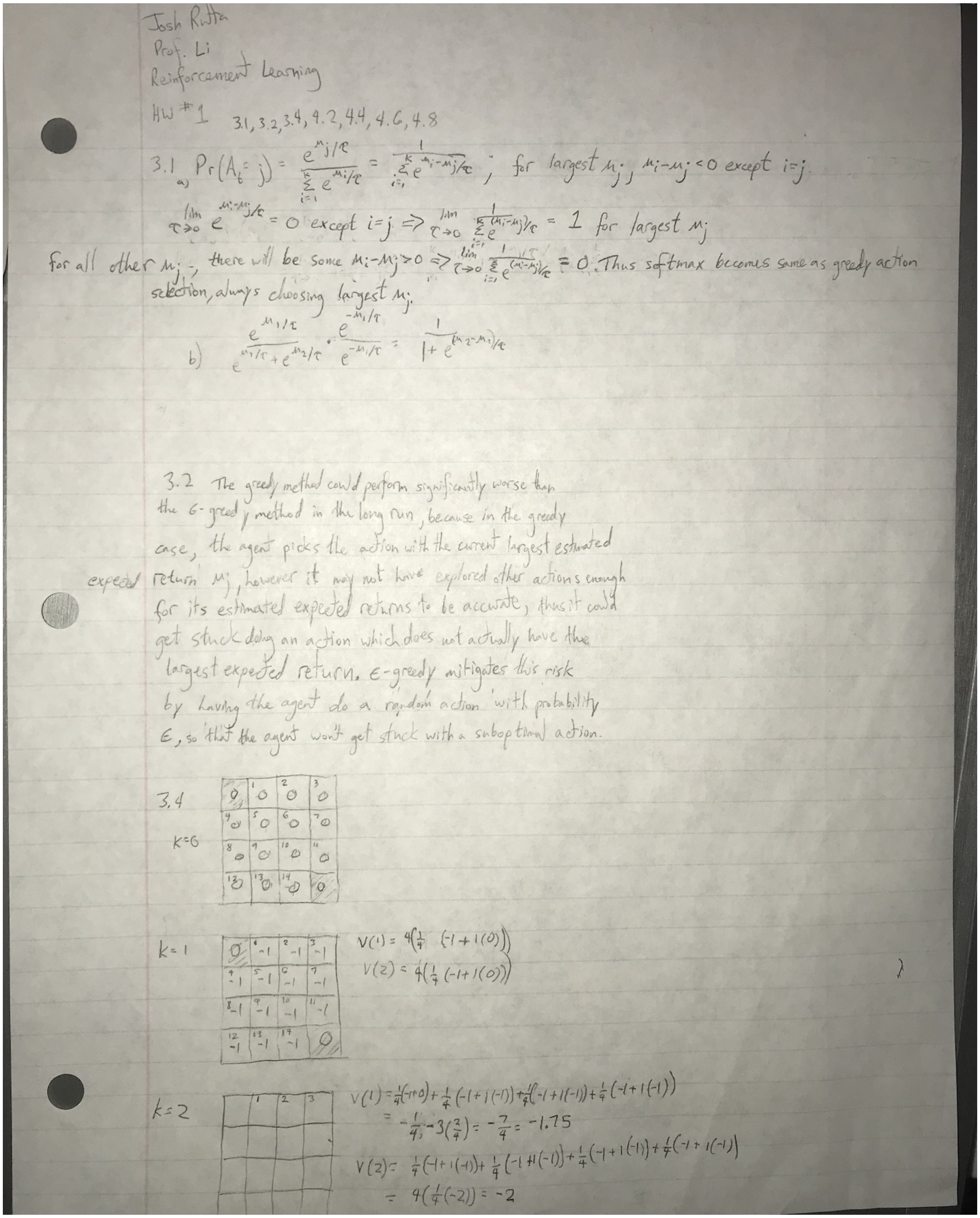
ELEN E6885: Reinforcement Learning
Homework 1

Josh Rutta

September 28, 2018



Josh Rutta
Prof. Li
Reinforcement Learning

HW #1 3.1, 3.2, 3.4, 4.2, 4.4, 4.6, 4.8

3.1 a) $\Pr(A_t = j) = \frac{e^{\mu_j/\tau}}{\sum_{i=1}^{k} e^{\mu_i/\tau}} = \frac{1}{\sum_{i=1}^{k} e^{\mu_i - \mu_j/\tau}}$; for largest μ_j, $\mu_i - \mu_j < 0$ except $i=j$.

