REINFORCEMENT LEARNING, Q-LEARNING, AND EXTENSIONS TO MULTI-AGENT SYSTEMS

MATTHEW SCHARF

Abstract. Reinforcement Learning (RL) is an interdisciplinary field which combines ideas from Dynamic Programming and Machine Learning to statistically optimize a control policy in a complex environment. As RL becomes increasingly prevalent in the automation of tasks, it is imperative to understand the theory of RL and to consider the result when multiple such systems interact. Furthermore, while the focus of the field is often in its applications, it also has rich mathematical underpinnings. The goals of this paper are to (1) provide a rigorous introduction to the field, (2) develop the theory to describe a core technique in RL called 'Q-Learning', and (3) draw upon game theory to explore extensions to multi-agent systems. Additionally, I will provide several illustrative implementations across the single agent and multi-agent settings.

Date: May 11, 2020.
This document is a senior thesis submitted to the Mathematics and Statistics Department of Haverford College in partial fulfillment of the requirements for a major in Mathematics.

Contents

Introduction

The field of Reinforcement Learning[1] (RL) deals with finding an optimal control policy (i.e. a decision-making strategy) in scenarios where the environment is too large, complex, or uncertain for such a policy to be identified using analytical techniques. RL drew worldwide attention in 2018 for solving the complex game of Go without using any human knowledge [8]. Since then, RL has found success in a wide variety of domains such as healthcare [11], robotics, and resource allocation.

To accomplish the goals of this paper, we will proceed as follows: In the first two sections, I will introduce Reinforcement Learning. Sections 3 & 4 build the tools and theory to describe Q-Learning. Sections 7 & 8 discuss Multi-Agent RL. In sections 5, 6 & 9, I provide three examples[2,3] across single agent and multi agent settings.

1. Introducing the RL Problem

Imagine you are given a maze-solving robot. When provided with a new maze, he will be allowed to explore the maze for short episodes (e.g. 5 minutes). When navigating the maze, the robot will repeatedly:

(1) observe his current state (e.g. 3 meters northeast of the start)
(2) decide on an action to take (e.g. north, south, east, or west)
(3) receive feedback on progress (e.g. success, failure, or not done)

At the end of each episode, the robot is transported back to the beginning of the maze to try again. Now, suppose you are tasked with providing an algorithm for the robot, such that, *given any new maze, he will identify a behavior which solves it.*

Intuitively, the robot must first explore the maze through trial and error for many episodes. He must then exploit the provided feedback in order to identify optimal sequences of actions. Such a problem underpins the motivation and conceptual framework for Reinforcement Learning. I will now provide the mathematical formulation of these ideas. [4]

1.1. **Task Formulation.** The RL task can be formulated as follows:

(1) We receive some environment, $\mathcal{M}$.
(2) We repeat the following steps:

[1]See Barto and Sutton's book [9] for a broad overview of the field.
[2]Code is on github: www.github.com/scharfm16/Reinforcement_Learning_Thesis
[3]For all three implementations, a very basic environment was provided by [4] but all the code for Q-Learning, simulation, and visualizations is my own.
[4]Section 1 contains my own formulations of the problem, using and building on the core RL concepts presented in [9] and [4].

(a) We choose some decision-making strategy or 'policy', π
(b) A model, $\mathcal{M}_\pi$ is constructed.
(c) We repeatedly sample from $\mathcal{M}_\pi$ in order to gain information about $\mathcal{M}$ and π
(d) Repeat until we have found an optimal policy π^*

Definition 1.1. We are *given* an 'environment' in the form of a 6-tuple

$$\mathcal{M} := (\mathcal{S}, \mathcal{A}, \mathcal{R}, f_0, \tau, \gamma)$$

where:

(1) $\mathcal{S}$ is a finite set whose elements are called 'states'. $\mathcal{S}_\infty \subset \mathcal{S}$ is a nonempty subset of states whose elements are called 'terminal states'.
(2) $\mathcal{A}$ is a finite set whose elements are called 'actions'.
(3) $\mathcal{R} \subset \mathbb{R}$ is a finite set whose elements are called 'rewards'.
(4) $f_0(s)$ is a probability mass function (pmf) over $\mathcal{S}$ describing the 'starting state'.
(5) τ is a set of pmfs over $\mathcal{S} \times \mathcal{R}$, one for each state-action pair:

$$\tau := \{\tau_{s,a}(s', r) \mid s \in \mathcal{S}, a \in \mathcal{A}\}$$

describing the 'transition' or reaction of the environment.

τ is restricted for terminal states $s_\infty \in \mathcal{S}_\infty$:
$\forall a \in \mathcal{A}$,

$$\tau_{s_\infty,a}(s, r) = \begin{cases} 1 & \text{if } (s, r) = (s_\infty, 0) \\ 0 & \text{else} \end{cases}$$

In other words, once a terminal state s_∞ is reached, all remaining states will be s_∞ and all remaining rewards will be 0.

(6) $\gamma \in (0, 1)$ is called the 'discount factor' and can be thought of as the proportionate preference for rewards over one time-step.

Definition 1.2. We *choose* a 'policy' π where π is a set of pmfs over $\mathcal{A}$, one for each state:

$$\pi := \{\pi_s(a) \mid s \in \mathcal{S}\}$$

This policy represents our decision-making choices in response to the environment. We will refer to the set of possible policies as

$$\Pi = \{\pi\}$$

1.2. **Model Description.**

Definition 1.3. Consider a *given* environment, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, f_0, \tau, \gamma)$, and a *chosen* policy, π.

First, let $S_0 \sim f_0$. Now, let $A_0 \sim \pi_{S_0}$[5]. Then, let $(S_1, R_1) \sim \tau_{S_0,A_0}$. Continuing this process for N steps, we can recursively define:

(1) $A_i \sim \pi_{S_i}$ for $i \in \{0, 1, \dots, N\}$,
(2) $(S_i, R_i) \sim \tau_{S_{i-1},A_{i-1}}$ for $i \in \{1, 2, \dots\}$.

This process gives us a joint distribution for the following sequence of random variables:

$$\mathcal{M}_\pi := (S_0, A_0, R_1, S_1, A_1, R_2, \dots)$$

This distribution serves as a model of the interactions between the agent and the environment given an chosen policy. An 'episode' of an agent trying the policy on the environment corresponds to sampling from $\mathcal{M}_\pi$. Importantly, $\mathcal{M}_\pi$ itself is unknown to the agent—it can only be sampled. For ease, we will refer to the history of the agent up to time t as $H_t := (S_0, A_0, R_1, \dots, S_{t-1}, A_{t-1}, R_t)$.

A few remarks on $\mathcal{M}_\pi$:

Remark 1.4. By how $\mathcal{M}_\pi$ is constructed, we have that $\forall t \in \mathbb{N}$:

$$\begin{aligned}
&P(S_{t+1} = s', A_{t+1} = a', R_{t+1} = r \mid S_t = s)) \\
&= \sum_{a \in \mathbb{A}} [P(S_{t+1} = s', A_{t+1} = a', R_{t+1} = r, A_t = a \mid S_t = s))] \\
&= \sum_{a \in \mathbb{A}} [P(A_t = a \mid S_t = s) * P(S_{t+1} = s', A_{t+1} = a', R_{t+1} = r \mid S_t = s, A_t = a))] \\
&= \sum_{a \in \mathbb{A}} [\pi_s(a) * \tau_{s,a}(s', r) * \pi_{s'}(a')]
\end{aligned}$$

Remark 1.5. By how $\mathcal{M}_\pi$ is defined, $\forall t \in \mathbb{N}$:

$$(S_{t+1}, R_{t+1} \mid S_t = s) \sim (\tau_{S_t,\pi_{S_t}} \mid S_t = s) = \tau_{s,\pi_s}$$

Because, τ and π are independently specified before $\mathcal{M}_\pi$ is constructed, we have that

$$(S_{t+1}, R_{t+1} \mid S_t = s) \perp\!\!\!\perp H_t$$

Remark 1.6. We make an assumption of 'irreducibility' for $\mathcal{M}_\pi$:

$$\forall s, s' \in \mathcal{S} \quad \exists i \in \mathbb{N}$$

such that

$$P(S_i = s \mid S_0 = s) > 0$$

[5]By abuse of notation, π_{S_0} is a random variable with codomain $\pi = \{\pi_s\}$. In other words, $(\pi_{S_0} \mid S_0 = s) = \pi_s$. This applies to all such instances of this notation.

Intuitively, irreducibility means that from any state $s \in \mathcal{S}$, there is a nonzero probability of reaching any other state $s' \in \mathcal{S}$. In other words, if we think about the MDP as a network of states, that network is fully connected.

Remark 1.7. We will assume that, for some computational limit on time steps $N \in \mathbb{N}$, we have $P(S_N \in \mathcal{S}_\infty) \approx 1$. In other words, when sampling from $\mathcal{M}_\pi$, we will hit a terminal state $s_\infty \in \mathcal{S}_\infty$ within a computationally feasible amount of time.

1.3. **Value Functions and Optimization.** Given some environment, $\mathcal{M}$, the goal is to choose $\pi \in \Pi$ that results in $\mathcal{M}_\pi$ which maximizes some objective. That objective can be formulated using the 'value function'.

Definition 1.8. Consider some environment, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, f_0, \tau, \gamma)$. Then, for a choice of policy, $\pi \in \Pi$, we have $\mathcal{M}_\pi$. We define the 'value function' for π, [4] $v_\pi : S \to \mathbb{R}$, to be:

$$v_\pi(s) := \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{k+1} \mid S_0 = s]$$

The value function gives the expected discounted future returns from state s. I will now define the objective function we are trying to optimize.

Definition 1.9. Given $\mathcal{M}$, we define $\mathcal{O} : \Pi \to \mathbb{R}$:

$$\mathcal{O}(\pi) := \mathbb{E}_{S_0 \sim f_0}[v_\pi(S_0)]$$

Intuitively, $\mathcal{O}$ corresponds to the overall expected returns of the MDP given some choice of π. We can now state the Reinforcement Learning optimization problem.

Definition 1.10. Given some environment, $\mathcal{M}$, we are trying to find some optimal $\pi^* \in \Pi^* \subset \Pi$ where:

$$\Pi^* := argmax_\pi \mathcal{O}(\pi)$$

Such a $\pi^* \in \Pi^*$ must exist but is not necessarily unique. The existence of π^* will be a direct result of section 4. Q-Learning is one algorithm which, given an environment $\mathcal{M}$, finds $\pi^* \in \Pi^*$. Q-Learning will be our optimization algorithm of choice for the rest of this paper.

