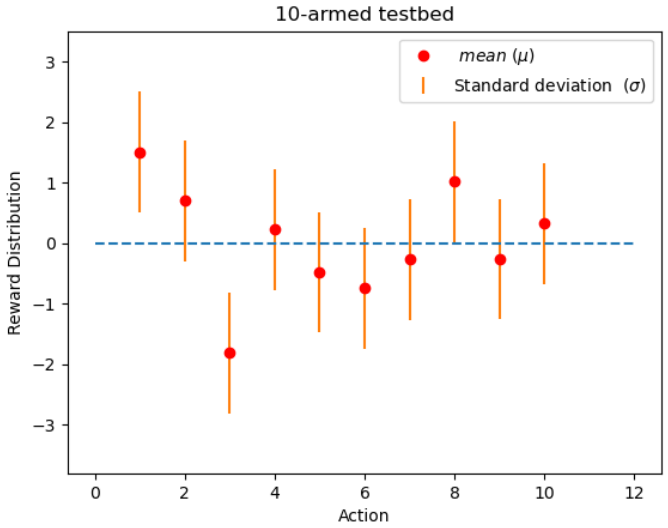
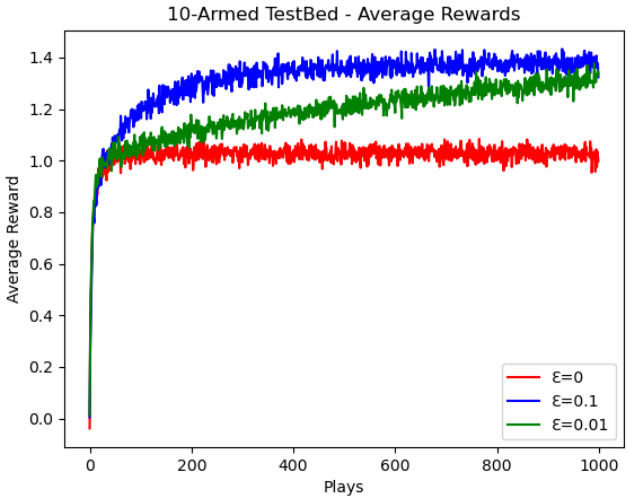
Assignment 1 – Reinforcement Learning

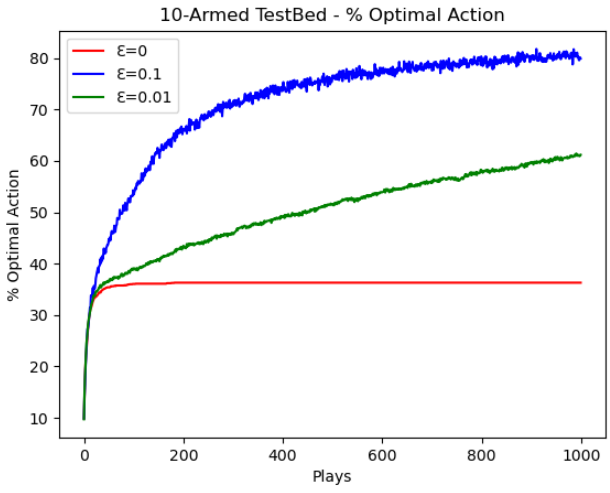
Group 7		
Srirag Jayakumar	201518670	s.jayakumar1@liverpool.ac.uk
Jishnu Prakash Kunnanath Poduvattil	201581347	j.kunnanath-poduvattil@liverpool.ac.uk
Akhil Raj	201594703	a.raj@liverpool.ac.uk

1) Problem 1

Comparison of Ɛ-Greedy Methods for values 0.01 and 0.1 is implemented in python and the source file is attached. The results are discussed below.







We recreated the same conditions required for the experiment and re-implemented the figure 2.2 in the Sutton & Barto book. Comments and documentation regarding running the code and recreating our results is explained in the source file “10armtestbed.py” and “Readme.md”.