$\lim_{\tau \to 0} e^{\mu_i - \mu_j/\tau} = 0$ except $i=j \Rightarrow \lim_{\tau \to 0} \frac{1}{\sum_{i=1}^{k} e^{(\mu_i - \mu_j)/\tau}} = 1$ for largest μ_j

for all other μ_j, there will be some $\mu_i - \mu_j > 0 \Rightarrow \lim_{\tau \to 0} \frac{1}{\sum_{i=1}^{k} e^{(\mu_i - \mu_j)/\tau}} = 0$. Thus softmax becomes same as greedy action selection, always choosing largest μ_j.

b) $\frac{e^{\mu_1/\tau}}{e^{\mu_1/\tau} + e^{\mu_2/\tau}} \cdot \frac{e^{-\mu_1/\tau}}{e^{-\mu_1/\tau}} = \frac{1}{1 + e^{(\mu_2 - \mu_1)/\tau}}$

3.2 The greedy method could perform significantly worse than the ϵ-greedy method in the long run, because in the greedy case, the agent picks the action with the current largest estimated expected return μ_j, however it may not have explored other actions enough for its estimated expected returns to be accurate, thus it could get stuck doing an action which does not actually have the largest expected return. ϵ-greedy mitigates this risk by having the agent do a random action with probability ϵ, so that the agent won't get stuck with a suboptimal action.