2. Markov Decision Processes

It is helpful to view the sequence of random variables $\mathcal{M}_\pi$ as a mathematical object called a 'Markov Decision Process' (MDP).[6]

2.1. Definition.

Definition 2.1 (Markov Decision Process)**.** For finite sets, $\mathcal{S}$, $\mathcal{A}$, and $\mathcal{R}$, consider a sequence of random variables:

$$\chi = (S_0, A_0, R_1, S_1, A_1, R_2, \dots)$$

such that

$$S_0, A_0 \text{ have codomain } \mathcal{S} \times \mathcal{A}$$

and

$$\forall i \in \{1, 2, \dots\},$$
$$(S_i, A_i, R_i) \text{ have codomain } \mathcal{S} \times \mathcal{A} \times \mathcal{R}$$

Such a sequence, χ is called a 'Markov Decision Process' (MDP) [5, pp. 1–3] if the following two conditions hold:

(1) Markov Property: Given any $t \in \{1, 2, \dots\}$, we need that[7]
$$((S_{t+1}, R_{t+1}) \perp\!\!\!\perp H_t) \mid S_t$$
(2) Time Invariance: Given any $t \in \{1, 2, \dots\}$ we need that
$$P(S_t = s', A_t = a', R_t = r \mid S_{t-1} = s) =$$
$$P(S_1 = s', A_1 = a', R_1 = r \mid S_0 = s)$$

2.2. Extension of Time Invariance.

The following theorem states that the conditional probabilities of MDPs depend only on the *time that has passed*.

Theorem 2.2. *Consider some MDP, χ. Given any $t, m \in \{1, 2, \dots\}$, it is the case that*

$$P(S_{t+m} = s', A_{t+m} = a', R_{t+m} = r \mid S_t = s) =$$
$$P(S_m = s', A_m = a', R_m = r \mid S_0 = s)$$

For the rest of this paper, when I use the property of 'time invariance' for an MDP, I will be referring to this theorem because it is a powerful extension of the time invariance defined above.

[6] I used [5] as a reference while writing Section 2 but the precise formulations and proofs are my own.

[7] In other words, (S_{t+1}, R_{t+1}) and H_t are conditionally independent given S_t. Specifically, $P(S_{t+1} = s_{t+1}, R_{t+1} = r_{t+1} \mid S_t = s_t, H_t = h_t) = P(S_{t+1} = s_{t+1}, R_{t+1} = r_{t+1} \mid S_t = s_t)$.

Proof. Consider some $\mathcal{M}_\pi$. Then, given any $m, t \in \{1, 2, \dots\}$, we define the set of 'trajectories' to be:

$$\mathcal{T} := \mathcal{S}^{m-1} \times \mathcal{A}^m \times \mathcal{R}^m$$

Each element $T = (a_0, r_0, s_1, a_1, r_1, \dots, s_{m-1}, a_{m-1}, r_{m-1}) \in \mathcal{T}$ represents a possible realization of

$$H_{t:t+m-1} := (A_t, R_t, S_{t+1}, A_{t+1}, R_{t+1}, \dots, S_{t+m-1}, A_{t+m-1}, R_{t+m-1})$$

For ease, let

$$T_i := (a_0, r_0, s_1, a_1, r_1, \dots, s_i, a_i, r_i)$$

Also, recall from probability that for random variables, $A, B, C, \dots$:

$$P(A \cap B \cap \dots \mid C) = P(A \mid B, C \dots) * P(B \mid C \dots) * \dots \tag{2.1}$$

Given some $T \in \mathcal{T}$:

$$\begin{aligned}
&P(S_{t+m} = s_m, A_{t+m} = a_m, R_{t+m} = r_m, H_{t:t+m-1} = T \mid S_t = s_0) \\
&= \prod_{i=1}^{m} P(S_{t+i} = s_i, A_{t+i} = a_i, R_{t+i} = r_i \mid H_{t:t+i-1} = T_{i-1}, S_t = s_0) \\
&\qquad \text{by equation 2.1} \\
&= \prod_{i=1}^{m} P(S_{t+i} = s_i, A_{t+i} = a_i, R_{t+i} = r_i \mid S_{t+i-1} = s_{i-1}) \\
&\qquad \text{by the Markov Property} \\
&= \prod_{i=1}^{m} P(S_i = s_i, A_i = a_i, R_i = r_i \mid S_{i-1} = s_{i-1}) \\
&\qquad \text{by Time Invariance} \\
&= \prod_{i=1}^{m} P(S_i = s_i, A_i = a_i, R_i = r_i \mid H_{0:t+i-1} = T_{i-1}, S_0 = s_0) \\
&\qquad \text{by the Markov Property} \\
&= P(S_m = s_m, A_m = a_m, R_m = r_m, H_{0:m-1} = T \mid S_0 = s_0) \\
&\qquad \text{by equation 2.1}
\end{aligned}$$

Then, using this result:

$$
\begin{aligned}
&P(S_{t+m} = s_m, A_{t+m} = a_m, R_{t+m} = r_m \mid S_t = s_0) \\
&= \sum_{T \in \mathcal{T}} P(S_{t+m} = s_m, A_{t+m} = a_m, R_{t+m} = r_m, H_{t+1:t+m-1} = T \mid S_t = s_0) \\
&\qquad \text{by using disjoint } T \in \mathcal{T} \\
&= \sum_{T \in \mathcal{T}} P(S_m = s_m, A_m = a_m, R_m = r_m, H_{1:m-1} = T \mid S_0 = s_0) \\
&\qquad \text{by the above result} \\
&= P(S_m = s_m, A_m = a_m, R_m = r_m \mid S_0 = s_0) \\
&\qquad \text{by using disjoint } T \in \mathcal{T}
\end{aligned}
$$

This completes the proof. □

2.3. $\mathcal{M}_\pi$ are MDPs.

Theorem 2.3. *For a given environment, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, f_0, \tau, \gamma)$, and a chosen policy, $\pi \in \Pi$, $\mathcal{M}_\pi = (S_0, A_0, R_1, S_1, A_1, R_2, \dots)$ is an MDP.*

Proof. It will be sufficient to show that any $\mathcal{M}_\pi$, by construction, has the Markov Property and Time Invariance. The Markov Property follows directly from remark 1.5. We will now prove Time Invariance.

Given any $t \in \{1, 2, \dots\}$,

$$
\begin{aligned}
&P(S_t = s', A_t = a', R_t = r \mid S_{t-1} = s) \\
&= \sum_{a \in \mathbb{A}} [\pi_s(a) * \tau_{s,a}(s', r) * \pi_{s'}(a')] \text{ by remark 1.4}
\end{aligned}
$$

Note that this probability does not depend on t. By using the instance where $t = 1$, we have proven Time Invariance. Therefore, $\mathcal{M}_\pi$ is an MDP. □

We can now think of $\mathcal{M}_\pi$ as the resulting MDP from environment $\mathcal{M}$ and policy π.

3. Q-Function and Bellman's Equations

In this section[8], will further develop ideas in RL that will be necessary to describe the Q-Learning optimization algorithm.

3.1. Bellman Equation for V.

The following equation allows us to view v_π recursively.

[8]The definitions and equations in this section come from [4] and [9], but the derivations and proofs are my own unless cited.

Theorem 3.1 (Bellman for V)**.** *Given $\mathcal{M}_\pi$, $\forall s \in \mathcal{S}$,*

$$v_\pi(s) = \mathbb{E}[R_1 + \gamma v_\pi(S_1) \mid S_0 = s] \tag{3.1}$$

Proof. First, I will prove the following lemma.

Lemma 3.2. *For random variable S_1:*

$$v_\pi(S_1) = \mathbb{E}(\sum_{k=0}^{\infty} \gamma^k R_{k+1} \mid S_0 = S_1) \text{ by definition of } v_\pi \text{ 1.8}$$

Now, from the extension of time invariance 2.2, we have that

$$P(R_{k+1} \mid S_0 = S_1) = P(R_{k+2} \mid S_1)$$

This implies that

$$E(R_{k+1} \mid S_0 = S_1) = E(R_{k+2} \mid S_1)$$

Using this for each reward in the summation,

$$\mathbb{E}(\sum_{k=0}^{\infty} \gamma^k R_{k+1} \mid S_0 = S_1) = \mathbb{E}(\sum_{k=0}^{\infty} \gamma^k R_{k+2} \mid S_1)$$

This gives us

$$v_\pi(S_1) = \mathbb{E}(\sum_{k=0}^{\infty} \gamma^k R_{k+2} \mid S_1)$$

Recall from probability that for random variables A, B, C we have,

$$E[A \mid B] = E[(A \mid C) \mid B] \tag{3.2}$$

by the law of total expectation. Now,

$$v_\pi(s) = \mathbb{E}[\sum_{k=0}^{\infty} (\gamma^k R_{k+1}) \mid S_0 = s]$$

$$= \mathbb{E}[R_1 + \sum_{k=1}^{\infty} (\gamma^k R_k) \mid S_0 = s] \text{ by pulling } R_1 \text{ out of the sum}$$

$$= \mathbb{E}[R_1 + \gamma \sum_{k=1}^{\infty} (\gamma^{k-1} R_{k+1}) \mid S_0 = s] \text{ by pulling } \gamma \text{ out of the sum}$$

$$= \mathbb{E}[R_1 + \gamma \sum_{k=0}^{\infty} (\gamma^k R_{k+2}) \mid S_0 = s] \text{ by re-indexing the summation}$$

$$= \mathbb{E}[R_1 + \gamma\, \mathbb{E}(\sum_{k=0}^{\infty} \gamma^k R_{k+2} \mid S_1) \mid S_0 = s] \text{ by the law of total expectation 3.2}$$

$$= \mathbb{E}[R_1 + \gamma v_\pi(S_1) \mid S_0 = s] \text{ by lemma 3.2}$$

□

3.2. **Q-Function.**

Definition 3.3. Consider some environment, $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, f_0, \tau, \gamma)$. Then, for a choice of policy, $\pi \in \Pi$, we have the resultant MDP, $\mathcal{M}_\pi$. Following [4], we define the 'Q function' for π to be $q_\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

$$q_\pi(s, a) := \mathbb{E}[R_1 + \gamma v_\pi(S_1) \mid S_0 = s, A_0 = a]$$

The Q-Function gives the expected discounted future returns from state s given that action a is the first action taken.

Another way to think about the relationship between q_π and v_π is through the following corollary of definition 3.3:

Corollary 3.4. *For MDP, M_π, we can describe the V-Function in terms of the Q-Function:*

$$v_\pi(s) = \mathbb{E}_{A\sim\pi_s}[q_\pi(s, A)]$$

Proof. First,

$$\mathbb{E}_{A\sim\pi_s}[q_\pi(s, A)] = \mathbb{E}_{A\sim\pi_s}[\mathbb{E}(R_1 + \gamma v_\pi(S_1) \mid S_0 = s, A_0 = A)]$$

Now, note that $A \sim \pi_s$ and $(A_0 \mid S_0 = s) \sim \pi_s$. Therefore, we can replace A with $(A_0 \mid S_0 = s)$ in the expectation. So,

$$\begin{aligned}
&\mathbb{E}[\mathbb{E}(R_1 + \gamma v_\pi(S_1) \mid S_0 = s, A_0 = A)] \\
&= \mathbb{E}[\mathbb{E}(R_1 + \gamma v_\pi(S_1) \mid S_0 = s, A_0 = (A_0 \mid S_0 = s))] \\
&= \mathbb{E}[\mathbb{E}(R_1 + \gamma v_\pi(S_1) \mid A_0) \mid S_0 = s] \\
&\qquad \text{by pulling the conditioning on } S_0 \text{ to the outer expectation} \\
&= \mathbb{E}[R_1 + \gamma v_\pi(S_1) \mid S_0 = s] \text{ by the law of total expectation 3.2} \\
&= v_\pi(s) \text{ by the Bellman Equation for V}
\end{aligned}$$

□

I will now extend Bellman's equation to a version for the Q-function.

