Homework 2 Template

Use this template to record your answers for Homework 2. Add your answers using LaTeXand then save your document as a PDF to upload to Gradescope. You are required to use this template to submit your answers. **You should not alter this template in any way** other than to insert your solutions. You must submit all 15 pages of this template to Gradescope. Do not remove the instructions page(s). Altering this template or including your solutions outside of the provided boxes can result in your assignment being graded incorrectly. You may lose points if you do not follow these instructions.

You should also export your code as a .py file and upload it to the **separate** Gradescope coding assignment. Remember to mark all teammates on **both** assignment uploads through Gradescope.

Instructions for Specific Problem Types

On this homework, you must fill in (a) blank(s) for each problem; please make sure your final answer is fully included in the given space. **Do not change the size of the box provided.** For short answer questions you should **not** include your work in your solution. Only provide an explanation or proof if specifically asked. Otherwise, your assignment may not be graded correctly, and points may be deducted from your assignment.

Fill in the blank: What is the course number?

10-703

Problem 0: Collaborators

Enter your team's names and Andrew IDs in the boxes below. If you do not do this, you may lose points on your assignment.

Name 1:	Eu Jing Chua	Andrew ID 1:	eujingc
Name 2:		Andrew ID 2:	
Name 3:		Andrew ID 3:	

Problem 1: Basics & MDPs (27 pts)

1.1 Value Function Proof (6 pts)

Let r_{max} be the maximum possible reward in our environment.

$$
\begin{aligned}
\sum_{k=0}^{\infty} \gamma^k r_{t+1+k} &\leq \sum_{k=0}^{\infty} \gamma^k r_{max} \\
&= r_{max} \sum_{k=0}^{\infty} \gamma^k \\
&= \frac{r_{max}}{1-\gamma} \\
\mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+1+k}\right] &= \sum_{k=0}^{\infty} \gamma^k \mathbb{E}\left[r_{t+1+k}\right] \\
&\leq \sum_{k=0}^{\infty} \gamma^k r_{max} \\
&= \frac{r_{max}}{1-\gamma}
\end{aligned}
$$

Hence the value function is bounded for all states.

1.2 True or False with Explanations (6 pts)

(a) True. For any finite MDP there exists policies and associated state-action functions, so there exists a greedy optimal state-action function over all policies. Thus, there exists an optimal policy induced from this optimal state-action function.

(b) True. If there were optimal policies with different value functions, then there exists some state which the value functions differ at. One must have a value lower than the other at this state, but this is a contradiction as one is not optimal then.

(c) False. Assuming this questions is dealing with stochastic policies vs the optimal policy only, the optimal policy takes the action that maximizes the expected reward for each state, which is deterministic. In a stochastic policy, we take actions that either have the best expected reward or lower, and so cannot get higher rewards than the deterministic optimal policy.

1.3.(a) Describe for each of three MPDs (6 pts)

1. Not possible. Any policy that always does a_2 at s_1 will have the maximum value for s_1.
2. Possible. Any policy such that $\pi(a_2|s_1) > 0$ will result in infinite value for s_1.
3. Possible. Any policy such that $\pi(a_1|s_1) = 1, \pi(a_1|s_2) > 0$, or $\pi(a_2|s_1) > 0, \pi(a_2|s_2) = 1$ will result in infinite value for s_1.

1.3.(b) Describe for each of three MPDs (3 pts)

1. Any policy such that $\pi(a_2|s_1) = 1$.
2. Any policy such that $\pi(a_2|s_1) = 1$.
3. Any policy such that $\pi(a_1|s_1) = 1$ and $\pi(a_2|s_2) = 1$.

1.4 Describe the MDP and two Policies (6 pts)

Let there be two states $\{s_1, s_2\}$ and two actions $\{a_1, a_2\}$.
The transition probabilities are:

1. $P(s_1|s_1, a_1) = P(s_2|s_2, a_1) = 1$ (a_1 is "Stay in state")
2. $P(s_2|s_1, a_2) = P(s_1|s_2, a_2) = 1$ (a_2 is "Change state")

The rewards are $R(s_1, a_1, s_1) = 1, R(s_2, a_1, s_2) = 100$.
Then if $0 < \gamma < 1$, the set of optimal policies are such that $\pi(a_2|s_1) = 1, \pi(a_1|s_2) = 1$.
If $\gamma = 0$, then the set of optimal policies are such that $\pi(a_1|s_1) = 0, \pi(a_1|s_2) = 1$.