3.4

k=0

0	1 0	2 0	3 0
4 0	5 0	6 0	7 0
8 0	9 0	10 0	11 0
12 0	13 0	14 0	0

k=1

0	1 -1	2 -1	3 -1
4 -1	5 -1	6 -1	7 -1
8 -1	9 -1	10 -1	11 -1
12 -1	13 -1	14 -1	0

$V(1) = 4\left(\frac{1}{4}\left(-1 + 1(0)\right)\right)$

$V(2) = 4\left(\frac{1}{4}\left(-1 + 1(0)\right)\right)$

k=2

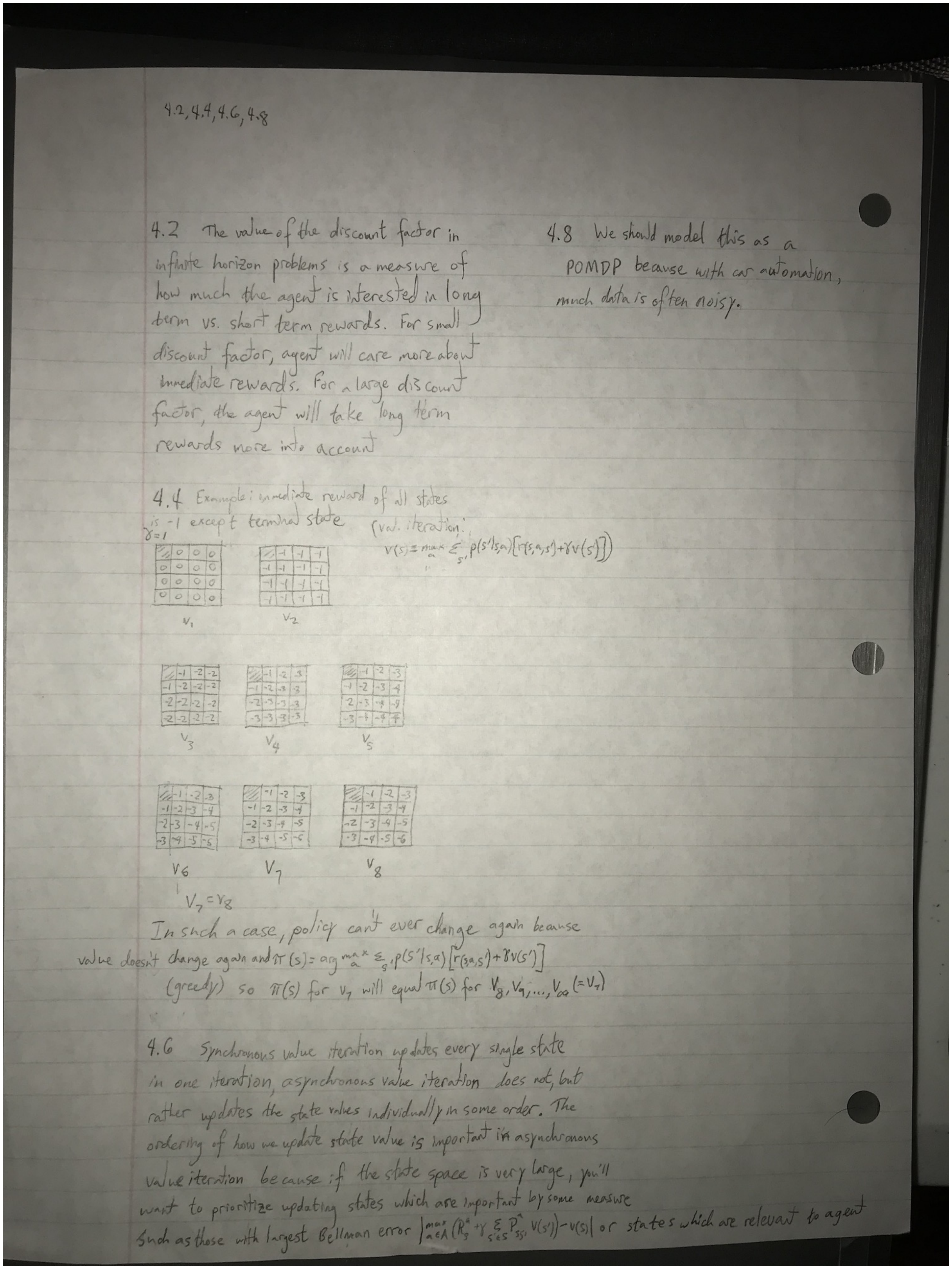
	1	2	3

$V(1) = \frac{1}{4}(-1+0) + \frac{1}{4}\left(-1 + 1(-1)\right) + \frac{1}{4}\left(-1 + 1(-1)\right) + \frac{1}{4}\left(-1 + 1(-1)\right)$

$= -\frac{1}{4} - 3\left(\frac{2}{4}\right) = -\frac{7}{4} = -1.75$

$V(2) = \frac{1}{4}\left(-1 + 1(-1)\right) + \frac{1}{4}\left(-1 + 1(-1)\right) + \frac{1}{4}\left(-1 + 1(-1)\right) + \frac{1}{4}\left(-1 + 1(-1)\right)$

$= 4\left(\frac{1}{4}(-2)\right) = -2$



4.2, 4.4, 4.6, 4.8

4.2 The value of the discount factor in infinite horizon problems is a measure of how much the agent is interested in long term vs. short term rewards. For small discount factor, agent will care more about immediate rewards. For a large discount factor, the agent will take long term rewards more into account

4.8 We should model this as a POMDP because with car automation, much data is often noisy.

4.4 Example: immediate reward of all states is -1 except terminal state

$\gamma=1$

(val. iteration: $v(s) = \max_a \sum_{s'} p(s'|s,a)[r(s,a,s')+\gamma v(s')]$)

▨	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

v_1

▨	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1
-1	-1	-1	-1

v_2

▨	-1	-2	-2
-1	-2	-2	-2
-2	-2	-2	-2
-2	-2	-2	-2

v_3

▨	-1	-2	-3
-1	-2	-3	-3
-2	-3	-3	-3
-3	-3	-3	-3

v_4

▨	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-4
-3	-4	-4	-4

v_5

▨	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-5

v_6

▨	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

v_7

▨	-1	-2	-3
-1	-2	-3	-4
-2	-3	-4	-5
-3	-4	-5	-6

v_8

$v_7 = v_8$

In such a case, policy can't ever change again because value doesn't change again and $\pi(s) = \arg\max_a \sum_{s'} p(s'|s,a)[r(s,a,s')+\gamma v(s')]$ (greedy) so $\pi(s)$ for v_7 will equal $\pi(s)$ for $v_8, v_9, \ldots, v_\infty (=v_7)$

4.6 Synchronous value iteration updates every single state in one iteration, asynchronous value iteration does not, but rather updates the state values individually in some order. The ordering of how we update state value is important in asynchronous value iteration because if the state space is very large, you'll want to prioritize updating states which are important by some measure such as those with largest Bellman error $\left|\max_{a\in A}\left(R_s^a + \gamma \sum_{s'\in S} P_{ss'}^a V(s')\right) - V(s)\right|$ or states which are relevant to agent