Theorem 3.5 (Bellman for Q)**.** *Given $\mathcal{M}_\pi$, $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$*

$$q_\pi(s, a) = \mathbb{E}[R_1 + \gamma q(S_1, A_1) \mid S_0 = s, A_0 = a] \tag{3.3}$$

Proof.

$$\begin{aligned}
&q_\pi(s, a) = \mathbb{E}[R_1 + \gamma v_\pi(S_1) \mid S_0 = s, A_0 = a] \\
&= \mathbb{E}[R_1 + \gamma\, \mathbb{E}(R_1 + \gamma v_\pi(S_1) \mid S_0 = S_1) \mid S_0 = s, A_0 = a] \\
&\qquad \text{by plugging in Bellman's Equation for } v_\pi(S_1) \text{ 3.2} \\
&= \mathbb{E}[R_1 + \gamma\, \mathbb{E}(R_1 + \gamma v_\pi(S_1) \mid S_0 = S_1, A_0 = A_1) \mid S_0 = s, A_0 = a] \\
&\qquad \text{by the law of total expectation 3.2 we can condition on } A_0 = A_1 \\
&= \mathbb{E}[R_1 + \gamma q_\pi(S_1, A_1) \mid S_0 = s, A_0 = a] \text{ by the definition of Q 3.3}
\end{aligned}$$

□

3.3. **Policy Improvement.** I will now take a brief detour to state and prove an important theorem in RL which we will use in section 3.4. The idea comes from Dynamic Programming and states that greedily choosing actions to improve q_π result in a global increase in returns.

Theorem 3.6 (Policy Improvement)**.** *For an MDP $\mathcal{M}$ and policy π, if there exists another policy π' such that:*

$$\forall s \in \mathcal{S}, \forall a \in supp(\pi'_s), \quad q_\pi(s,a) \geqslant v_\pi(s)$$

then

$$\forall s \in \mathcal{S}, v_{\pi'}(s) \geqslant v_\pi(s)$$

Proof. This proof will be a heavily adapted version of one from [9, p. 78]. Assume the conditions in theorem 3.6 are met. Then, for all $s \in \mathcal{S}$:

$v_\pi(s) \leqslant \mathbb{E}_{A\sim\pi'_s}[q_\pi(s,A)]$ from our assumptions

$= \mathbb{E}_{\pi'}[R_1 + \gamma v_\pi(S_1) \mid S_0 = s]$ by the same logic as the proof of corollary 3.4

$\leqslant \mathbb{E}_{\pi'}[R_1 + \gamma E[q_\pi(S_1,A)] \mid S_0 = s]$ from our assumptions

$= \mathbb{E}_{\pi'}[R_1 + \gamma\, \mathbb{E}_{\pi'}[R_1 + \gamma v_\pi(S_1) \mid S_0 = S_1] \mid S_0 = s]$

again by the same logic as the proof of corollary 3.4

$= \mathbb{E}_{\pi'}[R_1 + \gamma R_2 + \gamma^2 v_\pi(S_2) \mid S_0 = s]$ by time invariance and simplifying

$\vdots$

$$= \mathbb{E}_{\pi'}[\sum_{k=0}^{\infty} \gamma^k R_{k+1} \mid S_0 = s]$$

$$= v_{\pi'}(s)$$

In words, we are evaluating the expected returns of following π' and then π. First, using the Q-Function, we follow π' for one time step and then follow π. This increases the expected returns due to the condition. We then iteratively follow π' for more and more steps before reverting to π. As we do this, the expected returns continue to increase due to the condition. We continue this process until we hit a terminating state with high probability which we know will happen by remark 1.7. All rewards after that state will be 0. So, all nonzero rewards will have been accumulated by acting on π'. This is equal to $v_{\pi'}$ by definition. □

Corollary 3.7 (Strict Policy Improvement)**.** *If the conditions described in the policy improvement theorem 3.6 hold and also*

$$\exists s' \in \mathcal{S}, \forall a \in supp(\pi'_s), \quad q_\pi(s',a) > v_\pi(s')$$

then

$$\forall s \in \mathcal{S}, v_{\pi'}(s) > v_{\pi}(s)$$

Proof. The idea here is similar. However, additionally, following π' at one state s' strictly improves the rewards. Furthermore, this state has a nonzero probability of being reached by our assumption of irreducibility 1.6 and so will strictly increase the expected future returns for all other states.

These conditions will result in a string of relationships which is identical to the one in the proof of the theorem except that at least one of the '$\leqslant$' will be a strict '$<$'. So, the resulting inequality will be identical but strict. □

3.4. **Bellman Optimality.** Now, given some environment, M, let us suppose the existence of some $\pi^* \in \Pi^*$ and define $q^* := q_{\pi^*}$ and $v^* := v_{\pi^*}$. Our objective is to first identify q^* and then use it to construct some $\hat{\pi} \in \Pi^*$[9].

Assuming the ability to characterize q^*, we will use q^* to build an optimal $\hat{\pi} \in \Pi^*$. We will discuss the characterization of q^* afterward.

Definition 3.8. Given q^*, we construct $\hat{\pi}$ in the following way: For each state $s \in \mathcal{S}$ we restrict

$$supp(\hat{\pi}_s) \subset argmax_{\beta} q^*(s, \beta)$$

Theorem 3.9. *Any $\hat{\pi}$ constructed in this way is an optimal policy: $\hat{\pi} \in \Pi^*$.*

Proof. By how $\hat{\pi}$ is defined, given any $s \in \mathcal{S}$, $supp(\hat{\pi}_s) \subset argmax_{\beta} q^*(s, \beta)$. So, $\forall a \in supp(\hat{\pi}_s)$,

$$\begin{aligned} q^*(s,a) &= max_{\beta} q^*(s, B) \text{ by how } \hat{\pi} \text{ is constructed} \\ &\geqslant \mathbb{E}_{A \sim \pi^*}[q^*(s, A)] \text{ because } q^*(s,a) \text{ is maximized} \\ &= v^*(s) \text{ by corollary 3.4} \end{aligned}$$

So, by the Policy Improvement Theorem 3.6, we have that $\forall s \in \mathcal{S}$, $v_{\hat{\pi}}(s) \geqslant v^*$. Then,

$$\begin{aligned} \mathcal{O}(\hat{\pi}) &= E_{S_0 \sim f_0}[v_{\hat{\pi}}(S_0)] \text{ by the definition of the objective function} \\ &\geqslant E_{S_0 \sim f_0}[v^*(S_0)] \text{ by the above result} \\ &= \mathcal{O}(\hat{\pi}) \text{ by the definition of the objective function} \\ &= max_{\pi} \mathcal{O}(\pi) \text{ by how } \pi^* \text{ is defined} \end{aligned}$$

So, $\hat{\pi} \in \Pi^*$. □

[9]Interestingly, this $\hat{\pi}$ is not necessarily the same as the original π^*

Now, we focus on the identification of q^* so that $\hat{\pi}$ can be constructed. First we will prove an important property relating v^* to q^*.

Theorem 3.10. *Given $\mathcal{M}$ and π^*, it must be the case that $\forall s \in \mathcal{S}$:*

$$v^*(s) = max_\beta q^*(s, \beta) \tag{3.4}$$

Proof. We will first prove the '$\leqslant$' direction.

$$max_\beta q^*(s,\beta) \geqslant \mathbb{E}_{A\sim\pi_s^*}[q^*(s,A)] \text{ because } q^*(s,a) \text{ is maximized}$$
$$= v^*(s) \text{ by corollary 3.4}$$

We will now prove the '$\geqslant$' direction. Suppose toward contradiction that $\exists s' \in \mathcal{S}$ such that $V^*(s') < max_\beta q^*(s', \beta)$. Now, we construct $\hat{\pi}$ from q^* as described in 3.8. By construction of $\hat{\pi}$, $\forall a \in supp(\hat{\pi_{s'}})$ we have

$$q^*(s', a) = max_\beta q^*(s', \beta) > V^*(s')$$

Furthermore, note that $v_{\hat{\pi}}(s) \geqslant v^* \quad \forall s \in \mathcal{S}$ as a direct result of the '$\leqslant$' direction. So, by the strict policy improvement corollary 3.7, we have that $\forall s \in \mathcal{S}$, $v_{\hat{pi}}(s) > v^*(s)$. So,

$$\mathcal{O}(\hat{\pi}) = E_{S_0\sim f_0}[v_{\hat{\pi}}(S_0)] \text{ by the definition of the objective function}$$
$$> E_{S_0\sim f_0}[v^*(S_0)] \text{ by the above result}$$
$$= \mathcal{O}(\hat{\pi}) \text{ by the definition of the objective function}$$
$$= max_\pi \mathcal{O}(\pi) \text{ by how } \pi^* \text{ is defined}$$

This is obviously a contradiction. This proves the '$\geqslant$' direction. □

The following 'Bellman for Q^*' is a crucial characterization of q^* and a core piece of the Q-Learning algorithm described in the next section.

Theorem 3.11 (Bellman for Q^*)**.** *Given $\mathcal{M}$ and π^*:*

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A} \quad q^*(s,a) = \mathbb{E}[R_1 + \gamma max_\beta q^*(S_1, \beta) \mid S_0 = s, A_0 = a] \tag{3.5}$$

is a necessary and sufficient condition for q^*.

Proof. Proof of necessary condition: By the definition of Q 3.3 we have

$$q^*(s,a) = \mathbb{E}[R_1 + \gamma v^*(S_1) \mid S_0 = s, A_0 = a]$$

We then simply use equation 3.4 to replace $v^*(S_1)$ with $max_\beta q^*(S_1, \beta)$. The proof of the condition being sufficient is the same but in reverse. □

This process of identifying q^* and then constructing $\hat{\pi} \in \Pi^*$ from q^* is the core idea of the Q-Learning Algorithm.

4. Bellman Operator and Q-Learning Algorithm

I will now begin to describe the main components of the algorithm.[10]

[10]This section contains heavily adapted material found in [4] and [9].

4.1. **Bellman Operator.** I will use the Bellman Equation for Q^* to define the Bellman Operator. This will be the key to Q-Learning.

Given environment $\mathcal{M}$, consider the set of functions $F = \{f : S \times A \to \mathbb{R}\}$. We define a metric on F to be $d : F \times F \to \mathbb{R}$,

$$d(f, g) := max_{\mathcal{S}\times\mathcal{A}} \mid f(s, a) - g(s, a) \mid$$

We define the metric space F_d to be the set F with metric d.

Lemma 4.1. *F_d is a nonempty, complete metric space.*

Proof. First, F_d is nonempty: consider $f(s, a) = 0 \ \forall s \in \mathcal{S}, a \in \mathcal{A}$.

Now, note that F_d, the metric space of functions $\{f : S \times A \to \mathbb{R}\}$ is isomorphic to $\mathbb{R}_d^{|S|*|A|}$—the metric space induced by the normed vector space of $\mid S \mid * \mid A \mid$ dimensional real vectors with a *max* norm. R^n is complete with respect to any norm and so is complete with respect to the max norm. So, F_d is complete. □

Definition 4.2. [4] Given environment, $\mathcal{M}$, the Bellman operator is a function $\mathbb{B} : F_d \to F_d$,

$$\mathbb{B}(f)(s, a) := \mathbb{E}[R_1 + \gamma max_\beta f(S_1, \beta) \mid S_0 = s, A_0 = a]$$

Remark 4.3. Note that for a function $f \in F_d$, being a fixed point of this operator, $\mathbb{B}(f) = f$ is equivalent to satisfying the Bellman Equation for Q^*. Theorem 3.11 specifies that the Bellman Equation for Q^* is a necessary and sufficient condition for q^*. It follows that q^* being a fixed point of the Bellman Operator, $\mathbb{B}(q^*) = q^*$, is a necessary and sufficient condition for q^*.