Problem 2: Value Iteration & Policy Iteration (26 pts)

2.1 Table: Policy Iteration (4 pts)

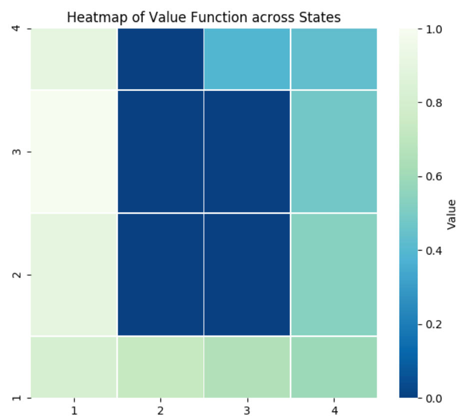
Environment	# Policy Improvement Steps	Total # Policy Evaluation Steps
Deterministic-4x4	8	51
Deterministic-8x8	14	180

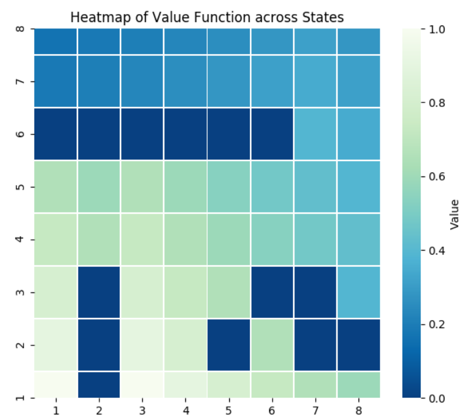
2.2 Optimal Policies for Deterministic-4x4 and 8x8 Maps (2 pts)

```
DLRD
RLLD
ULLD
ULLL
```

```
DDDDDDDL
RRRRRRDL
LLLLLLDL
DLDLLLLL
DLDLLLLL
DLDLLLLU
DLDLLDLL
RLLLLLLL
```

2.3 Value Functions of the Optimal Policies (2 pts)

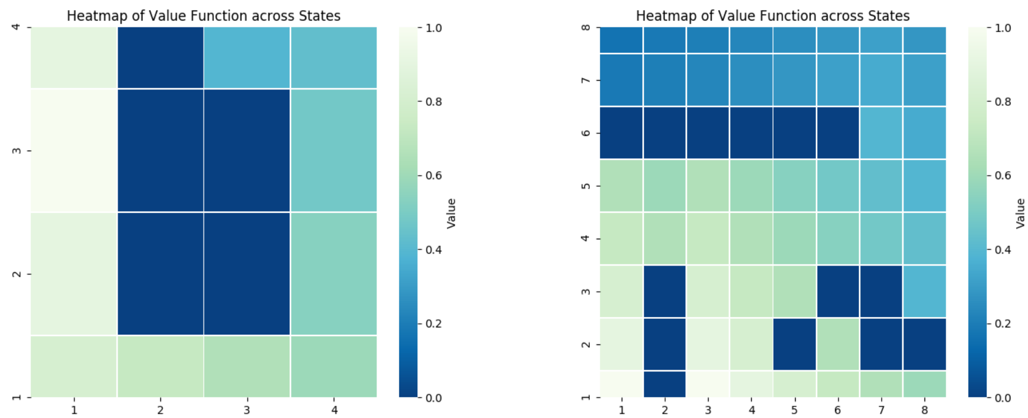




2.4 Table: Synchronous Value Iteration (4 pts)

Environment	# Iterations
Deterministic-4x4	11
Deterministic-8x8	19

2.5 Value Functions from Synchronous Value Iteration (2 pts)



2.6 Optimal Policies from Synchronous Value Iteration (2 pts)

```
DLRD
RLLD
ULLD
ULLL
```

```
DDDDDDDL
RRRRRRDL
LLLLLLDL
DLDLLLLL
DLDLLLLL
DLDLLLLU
DLDLLDLL
RLLLLLLL
```

2.7 Answer PI v.s. VI (2 pts)

Value iteration is faster, and this is expected as it does not have to alternate between updating a policy and value function.
In general, if both algorithms were given enough maximum iterations and the same tolerance in delta, there should be no differences in value function as they will both converge to the optimal.