COMP532 – Assignment 1 Reinforcement Learning

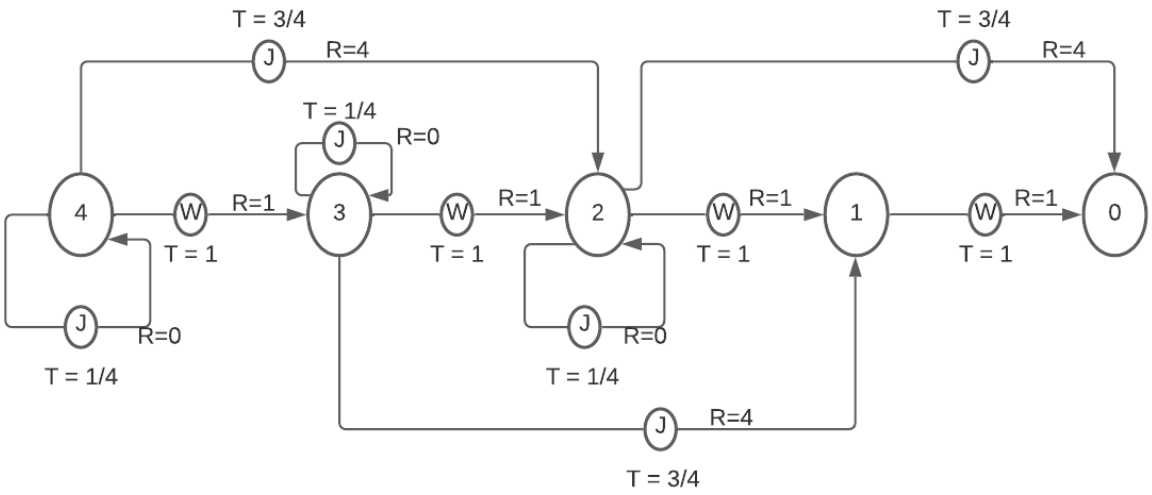
The Exploration and Exploitation Dilemma: -

The notion to take an optimal decision based on available knowledge assuming it is sufficient and the notion to go with a suboptimal decision thinking the available knowledge is insufficient, had been studied and discussed by the scientific community for a long time now and remains as first interest. This dilemma to choose between exploitation and exploration consists of a phenomenal trade-off. When a greedy strategy chooses the best action from current knowledge, the disadvantage is that the performance depends on the accuracy of the knowledge. if our knowledge is not accurate enough, the agent will be stuck choosing the suboptimal actions. In an Ɛ-Greedy strategy, the agent explores for the given probability value of Ɛ. This leads to better learning and performance in some cases.

Given the 10-armed test bed case, sample average technique is used to compute action value estimates in all 3 methods. In terms of exploitation, the greedy method (Ɛ=0) showed a fast learning at the beginning but levelled off at a later stage whereas speaking in terms of exploration, the Ɛ-greedy method shows a better performance. Two plays of 2000 independent runs were conducted using two different values of epsilon. For the case (Ɛ=0.1), they improved their chance of finding the optimal action by exploring. The learning was slow at first for the case (Ɛ=0.01), but eventually it performed well than the other two methods.

We can choose between exploration and exploitation with respect to the solution we want to achieve for a particular task. If the reward variances are 0, the greedy method can perform well and the noisier the rewards can get, including exploration using the Ɛ-greedy methods can do better performance.

2) Problem 2



The above diagram is drawn according to the specifications given. Rewards are calculated as per the given equation and labelled in the above diagram. As the policy is not specified, assuming that action probability is 1.

Given that, $R(s,a,s') = (s-s')^2$ for all (s,a,s')

COMP532 – Assignment 1 Reinforcement Learning

Reward	Value
R (4, W, 3)	1
R (3, W, 2)	1
R (2, W, 1)	1
R (1, W, 0)	1
R (4, J, 2)	4
R (3, J, 1)	4
R (2, J, 0)	4
R (4, J, 4)	0
R (3, J, 3)	0
R (2, J, 2)	0

State Value: -

We know that,

$$V^{\pi}(s) = \sum_{a} \pi(s,a) \sum_{s'} P(s'|s,a)\ [R(s,a,s') + \gamma V^{\pi}(s')]$$

$$V^{*}(s) = max_{\pi}[V^{\pi}(s)], \quad for\ every\ s\ \epsilon\ s$$

$$V^{*}(0) = 0\ (Terminal\ state)$$

$$V^{*}(1) = [1 * \left[1 + \frac{1}{2} * V^{*}(0)\right] = 1$$

$$V^{*}(2) = max_{\pi}\{V_{W}^{\pi}(2), V_{J}^{\pi}(2)\}$$

$$V_{W}^{\pi}(2) = [1 * \left[1 + \frac{1}{2} * V^{*}(1)\right] = \frac{3}{2}$$

$$V_{J}^{\pi}(2) = \{\frac{3}{4} * \left[4 + \frac{1}{2} * V^{\pi}(0)\right] + \frac{1}{4} * \left[0 + \frac{1}{2} * V^{\pi}(2)\right]$$

$$V_{J}^{\pi}(2) = \{3 + \tfrac{1}{8} *\ V^{\pi}(2)\}$$

Solving the above,

$$V^{\pi}(2) = \frac{24}{7}$$

$$V^{*}(2) = max_{\pi}\left[\frac{3}{2}, \frac{24}{7}\right]$$

$$V^{*}(2) = \frac{24}{7}$$

Action Value: -

We know that,

$$Q^{\pi}(s,a) = \sum_{s'} P(s'|s,a)\left[R(s,a,s') + \gamma \sum_{a'} \pi(a'|s') * Q^{\pi}(s',a')\right]$$