4.2. **Bellman Operator Convergence.**

Lemma 4.4. *The Bellman Operator, $\mathbb{B}$ is a contraction mapping on F_d with Lipschitz constant γ.*

Proof. Following [4], for $f, g \in F$:

$$
\begin{aligned}
& d(\mathbb{B}(f), \mathbb{B}(g)) \\
&= max_{\mathcal{S}\times\mathcal{A}} \mid \mathbb{E}[R_1 + \gamma max_\beta f(S_1, \beta) \mid S_0 = s, A_0 = a] - \\
&\qquad \mathbb{E}[R_1 + \gamma max_\beta g(S_1, \beta) \mid S_0 = s, A_0 = a] \mid \\
&= \gamma max_{\mathcal{S}\times\mathcal{A}} \mid \mathbb{E}[max_\beta f(S_1, \beta) - max_\beta g(S_1, \beta) \mid S_0 = s, A_0 = a] \mid \\
&\qquad \text{by linearity of expectation} \\
&\leqslant \gamma max_{\mathcal{S}\times\mathcal{A}} \mathbb{E}[\mid max_\beta f(S_1, \beta) - max_\beta g(S_1, \beta) \mid S_0 = s, A_0 = a \mid] \\
&\qquad \text{because} \mid \mathbb{E}[X] \mid \leqslant \mathbb{E}[\mid X \mid] \\
&\leqslant \gamma max_{\mathcal{S}\times\mathcal{A}} \mathbb{E}[max_\beta \mid f(S_1, \beta) - g(S_1, \beta) \mid S_0 = s, A_0 = a \mid] \\
&\qquad \text{because} \mid max_\beta X(\beta) - max_\beta Y(\beta) \mid \leqslant max_\beta \mid X(\beta) - Y(\beta) \mid \\
&\leqslant \gamma max_{\sigma,\beta} \mid f(\sigma, \beta) - g(\sigma, \beta) \mid \\
&\qquad \text{because maximizing is larger than any expectation} \\
&= \gamma d(f, g)
\end{aligned}
$$

□

Theorem 4.5 (Convergence of $\mathbb{B}$)**.** *Given $\mathcal{M}$, for any $f_0 \in F_d$, $\exists \pi^* \in \Pi^*$ such that*

$$\lim_{n\to\infty} \mathbb{B}^n(f_0) = q^*$$

Proof. We know from lemmas 4.1 and 4.4 that $\mathbb{B}$ is a contraction mapping on a nonempty, complete metric space. So, by Banach's Fixed Point Theorem, we have that $\mathbb{B}$ has a unique fixed point $f^* \in F_d$ and $\forall f_0 \in F_d$,

$$\lim_{n\to\infty} \mathbb{B}^n(f_0) = f^*$$

From remark 4.3, we know that $f^* = q^*$. □

This is the main result driving the Q-Learning Algorithm. Furthermore, this proof gives us the following corollaries.

Corollary 4.6. *As a corollary to the proof of Theorem 4.5 we have the existence and uniqueness of q^*.*

Proof. By 4.3, $\mathbb{B}(q^*) = q^*$ is a necessary and sufficient condition for q^*. Furthermore, in the proof of 4.5, Banach's Fixed Point Theorem, gives us that the fixed point of $\mathbb{B}$ both exists and is unique. Therefore, q^* must both exist and be unique. □

Corollary 4.7. *As a corollary to Corollary 4.6 we have that Π^* is nonempty, and all $\pi^* \in \Pi^*$ give us the same q^* and v^*.*

Proof. Corollary 4.6 gives us the existence and uniqueness of q^*. We know from our discussion of $\hat{\pi}$ in Section 3 that from this q^* we can construct some

$\hat{\pi} \in \Pi^*$. So, Π^* must be nonempty. Furthermore, because q^* is unique, any $\pi^* \in \Pi^*$ must give us the same q^*. Then, because $v^*(s) = max_\beta q^*(s, \beta)\ \forall s$ by equation 3.4, all $\pi^* \in \Pi^*$ must give us the same v^* as well. □

4.3. **Estimator for Bellman Operator.** Ideally, Q-Learning would work by repeatedly applying the Bellman Operator to some $f \in F_d$ until it converges to q^*. Note, however, that while the Bellman Operator has the convergence guarantees we need, the MDP is not available to evaluate analytically—we can only sample from $\mathcal{M}_\pi$. As such, we need to sample from $\mathcal{M}_\pi$ and approximate $\mathbb{B}$ using a statistical estimator. Specifically, we will use single samples to estimate $\mathbb{B}(f)(s, a)$:

$$\hat{\mathbb{B}}(f)(s,a) := (R_1 + \gamma max_\beta f(S_1, \beta) \mid S_0 = s, A_0 = a)$$
$$= (R_{t+1} + \gamma max_\beta f(S_{t+1}, \beta) \mid S_t = s, A_t = a) \text{ by Time Invariance 2.2}$$

as a replacement for $\mathbb{B}$.

This begs the question: does $\hat{\mathbb{B}}$ have the same convergence properties? I will not prove this result, but in his 1994 paper, Tsitsiklis [10] proved that such an estimator for $\mathbb{B}$ has the same convergence properties as long as the following conditions hold:

(1) Every state-action pair is sampled a large number of times
(2) The estimator is unbiased[11]
(3) The variance of the estimator at after t iterations is bounded above by $A + B * [max_{s,a} q_t(s, a)]^2$ for some constants A and B where $q_t \in F$ is the starting function after t applications of $\hat{\mathbb{B}}$.

By repeatedly drawing samples from the MDP, our assumption of irreducibility 1.6 guarantees that each state-action pair will be sampled a large number of times. Furthermore, our estimator is unbiased as it is the MLE estimator for a sample of size 1.

Remark 4.8. It is important to note that the estimator's adherence to the bounds on variance ultimately relies on the given environment and so cannot be guaranteed. In fact, it is often the case that this bound is not met, causing failure in finding q^*. Reducing this variance is very much an open area of research in RL and Machine Learning in general. Often, variance is reduced by introducing a simplifying bias in the estimator. Obviously, introducing bias into the estimator will cause problems with accuracy, and so the key becomes finding the right trade-off between low bias and low variance.

[11] In other words, the expectation of the estimator is equal to the parameter it is estimating

4.4. **Statement of the Algorithm.** We are now ready to describe the Q-Learning Algorithm. Given some environment, $\mathcal{M}$, the algorithm works by initializing $\hat{q} \in F$, choosing some training policy π, and then iteratively sampling from $\mathcal{M}_\pi$, and applying $\hat{\mathbb{B}}$ to $\hat{q}$. This process is repeated for many episodes until $\hat{q}$ converges to q^*. We then obtain $\hat{\pi} \in \Pi^*$ using the construction from definition 3.8.

Algorithm 1 Q-Learning [9, p.131-132]

Choose number of episodes, $N \in \mathbb{N}$
$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{R}, f_0, \tau, \gamma)$
Initialize $\hat{q}(s,a) = 0$, $\forall s \in \mathcal{S}, a \in \mathcal{A}$
for N episodes **do**
  Initialize $t \leftarrow 0$ and sample $s_0 \sim f_0$
  while $s_t \notin \mathcal{S}_\infty$ **do**
    $a_t \sim Uniform(\mathcal{A})$
    Sample $(s_{t+1}, r_{t+1}) \sim \tau_{s_t, a_t}$
    $\hat{q}(s_t, a_t) \leftarrow r_{t+1} + \gamma max_\beta \hat{q}(s_{t+1}, \beta)$ Application of $\hat{\mathbb{B}}$
    $t \leftarrow t + 1$
  end while
end for
$\hat{\pi}(s) \leftarrow \beta \in argmax_\beta \hat{q}(s, \beta)$, $\forall s \in \mathcal{S}$
return $\hat{\pi}$

5. Implementation of Q-Learning for a Maze

5.1. **Additional Features.** Before jumping into specific aspects of my implementation, I will take a moment to briefly address some common modifications to Q-Learning (algorithm 1) that improve stability and convergence rates. While many variations of the algorithm exist, here I will focus on two basic modifications that I use in the implementations seen in this paper.

The first modification involves a learning rate $\alpha \in (0, 1)$.

Definition 5.1. We use a different estimator for $\mathbb{B}$:

$$\hat{\mathbb{B}}_\alpha(f)(s,a) := (1-\alpha) f(s,a) + \alpha \hat{\mathbb{B}}(f)(s,a)$$

This estimator precisely follows the ideas in remark 4.8 in reducing the variance of $\hat{\mathbb{B}}$ by introducing a simplifying bias: the original value of $f(s,a)$. The idea is that this simplifying bias will not prohibit convergence, because it corresponds to 'skipping an iteration of the operator'. We can see that the

variance has been reduced,

$$\begin{aligned} &var[\hat{\mathbb{B}}_\alpha(f)(s,a)] = var[(1-\alpha)f(s,a) + \alpha\hat{\mathbb{B}}(f)(s,a)] \\ &= \alpha^2 var[\hat{\mathbb{B}}(f)(s,a)] \text{ because } f(s,a) \text{ is a constant} \\ &< var[\hat{\mathbb{B}}(f)(s,a)] \text{ because } \alpha \in (0,1) \end{aligned}$$

but bias has been introduced,

$$\begin{aligned} &\mathbb{E}(\hat{\mathbb{B}}_\alpha(f)(s,a)) = \mathbb{E}[(1-\alpha)f(s,a) + \alpha\hat{\mathbb{B}}(f)(s,a)] \\ &= (1-\alpha)\,\mathbb{E}[f(s,a)] + \alpha\,\mathbb{E}[\hat{\mathbb{B}}(f)(s,a)] \\ &= (1-\alpha)\,\mathbb{E}[f(s,a)] + \alpha\mathbb{B}(f)(s,a) \text{ because } \hat{\mathbb{B}} \text{ is unbiased} \\ &\neq \mathbb{B}(f)(s,a)] \text{ unless } f(s,a) = \mathbb{B}(f)(s,a) \end{aligned}$$

The second modification involves constructing a new training policy π at the beginning of each episode. The reasoning for this is as follows:

Recall that the Bellman Operator guarantees convergence to $q^*(s,a)$ for *all* state-action pairs. However, if our goal is just to identify an optimal policy, then $q^*(s,a)$ actually provides more information than we need. In order to find $\pi^* \in \Pi^*$, for each state, $s \in \mathcal{S}$ we just need to identify *one* action $a \in \mathcal{A}$, such that $a \in argmax_\beta q^*(s,\beta)$.