2.8 Table: Asynchronous Policy Iteration(4 pts)

Heuristic	Policy Improvement Steps	Total Policy Evaluation Steps
Ordered	14	119
Randperm	14	117.3

2.9 Table: Asynchronous Value Iteration(4 pts)

Heuristic	# Iterations
Ordered	15
Randperm	10.2

2.10 (Bonus) Asynchronous VI with Domain-specific Heuristic

Env	# Iterations
Deterministic-4x4	6
Deterministic-8x8	7

The "goal distance" heuristic is expected to perform the best in terms of converging the fastest if the actual distance from each state to the goal is not far off from the Manhattan Distance, i.e. the paths are straightforward without much obstacles inbetween. In this case, we come close to approximating the dynamic programming approach by building up values from the goal state backwards.

Problem 3: DQN (47+20 pts)

3.1.1 Bellman Equation for $C(s, a)$ (3 pts)

$$C(s_1, a_1) = \gamma \sum_{s_2} T(s_1, a_1, s_2) V(s_2) = \gamma \sum_{s_2} T(s_1, a_1, s_2) \max_{a_2} \left(R(s_2, a_2) + C(s_2, a_2)\right)$$

3.1.2 Relations among Q & V & C (4 pts)

label=() $V(s) = \max_a (R(s,a) + C(s,a))$
lbbel=() $Q(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s') V(s')$
lcbel=() $Q(s,a) = R(s,a) + C(s,a)$
ldbel=() $C(s,a) = Q(s,a) - R(s,a)$

3.1.3 $\sim$ 5 True or False with Explanations (6 pts)

3. True. The optimal policy is $\pi(s) =_a Q(s,a)$.
4. False. We need to know the system dynamics too if we are given the value function, which having only $R(s,a)$ is insufficient.
5. True. $Q(s,a) = R(s,a) + C(s,a)$ so this is similar to 3).

3.2: Understanding TD & MC (4 pts)

1. False. TD methods do not need the full trajectories and learn in an online manner as they learn from every state transition from interacting with the environment.
2. False. We need trajectories to terminate so we can accumulate multiple trajectories to get empirical distributions.

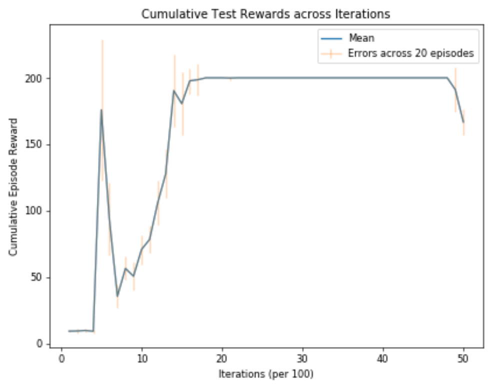
3.3.1.(a) Description & Hyperparameters (20 pts)

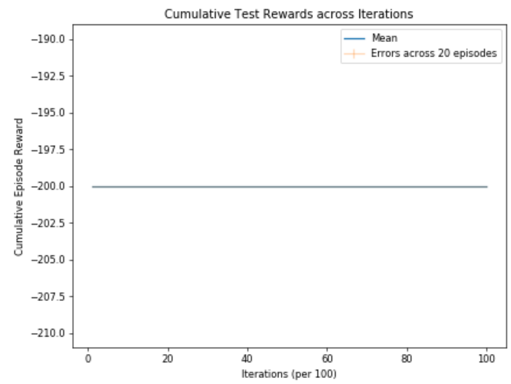
Both implementations follow the algorithm of the Atari paper, but with different network architectures and hyperparameters:

1. Cart Pole:
   (a) Network: Input $\Rightarrow$ Dense 25 units w/ relu $\Rightarrow$ Dropout p=0.5 $\Rightarrow$ Dense 25 units w/ relu $\Rightarrow$ Output
   (b) Gamma: 0.99
   (c) Optimizer: Adam with learning rate of 0.0002
   (d) Epsilon Decay: Linear from 0.8 decreasing by 4.5×10^{-5} each episode
   (e) Episodes: 5000
   (f) Memory size: 5×10^6, Burn-in: 1×10^4 random actions sampled from action-space
   (g) Batch size: 32
2. Mountain Car:
   (a) Network: Input $\Rightarrow$ Dense 80 units w/ relu $\Rightarrow$ Dropout p=0.5 $\Rightarrow$ Dense 80 units w/ relu $\Rightarrow$ Output
   (b) Gamma: 1
   (c) Optimizer: Adam with learning rate of 0.001
   (d) Epsilon Decay: Linear from 0.5 decreasing by 4.5×10^{-6} each episode
   (e) Episodes: 5000
   (f) Memory size: 2×10^4, Burn-in: 1×10^4 actions sampled from epsilon-greedy freshly initialized network $\epsilon = 0.5$
   (g) Batch size: 640

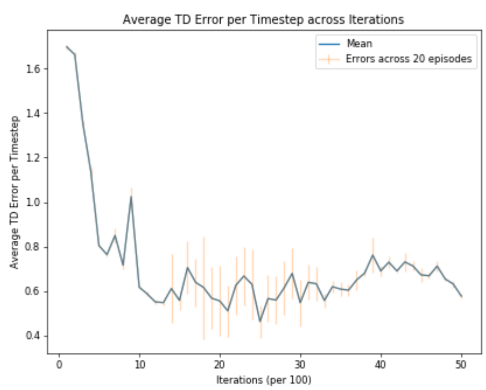
I think cart pole was successful, but mountain car was a very difficult problem to solve from an exploration point of view. I spent many days tuning hyper-parameters for it and never got a network to learn to reach the goal within 100k episodes. Although I occasionally get < 10 episodes reaching the goal in all the training, there was simply not enough signal for my network to learn to reach it. I have tried increasing the replay buffer to a very large amount, and a smaller amount, increasing batch size, training from multiple mini-batches each timestep, using large (100 hidden units) networks, small and large learning rates, varying ϵ to large amounts and nothing would make the network learn besides augmenting the reward, which was not allowed in this case. Looking at the performance on cart pole, I think my algorithm should be quite right and maybe I just never got lucky with hyper-parameter tuning for mountain car, but I think I have spent too much time on it and hope I will not get penalized too much for a failure on solving mountain car.

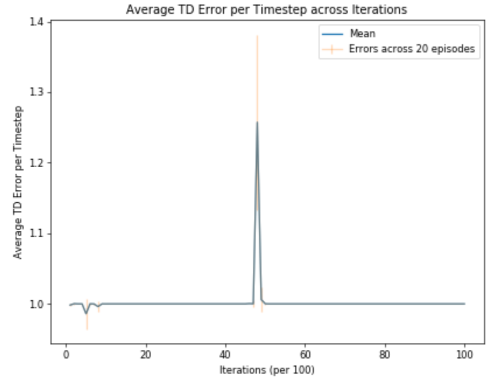
3.3.1.(b) Two Plots: Average Cumulative Reward (5 pts)





3.3.1.(c) Two Plots: TD Error (5 pts)





For cart pole on the left, we see that the TD error does roughly decrease as reward is increased. This makes sense as the Q values predicted by the network slowly converge on the optimal ones, where we know that if the Q values were optimal there would be 0 TD error. However, Q values can go over the maximum possible (200 in this case) due to over-fitting issues, so a dropout layer was introduced to regularize network weights. Hence, we observe average TD errors that are more than 1 while the network is still adjusting in the learning stage, far from the optimal values.

Extra Credit

3.3.2.(a) ALGORITHM: Description & Hyperparameters (14 pts)

3.3.2.(b) Two Plots: Average Cumulative Reward (3 pts)

3.3.2.(c) Two Plots: TD Error (3 pts)

Extra (2pt)

Feedback (1pt): You can help the course staff improve the course for future semesters by providing feedback. You will receive a point of you provide actionable feedback. What was the most confusing part of this homework, and what would have made it less confusing?

Hyper-parameter tuning took up significantly more time than actually implementing the core algorithm, which I think actually disrupted work in my other classes. I think it would be good if there was less focus on finding good hyperparameters and network designs (standardized things that could be provided) and more focus on actually implementing the core algorithm.

Time Spent (1pt): How many hours did you spend working on this assignment? Your answer will not affect your grade.

Alone	$>$ 30 hours
With teammates	0
With other classmates	0
At office hours	2