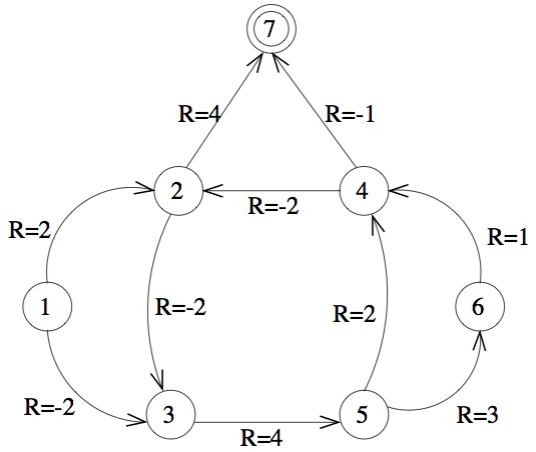
$$Q^{*}(3,J) = \sum_{s'} P(3|3,J)\left[R(3,J,3) + \frac{1}{2} * \{\pi(J,3) * Q^{\pi}(3,J) + \pi(J,1) * Q^{\pi}(1,J) + \pi(W,2) * Q^{\pi}(2,W)\}\right] + \sum_{s'} P(1|3,J)\left[R(3,J,1) + \frac{1}{2} * \{\pi(W,1) * Q^{\pi}(1,W)\}\right]$$

$$Q^{*}(3,J) = \frac{1}{4} * \left[0 + \frac{1}{2} * \left\{1 * \frac{1}{4} + 1 * \frac{3}{4} + 1 * 1\right\}\right] + \frac{3}{4} * \left[4 + \frac{1}{2} * \{1 * 1\}\right]$$

$$Q^{*}(3,J) = \frac{1}{4} + \frac{3}{4} * \frac{9}{2} = \frac{29}{8}$$

3) Problem 3

The objective is to find state values for the given reinforcement learning problem.



We know that for a given policy the state-value function estimate using temporal difference is,

$$V(S_t) = V(S_t) + \alpha[r_{t+1} + \gamma * V(S_{t+1}) - V(S_t)]$$

Given that initial value of all the states is 0, $\alpha = 0.5, \gamma = 1$.

V(1)	V(2)	V(3)	V(4)	V(5)	V(6)	V(7)
0	0	0	0	0	0	0

Episode 1: -

{1, 3, 5, 4, 2, 7}

Value state	Calculation	Value
$V(1) = V(1) + 0.5(R_{t+1}+1*V(3)-V(1))$	0+0.5(-2+0-0)	-1
$V(3) = V(3) + 0.5(R_{t+1}+1*V(5)-V(3))$	0+0.5(4+0-0)	2
$V(5) = V(5) + 0.5(R_{t+1}+1*V(4)-V(5))$	0+0.5(2+0-0)	1
$V(4) = V(4) + 0.5(R_{t+1}+1*V(2)-V(4))$	0+0.5(-2+0-0)	-1
$V(2) = V(2) + 0.5(R_{t+1}+1*V(7)-V(2))$	0+0.5(4+0-0)	2

So after episode 1, the value states are

V(1)	V(2)	V(3)	V(4)	V(5)	V(6)	V(7)
-1	2	2	-1	1	0	0

Episode 2: -

{2, 3, 5, 6, 4, 7}

Value state	Calculation	Value
$V(2) = V(2) + 0.5(R_{t+1}+1*V(3)-V(2))$	2+0.5(-2+2-2)	1
$V(3) = V(3) + 0.5(R_{t+1}+1*V(5)-V(3))$	2+0.5(4+1-2)	3.5
$V(5) = V(5) + 0.5(R_{t+1}+1*V(6)-V(5))$	1+0.5(3+0-1)	2
$V(6) = V(6) + 0.5(R_{t+1}+1*V(4)-V(6))$	0+0.5(1-1-0)	0
$V(4) = V(4) + 0.5(R_{t+1}+1*V(7)-V(4))$	-1+0.5(-1+0-1)	-2

So after episode 2, the value states are

V(1)	V(2)	V(3)	V(4)	V(5)	V(6)	V(7)
-1	1	3.5	-2	2	0	0

Episode 3: -

{5, 4, 2, 7}

Value state	Calculation	Value
$V(5) = V(5) + 0.5(R_{t+1}+1*V(4)-V(5))$	2+0.5(2-2-2)	1
$V(4) = V(4) + 0.5(R_{t+1}+1*V(2)-V(4))$	-2+0.5(-2+1+2)	-1.5
$V(2) = V(2) + 0.5(R_{t+1}+1*V(7)-V(2))$	1+0.5(4+0-1)	2.5

So after episode 3, the value states are

V(1)	V(2)	V(3)	V(4)	V(5)	V(6)	V(7)
-1	2.5	3.5	-1.5	1	0	0

4) Problem 4

Akhil

5) Problem 5

We recreated the same conditions for the comparison of SARSA and Q-Learning in the cliff walking task and re-implemented the results in figure 6.4 in the Sutton & Barto book. Comments and documentation regarding running the code and recreating our results is explained in the source file "tempDifference.py" and "Readme.md".