Now, if we make the heuristic assumption that earlier versions of $\hat{q}$ tell us something about q^* then it would make sense to sample more frequently from state-action pairs which satisfy $a \in argmax_\beta \hat{q}(s,\beta)$ because they are 'more likely' to ultimately satisfy $a \in argmax_\beta q^*(s,\beta)$. So, we adjust our training policy to prioritize sampling from these state-action pairs. In doing this, we aim to reduce the number of iterations it takes to identify $\pi^* \in \Pi^*$. However, generally when doing this, we leave some noise in the policy so that in the long term, we still sample from all the state-action pairs.

Definition 5.2. To implement this modification, we choose a noise parameter $\epsilon \in (0,1)$ and, at the beginning of each episode, we construct a new training policy π^ϵ such that $\forall s \in \mathcal{S}$:

$$\pi^\epsilon_s := \begin{cases} Uniform(\mathcal{A}) & \text{with probability } \epsilon \\ Uniform(argmax_\beta \hat{q}(s,\beta)) & \text{with probability } 1-\epsilon \end{cases}$$

for the current $\hat{q}$.

This results in the full Q-Learning algorithm:

Algorithm 2 Q-Learning Implementation [9, p.131-132]

Choose $N \in \mathbb{N}$, $\alpha \in (0,1)$, and $\epsilon \in (0,1)$
$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathbb{R}, f_0, \tau, \gamma)$
Initialize $\hat{q}(s,a) = 0$, $\forall s \in \mathcal{S}, a \in \mathcal{A}$
for N episodes **do**
  Initialize $t \leftarrow 0$ and sample $s_0 \sim f_0$
  while $s_t \notin \mathcal{S}_\infty$ **do**
    $a_t \sim \pi^\epsilon_{s_t}$
    Sample $(s_{t+1}, r_{t+1}) \sim \tau_{s_t,a_t}$
    $\hat{q}(s_t,a_t) \leftarrow (1-\alpha)\hat{q}(s_t,a_t) + \alpha[r_{t+1} + \gamma max_\beta \hat{q}(s_{t+1},\beta)]$ (Apply $\hat{\mathbb{B}}_\alpha$)
    $t \leftarrow t+1$
  end while
end for
$\hat{\pi}(s) \leftarrow \beta \in argmax_\beta \hat{q}(s,\beta)$, $\forall s \in \mathcal{S}$
return $\hat{\pi}$

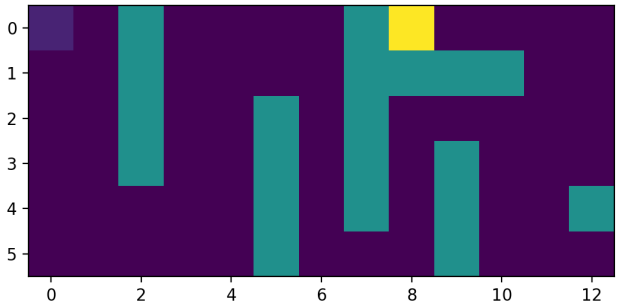


FIGURE 1. Maze

5.2. **The Maze.** I will now implement Q-Learning in order to solve the maze in Figure 1. The top left light purple spot is the start, the yellow is the goal, the light green are holes, and the dark purple are normal spaces. The objective is to find a policy which safely gets the agent to the goal without falling off the edge of the grid or down a hole.

5.3. **RL Formulation.** We can model this environment as

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, f_0, \tau, \gamma)$$

where

- $\mathcal{S} = \{normal_i\} \cup \{start, goal, fall\}$. The goal state is a terminal state representing success and the fall state is a terminal state representing failure.
- $\mathcal{A} = \{up, down, left, right\}$
- $\mathcal{R} = \{100, -100, -1, 0\}$
- $$f_0(s) = \begin{cases} 1 & \text{for } start \\ 0 & \text{else} \end{cases}$$
  In words, the agent always starts in the starting state.
- τ: For all state-action pairs, $\tau_{s,a}$ is deterministic:
  - If the agent is in the fall or goal state (terminal states) then it receives a reward of 0 and stays in that state regardless of the action taken.
  - If the agent is in a normal state or start state:
    - If the agent moves toward a normal state or start state then it arrives at the space and receives a reward of -1.
    - If the agent moves off the board or into a hole it receives a reward of -100 and goes to the fall state.
    - If the agent moves toward the goal then it receives a reward of 100 and goes to the goal state.
- $\gamma = .99$

To solve this problem, we will use algorithm 2. This maze MDP is fairly simple so we choose

- a fairly high learning rate $\alpha = .5$
- low noise $\epsilon = .01$
- a modest number of episodes $N = 500$

5.4. **Results.** Two important remarks regarding how we evaluate the progress of Q-Learning as it is occurring:

Remark 5.3. While running Q-Learning, π^0 (i.e. π^ϵ where $\epsilon = 0$) represents the best policy we can construct given our current $\hat{q}$. As such, we will use it to represent our best approximation of π^* at that point in time. We will test the current π^0 periodically on the environment while training to assess how our search for $\pi^* \in \Pi^*$ is progressing.

Remark 5.4. While running Q-Learning, $\hat{q}$ is our best approximation of q^* so it is natural that, from equation 3.4, we will use

$$\hat{v} := max_\beta \hat{q}(s, \beta)$$

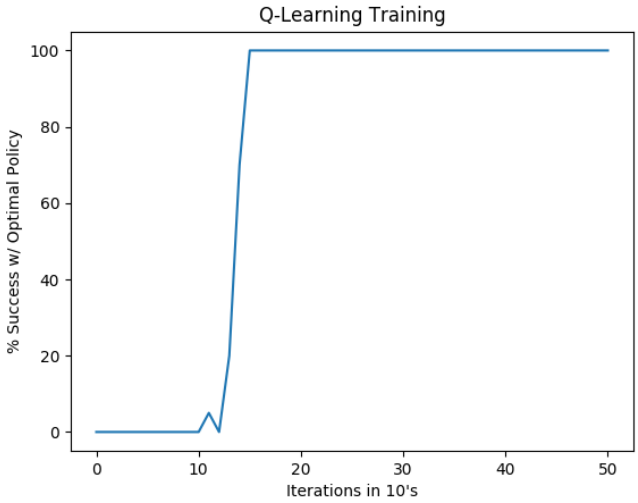


FIGURE 2. π^0 Success Rate

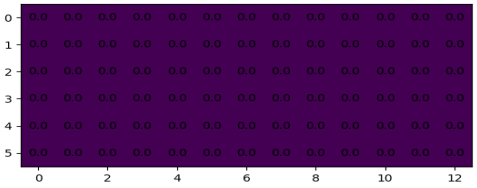
as our best approximation of v^*. Furthermore, it follows from the way in which $\hat{v}$ and π^0 are derived from $\hat{q}$, that $\hat{v}$ is the value function for π^0. In other words, $\hat{v}(s)$ represents our best estimate of the expected discounted returns from s while following π^0.

Figure 2 plots the (average over 20 runs) percentage of the time that the agent was able to find the goal using π^0 after i episodes. In running Q-Learning 20 times, the agent's final policy, $\hat{\pi}$ reached the goal 100% of time in all 20 runs. So, clearly Q-Learning is successful in consistently finding a policy which solves the maze.

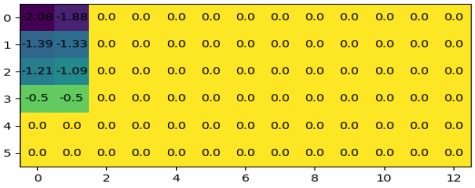
Figure 3 shows (the average over 20 runs) $\hat{v}(s)$ after 0, 10, 100, and 250 episodes.It is interesting to note that around 100 episodes:

(1) In Figure 2, the agent solves the maze (i.e. π^0 reaches 100% accuracy)
(2) In Figure 3, the heat map of $\hat{v}$ after 100 episodes is beginning to look like the maze.

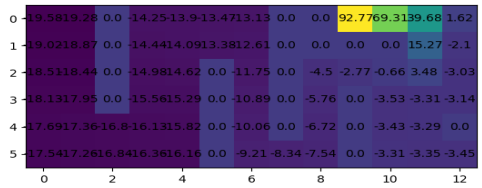
So, it seems that at around the 100 episode mark, the agent both has an accurate schema of the maze and its rewards (2) and is able to use that schema to solve the maze (1).



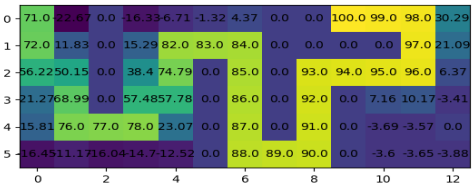
After 0 Episodes



After 10 Episodes



After 100 Episodes



After 250 Episodes

FIGURE 3. Heat Map of $\hat{v}(s)$

6. Naive Q-Learning for Tic-Tac-Toe

The results of Section 5 indicate that, given a well defined $\mathcal{M}$, Q-Learning works fairly well. However, a natural question follows: in what situations is the MDP model sufficient and in what situations do the model assumptions break down?

For example, perhaps I would like to use Q-Learning to identify a policy which plays Tic-Tac-Toe optimally. Tic-Tac-Toe is notably different from the maze example because *the opponent is actively working against you*. Let us suppose we have the specific objective of playing Tic-Tac-Toe optimally against an optimal opponent.

A natural approach to solving this problem is to have two separate RL agents play against each other. As each one improves, the other one will have to adapt, and, ideally, each will eventually identify an optimal playing policy. In this section, I implement this naive approach to such a multi-agent system and discuss the results.

6.1. **The Game.**

1	2	3
4	5	6
7	8	9

There are 9 spots on a Tic-Tac-Toe board, each of which can be filled with an X, 0, or be left blank. The board starts off as all blank and then each player (starting with X and then 0) takes turns placing their piece. Players can only place pieces on blank spaces and pieces cannot be removed. A player wins (and the other player loses) if the board has three of their pieces in a row (horizontally, vertically, or diagonally) and the game immediately ends. If the board is filled without this happening, then the players tie.

6.2. **RL Formulation.** For the first player (player X), we will assume that the opponent (player O) is playing their current best strategy, π^0 which we will denote π^O. Following with this notation, we will refer to player X's current best strategy π^0 as π^X. From player's X perspective, we will refer to the state of the board *after* X goes but *before* O goes as a 'post-state'. Note that, from player O's perspective, this post-state is just the given state.

Now we can model player X's environment as

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{R}, f_0, \tau, \gamma)$$

where

- $\mathcal{S} = \{(b_1, \dots, b_9) \mid b_i \in \{X, O, '\quad'\}\}$. The terminal states are the ones for which the game has ended, as defined above.
- $\mathcal{A} = \{1, \dots, 9\}$[12]
- $\mathcal{R} = \{-10, 0, 10\}$
- $$f_0(s) = \begin{cases} 1 & \text{for the empty board} \\ 0 & \text{else} \end{cases}$$
  In words, the agent always starts with the empty board.
- τ: we can describe $\tau_{s,r}$ in the following way:
  - If the agent is in a terminal state, then it receives a reward of 0 and stays in that state regardless of the action taken.
  - If the agent's move wins the game, then it receives a reward of 10 and goes to that terminal state.
  - If the agent's move ties the game, then it receives a reward of 0 and goes to that terminal state.
  - Otherwise: First the board changes according to the action taken by X to post-state, s'. then 0 takes its action using $\pi^O_{s'}$:

[12]In reality, the actions are actually restricted to the *open* spots on the board and so are dependent on the current state. We will not worry about this however, because in practice we can just remove these actions and everything else will follow identically.

    - If 0's move wins it the game, then X loses. So, X receives a reward of -10 and goes to that terminal state.
    - If 0's ties the game then X receives a reward of 0 and goes to that terminal state.
    - Otherwise, X receives a reward of 0 and the board changes again according to O's action.
- $\gamma = .999$

Remark 6.1. It is important to note the 'naive' aspect of this environment: the transitions are dependent on π^O which changes over time. This breaks the model assumption that $\mathcal{M}$ is provided first and is unchanging.

Player O's environment is identical with the roles of X and 0 flipped except that:

$$f_0 = \pi_s^X$$

where s is the empty board.

To solve this problem, each agent will use algorithm 2.

The Tic-Tac-Toe setting is far more complicated than the maze with $\mid \mathcal{S} \mid = 3^9 \approx 20,000$ and two agents, so we conservatively choose

- a low learning rate $\alpha = .1$ to reduce variance
- high noise: $\epsilon = 1$ to maximize exploration
- a large number of episodes $N = 10,000$

for both agents.

6.3. **Results.** Part of the difficulty in assessing the results of a multi-agent system such as Tic-Tac-Toe is that the notion of a *system-wide objective*[13] is not well defined. Furthermore, even once we have identified a system-wide objective, we still have the difficulty of checking whether that system-wide objective has been achieved.

Remark 6.2. One useful fact about Tic-Tac-Toe is that, no matter the opponent's strategy, each player can always at least force a tie. Because this is true for both players, it must be that two optimal players will always tie. In other words, consistently tying is a necessary (but insufficient) condition for both agents being optimal. As such, we use the rate of tying, along with a qualitative assessment of gameplay, as an approximate assessment of optimality.

Now, Figure 4 plots the (average over 20 runs) percentage of wins, losses, and ties for *Player X* using π^X against π^O after i iterations. As we were

[13] I will address this issue in the next section.

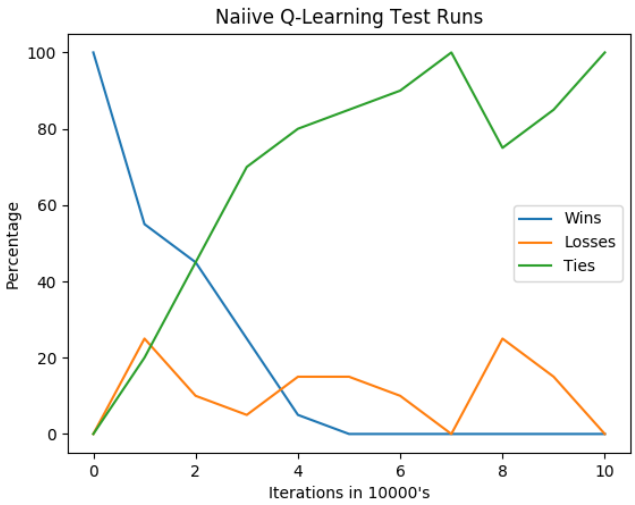


FIGURE 4. Game Results for π^X vs. π^O

hoping, by the end, the agents are mostly tying.

Furthermore, Figure 5 is a randomly selected game at the end of training to assess qualitatively whether the agents seem to be playing optimally. The game is read left to right and recall that 'X' always goes first. As we can observe, the players seem to be playing optimally and without error.

To better understand how the agents are making decisions, Figure 6 shows (averaged over 20 runs) $\hat{q}(s, a)$ at the end of training at important states for each agent.

The left heat map shows $\hat{q}(s, a)$ across actions for player X where s is fixed as the empty board. Note that 5 is considered the optimal action, followed by the corners, and finally the middle edges are deemed the worst. While these values correspond to what we would expect from an optimal strategy, the symmetry of the game indicates that some of the discrepancies in $\hat{q}(s, a)$ between actions are somewhat arbitrary.

The right heat map shows $\hat{q}(s, a)$ across actions for player O where s is fixed as $(X,'\ \ ', \dots)$ (i.e. the board after player X's first action is in the top left).

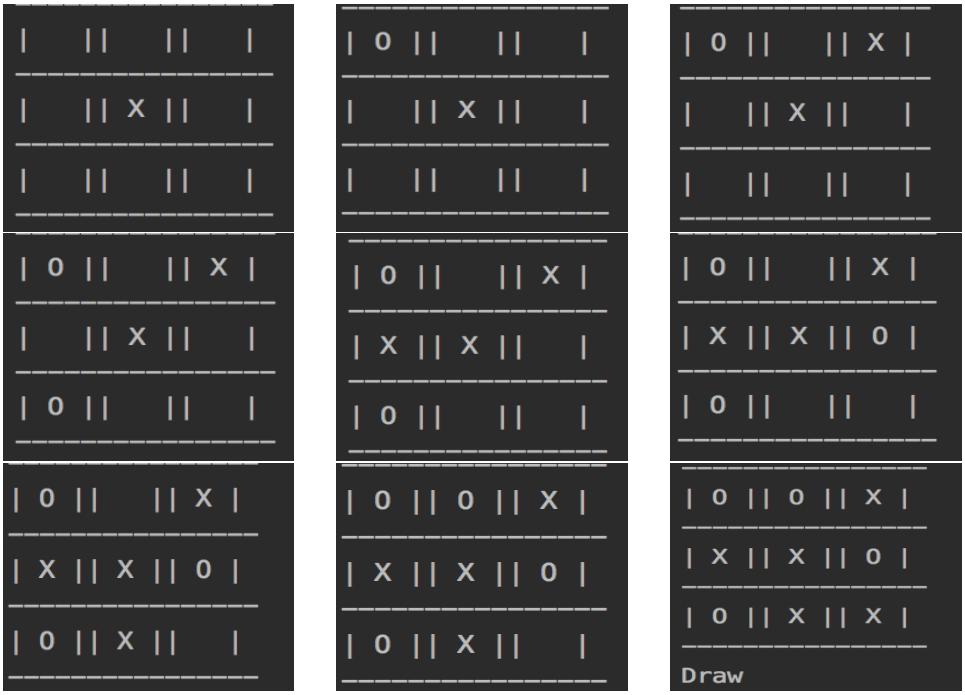


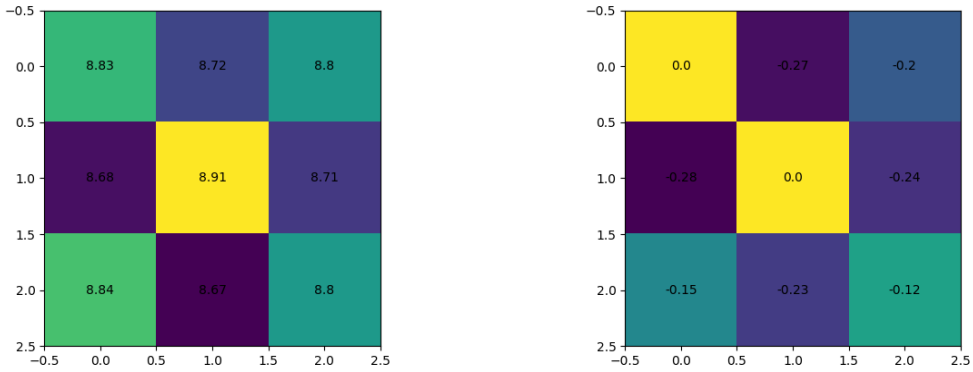
FIGURE 5. Game Played After Full Training

[14] We see that the best actions in decreasing order are 5 (which evidently forces a draw), 9, 3 & 7, 6 & 8, and finally, 2 & 4. We can also observe the same presence of discrepancies in $\hat{q}(s, a)$ between actions that should be identical due to symmetry.

Despite the somewhat positive results of this naive approach, it is important to note its shortcomings[15]. If we look at Figure 4, there is some noticeable instability throughout the training, and win rates continue to fluctuate even after 100,000 episodes. This is distinct from the very efficient and stable results seen above in the single-agent maze problem. Furthermore, the heat maps in Figure 6 seem to indicate that, even at the end, $\hat{q}$ did not fully capture the symmetry of the game.

These shortcomings are understandable, as we have naively thrown two independent Q-Learning agents at each other. In the following sections, I will

[14]Note that the top left square is not actually a viable action now and so should be ignored.
[15]I was surprised that this approach worked at all. I attribute it to the relative simplicity of Tic-Tac-Toe. In more complicated multi-agent settings, such an approach will likely fail.



Player X at Empty Board Player O After First Action is 1

FIGURE 6. $\hat{q}(s, a)$ Across Actions at Critical States

explore multi-agent systems more carefully, which will lead to a more efficient and stable alternative to this naive implementation.

7. Introduction to Multi-Agent RL

Now, we will extend the theory we have developed to multi-agent systems and formalize some ideas introduced in the previous section. While the task formulation and model description are natural extensions of the single agent case, defining the optimization problem requires a slightly more nuanced analysis. [16]

7.1. **Task Formulation.**

The Multi-Agent RL task can be formulated as follows:

(1) Environment $\mathcal{M}$ is received.
(2) The following steps are repeated:
   (a) m different agents each choose a policy π^j for each $j \in \{1, \ldots, m\}$
   (b) Model $\mathcal{M}_\pi$ is constructed.
   (c) $\mathcal{M}_\pi$ is sampled from in order to gain information about $\mathcal{M}$ and π
   (d) Repeat until some optimal vector of policies π^* is found

$\mathcal{M}$ is now in the form of a 7-tuple,

$$\mathcal{M} := (m, \mathcal{S}, \mathcal{A}, \mathcal{R}, f_0, \tau, \gamma)$$

where:

(1) m is the number of agents.

[16] This section builds on my work in Section 1 using concepts from [1] and [7].

(2) $\mathcal{A} = \times_{j=1}^{m} \mathcal{A}^j$ is a cartesian product of m finite sets of actions.
(3) $\mathcal{R} \subset \mathbb{R}^m$ is a finite set where the elements are reward vectors.
(4) The rest are the same as in definition 1.1.

Now, a policy vector π is the vector

$$\pi := (\pi^1, \dots, \pi^m)$$

where each π^j is the single agent policy for agent j:

$$\pi^j := \{\pi^j_s(a) \mid s \in \mathcal{S}\}$$

Remark 7.1. It is important to remark that the *ordering* of how the agents choose their policies matters. For example, an environment where each agent $\{1, \dots, m\}$ chooses their policy sequentially will be different than one where they all choose simultaneously. A sequential setup would allow, for example, agent 5 to make their policy dependent on the policy choice of agent 3. Unless otherwise specified, we assume that each agent chooses their policy simultaneously.

We will refer to the set of possible policies for agent j as

$$\Pi^j := \{\pi^j\}$$

and the set of possible policy vectors:

$$\Pi := \bigtimes_{i=j}^{m} \Pi^j$$

7.2. **Model Description.** Given a multi-agent environment $\mathcal{M}$ and policy vector $\pi \in \Pi$, we build $\mathcal{M}_\pi$ very similarly to how we did in the single agent case:

First, let $S_0 \sim f_0$. Now, for each agent $\jmath \in \{1, \dots, m\}$, let $A_0^j \sim \pi^i_{S_0}$. Then, let $(S_1, R_1) \sim \tau_{S_0, A_0}$. Continuing this process, we can recursively define:

(1) $\forall j \in \{1, \dots, m\}, \quad A_i^j \sim \pi^j_{S_i}$ for $i \in \{0, 1, \dots, N\}$,
(2) $(S_i, R_i) \sim \tau_{S_{i-1}, A_{i-1}}$ for $i \in \{1, 2, \dots\}$.

Again, this gives us a joint distribution for the following sequence of random variables:

$$\mathcal{M}_\pi := (S_0, A_0, R_1, S_1, A_1, R_2, \dots)$$

The only difference is that, now, A_i, R_i, and π are all vectors of length m.

Just like in the single-agent formulation, $\mathcal{M}_\pi$ forms an MDP and so much of the theory can be extended naturally. The main complication of multi-agent setting stems from the fact that each agent, j, has its own rewards $(R_0^j, R_1^j, \dots)$ and so each agent is optimizing *its own value function.*

7.3. **Value Functions and Optimization.** Given some environment, $\mathcal{M}$, each agent j chooses π^j which results in the policy vector π. Then, each agent j has its individual value function, $v_\pi^j : \mathcal{S} \to \mathbb{R}$,

$$v_\pi^j(s) := \mathbb{E}[\sum_{k=0}^{\infty}(\gamma^k R_{k+1}^j \mid S_0 = s)]$$

Using this, we will represent the global value function as $v_\pi : \mathcal{S} \to \mathbb{R}^m$

$$v_\pi(s) := (v_\pi^1(s), \ldots, v_\pi^m(s))$$

Given an environment, $\mathcal{M}$, we define the objective vector, $\mathcal{O} : \Pi \to \mathbb{R}^m$:

$$\mathcal{O}(\pi) := (\mathbb{E}[v_\pi^1(S_0)], \ldots, \mathbb{E}[v_\pi^m(S_0)])$$

For each agent, j, $\mathcal{O}^i$ is analogous to the objective function from definition 1.9. So, intuitively, $\mathcal{O}$ is the vector of expected returns for all m agents, give the policy vector, π.

Typically, we classify multi-agent environments as:

(1) Fully Cooperative:

Definition 7.2. A multi-agent environment $\mathcal{M}$, is called 'Fully Cooperative' if all agents receive the same rewards:

$$\forall r \in \mathcal{R}, \forall j \in \{1, \ldots, m\} \quad r^j = r^1$$

(2) Fully Competitive:

Definition 7.3. A multi-agent environment $\mathcal{M}$, is called 'Fully Competitive' if

$$\forall r \in \mathcal{R} \quad \sum_{j=1}^{m} r^j = 0$$

(3) Mixed:

Definition 7.4. A multi-agent environment $\mathcal{M}$, is called 'Mixed' if it is neither fully cooperative nor fully competitive.

For fully cooperative environments, the objective can be formulated identically to the single case: to find $\pi^* \in \Pi^* = argmax_\pi \mathcal{O}^1(\pi)$. However, for fully competitive and mixed environments (i.e. non-cooperative environments), we face the question: what is the system-wide objective?

To answer this question, it is helpful to look to the field of Game Theory. In 1950, John Nash famously formulated the Nash Equilibrium [3] as a stable system-wide equlibrium. For our purposes, we can view a policy π as a game-playing strategy and therefore, the set of Nash Equlibria is a natural choice for the set of optimal policies, Π^*. A policy is optimal (i.e. a 'Nash

Equilibrium') if it is contained in Π^*. We denote such a policy as π^*.

We formally define the set of Nash Equilibria Π^* as follows:

Definition 7.5 (Nash Equlibria)**.** Given some environment $\mathcal{M}$, Π^* is the set of $\pi_E \in \Pi$ where, for all agents $j \in \{1, \ldots, m\}$:

$$\pi_E^j \in argmax_{\pi^j \in \Pi^j} \mathcal{O}^j(\pi^1, \ldots, \pi^j, \ldots, \pi^m)$$

Intuitively, each agent j has chosen a policy π^j which maximizes its own returns $\mathcal{O}^j$, when fixing all other agents' policies $\pi^{k \neq j}$.

Remark 7.6. Conveniently, in [3], Nash also proved the existence of Nash Equilibria for finite games with mixed strategies. Because we have defined $\mathcal{M}$ to have finite states, actions, and rewards, and because we have defined policies to be *distributions* over actions, this result guarantees that for any $\mathcal{M}$, Π^* is nonempty. This legitimizes our choice of Nash Equilibrium as a generalized non-cooperative multi-agent objective.

8. Minimax Q-Learning Algorithm

The class of multi-agent environments is broad and varied, so we will focus our attention on a subclass of non-cooperative multi-agent environments: fully competitive 2-agent (FC2) environments with sequential policy selection.[17,18] This will help us to improve our algorithm for solving the Tic-Tac-Toe problem from Section 6.

To be clear, a 'fully competitive 2-agent environment' is just a fully competitive environment, $\mathcal{M}$ where $m = 2$. Recall that, in this case, fully competitive means that $\forall r \in \mathcal{R}$, $r^1 = -r^2$. Because v_π is just a sum of rewards, this means that it's always the case that $v_\pi^1 = -v_\pi^2$. So, for simplicity, we will focus on the first component v_π^1 and refer to it as v_π.

Now, I will draw upon ideas from game theory to develop an algorithm, 'Minimax Q-Learning' which is analogous to Q-Learning for FC2 environments. Notably, we will replace the Bellman Operator with a new 'Minimax Operator' and our objective will now be to find a q^* associated with a Nash Equilibrium, π^*. I will not dive into the details in this section as it closely parallels the work in Sections 3, 4, and 5.

8.1. Minimax Operator.

First, we will define the Q function for FC2 environments:

[17] Refer to remark 7.1 for a description of 'sequential policy selection'

[18] This section draws upon [2] and [1].

Definition 8.1 (Q-Function)**.** For an FC2 environment, $\mathcal{M}$ and policy $\pi \in \Pi$, we define the Q function to be: $q_\pi : \mathcal{S} \times \mathcal{A}^1 \times \mathcal{A}^2 \to \mathbb{R}$, to be:

$$q_\pi(s, a^1, a^2) := \mathbb{E}[R_1^1 + \gamma v_\pi(S_1) \mid S_0 = s, A_0^1 = a^1, A_0^2 = a^2]$$

By remark 7.6, we have that the set of Nash equilibria, Π^* is nonempty. Just as in Section 3.4, we denote the Q and V functions associated with a Nash Equilibrium as q^* and v^* respectively. Now, following [2] and [1], FC2 environments with sequential policy selection give us the following theorem about Π^*.

Theorem 8.2 (Minimax Theorem)**.** *Given FC2 $\mathcal{M}$ and Nash Equilibrium π^* it must be the case that $\forall s \in \mathcal{S}$:*

$$v^*(s) = max_{\beta^1} min_{\beta^2} q^*(s, \beta^1, \beta^2) \tag{8.1}$$

Intuitively, agent 1 is maximizing its returns, given that agent 2 is minimizing agent 1's returns. Note that this relies on agent 2 knowing the policy of agent 1, which is the case due to the sequential policy selection.

Now, the optimality condition for q^* follows as a direct result of definition 8.1 and theorem 8.2:

Definition 8.3. (Bellman for Q^*) Given FC2 $\mathcal{M}$ and Nash Equilibrium policy vector π^*, $\forall (s, a^1, a^2) \in \mathcal{S} \times \mathcal{A}^1 \times \mathcal{A}^2$:

$q_\pi^*(s, a^1, a^2) = \mathbb{E}[R_1^1 + \gamma max_{\beta^1} min_{\beta^2} q^*(s, \beta^1, \beta^2) \mid S_0 = s, A_0^1 = a^1, A_0^2 = a^2]$

is a necessary and sufficient condition for q^*.

This allows use to define the Minimax Operator:

Definition 8.4 (Minimax Operator)**.** Given FC2 environment $\mathcal{M}$ the Minimax Operator is a function:

$\mathbb{M}(f)(s, a^1, a^2) := \mathbb{E}[R_1^1 + \gamma max_{\beta^1} min_{\beta^2} f(s, \beta^1, \beta^2) \mid S_0 = s, A_0^1 = a^1, A_0^2 = a^2]$

I will not prove the convergence properties of the Minimax Operator but similar to the Bellman Operator, we have:

Theorem 8.5 (Convergence of $\mathbb{M}$)**.** *For any function $f : \mathcal{S} \times \mathcal{A}^1 \times \mathcal{A}^2 \to \mathbb{R}$, we have:*

$$\lim_{n \to \infty} \mathbb{M}^n(f) = q^*$$

for q^* associated with a Nash equlibrium policy.

Now that we have a method for identifying q^* we will describe a method to derive a Nash Equilibrium $\pi \in \Pi^*$ given q^*.

Definition 8.6. We define policy vector $\hat{\pi} = (\hat{\pi}^1, \hat{\pi}^2)$ to be one such that

$$supp(\hat{\pi}) \in argmax_{\beta^1} min_{\beta^2} q^*(s, \beta^1, \beta^2)$$

Theorem 8.7. *For $\hat{\pi}$ constructed as described, we have that $\hat{\pi}$ is a Nash equalibrium: $\hat{\pi} \in \Pi^*$.*

Minimax Q-Learning will apply the Minimax Operator repeatedly to find q^* and then build $\hat{\pi} \in \Pi^*$.

8.2. **Statement of Algorithm.** First, following Section 4, we will use single samples to estimate $\mathbb{M}(f)(s, a^1, a^2)$:

$$\hat{\mathbb{M}}(f)(s, a^1, a^2) := (R_1^1 + \gamma max_{\beta^1} min_{\beta^2} f(s, \beta^1, \beta^2) \mid S_0 = s, A_0^1 = a^1, A_0^2 = a^2)$$

Finally, we will define two additional features analogous to those outlined in Section 5.1:

$$\hat{\mathbb{M}}_\alpha(f)(s, a^1, a^2) := (1 - \alpha) f(s, a^1, a^2) + \alpha \hat{\mathbb{M}}(f)(s, a^1, a^2)$$

and

$$\pi_s^\epsilon := \begin{cases} Uniform(\mathcal{A}^1 \times \mathcal{A}^2) & \text{with probability } \epsilon \\ Uniform(argmax_{\beta^1} min_{\beta^2} \hat{q}(s, \beta^1, \beta^2)) & \text{with probability } 1 - \epsilon \end{cases}$$

We are ready to describe Minimax Q-Learning:

Algorithm 3 Minimax Q-Learning [1]

Choose $N \in \mathbb{N}$, $\alpha \in (0, 1)$, and $\epsilon \in (0, 1)$
$\mathcal{M} = \big(m = 2, \mathcal{S}, \mathcal{A} = \mathcal{A}^1 \times \mathcal{A}^2, \mathcal{R}, f_0, \tau, \gamma\big)$
Initialize $\hat{q}(s, a^1, a^2) = 0$, $\forall s \in \mathcal{S}, a^1 \in \mathcal{A}^1, a^2 \in \mathcal{A}^2$
for N episodes **do**
  Initialize $t \leftarrow 0$ and sample $s_0 \sim f_0$
  while $s_t \notin \mathcal{S}_\infty$ **do**
    $(a_t^1, a_t^2) \sim \pi_{s_t}^\epsilon$
    Sample $(s_{t+1}, r_{t+1}) \sim \tau_{s_t, (a_t^1, a_t^2)}$
    $\hat{q}(s_t, a_t^1, a_t^2) \leftarrow (1 - \alpha) \hat{q}(s_t, a_t^1, a_t^2) + \alpha [r_{t+1}^1 + \gamma max_{\beta^1} min_{\beta^2} \hat{q}(s, \beta^1, \beta^2)]$

    $t \leftarrow t + 1$
  end while
end for
$(\hat{\pi}_s^1, \hat{\pi}_s^2) \leftarrow (\beta^1, \beta^2) \in argmax_{\beta^1} min_{\beta^2} f(s, \beta^1, \beta^2)$, $\forall s \in \mathcal{S}$
return $\hat{\pi} = (\hat{\pi}^1, \hat{\pi}^2)$

9. Minimax Q-Learning for Tic-Tac-Toe

We will now revisit Tic-Tac-Toe from Section 6 but with the tools to tackle the problem more effectively. We also now have the well-defined system wide

goal of identifying a Nash Equilbrium. So, we will use Minimax Q-Learning to find the Nash Equilibrium for Tic-Tac-Toe.

9.1. **Multi-Agent RL Formulation.** We can model the multi-agent environment as

$$\mathcal{M} = (m, \mathcal{S}, \mathcal{A}, \mathcal{R}, f_0, \tau, \gamma)$$

where:

- m=2
- $\mathcal{S} = \{(b_1, \ldots, b_9) \mid b_i \in \{X, O, '\quad'\}\}$. The terminal states are the ones for which the game has ended, as defined in Section 6.1.
- $\mathcal{A} = \mathcal{A}^1 \times \mathcal{A}^2 = \{1, \ldots, 9\} \times \{1, \ldots, 9\}$[19]
- $\mathcal{R} = \{(-10, 10), (0, 0), (10, 10)\}$
- $$f_0(s) = \begin{cases} 1 & \text{for the empty board} \\ 0 & \text{else} \end{cases}$$
  In words, we always starts with the empty board.
- τ: $\tau_{s,r}$ is now deterministic for any state-action pair and can be described in the following way:
  - If the agents are a terminal state, then they receive a reward of (0,0) and stay in that state regardless of the action taken.
  - If the agent X's move wins the game, then they receive a reward of (10,-10) and go to that terminal state.
  - If either agent's move ties the game, then they receive a reward of (0,0) and go to that terminal state.
  - If agent X's move does not cause a win or tie:
    - If Agent O's move wins the game then they receive a reward of (-10,10) and go to that terminal state.
    - If neither agent's move causes a transition to a terminal state, then they receive a reward of (0,0) and the state changes according to both moves.
- $\gamma = .999$

Also, the policy selection for the environment is specified to be *sequential*.

To solve this problem we will use algorithm 3 and choose the same parameters as in Section 6:

- $\alpha = .1$
- $\epsilon = 1$
- $N = 10,000$

[19]Just as in the single-agent case, the actions are actually restricted to the *open* spots on the board. We will not worry about this however, because in practice we can just remove these actions and everything else will follow identically.

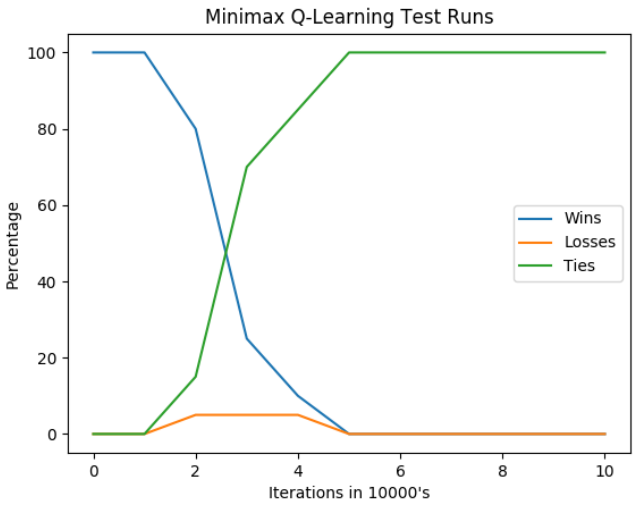


FIGURE 7. Game Results for π^0

We choose the same parameters both because the reasons for the original choices remain and also so that we can accurately compare the results between the algorithms.

9.2. **Results.** While we now know that we are specifically looking for a Nash Equilibrium, it is still difficult to assess if that Nash Equilibrium has been found. So, following the reasoning of remark 6.2, we will use the rate of tying, along with a qualitative assessment of gameplay, as an approximate assessment of whether a Nash Equilibrium has been reached.

Figure 7 plots the (average over 20 runs) percentage of wins, losses and ties for Player X using π^0 after i iterations. Note that this plot shows a much smoother and faster convergence to tying when compared to the naive implementation (Figure 4). Furthermore, if we assess Figure 8 we see, again, that the players seem to be playing optimally.

To better understand how the agents are making decisions, Figure 9 helps to visualize (averaged over 20 runs) $\hat{q}(s, a^1, a^2)$ after 10,000 episodes and is constructed to be analogous to Figure 6.

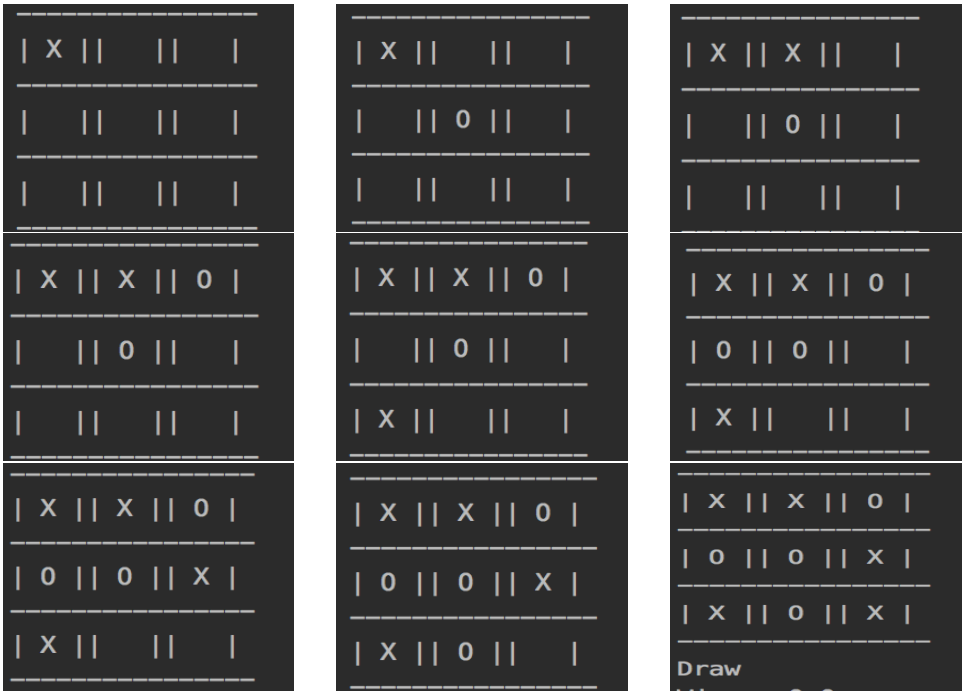
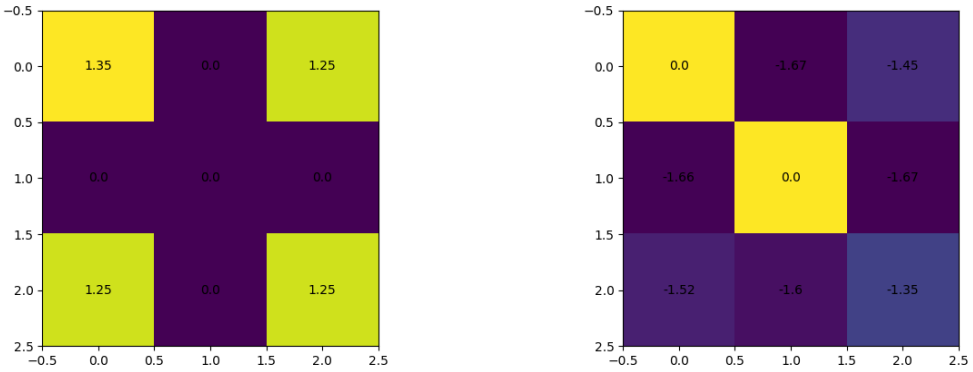


FIGURE 8. Game Played After Full Training

The left heat map is $min_{a^2}\hat{q}(s, a^1, a^2)$ across $a^1 \in \mathcal{A}^1$ at the starting state. We have that a^2 minimizes $\hat{q}$ due to the assumption that agent 2 is maximizing it's own rewards (i.e. minimizing agent 1's rewards) given agent 1's action. So, it represents the expected rewards for agent 1 after the full tuple of actions (a^1, a^2) has been taken. This is analogous to the left heat map in Figure 6. Note, however, that in this setting, the heat map (and therefore $\hat{q}$) accurately captures the inherit symmetries of the game. By this I mean, for example, that the corners, which should be identical from a strategic perspective, all have very similar values. This is an improvement over the analogous heat map from the naive implementation. This indicates that this approach yields a better model of the environment.

Similarly, the right heat map is $-\hat{q}(s_0, 1, a^2)$ across $a^2 \in \mathcal{A}^2$ where the '1' indicates that agent 1 has taken action 1 as its first move. The negative sign is due to agent 2's rewards being the opposite of agent 1's rewards. So, it represents the expected rewards for agent 2 after the full tuple of actions (a^1, a^2) has been taken. This is analogous to the right heat map in Figure 6. Again, note that this heat map (and therefore $\hat{q}$) better captures the inherit symmetries of the game than the naive approach.



$min_{a^2}\hat{q}(s_0, a^1, a^2)$ Across a^1 $\quad$ $-\hat{q}(s_0, 1, a^2)$ Across a^2

FIGURE 9. $\hat{q}(s, a^1, a^2)$ at the Starting State, s_0

It is clear that using Minimax Q-Learning improved our results. A comparison between Figure 7 and Figure 4 shows that π^0 converged to tying strategies (our proxy for finding a Nash Equilibrium) faster and more reliably than in the naive approach. A comparison between Figure 9 and Figure 6 demonstrates that $\hat{q}$ better captures the symmetries and, likely, the overall dynamics of the game better than in the naive approach. This improvement is due to the fact that the multi-agent approach *takes into account the objectives of the other agents* while a naive single-agent approach does not.

FUTURE DIRECTIONS

In this paper, I defined the Nash Equilibrium as a natural system-wide objective for multi-agent RL, and indeed, it is a natural starting point. But there are many other possible considerations. For example, Nash Equilibrium assumes that each player is acting optimally. What if you would like to ensure each agent acts optimally, even if the other agents are not? This task of defining multi-agent objectives is an open problem in the field.

In Section 7, I developed a general multi-agent framework for RL. Initially, I had hoped to maintain such a broad perspective when moving on to algorithms which leverage this multi-agent formulation. Unfortunately, I found that most multi-agent RL algorithms only work for very specific subclasses of environments.[20] An important future direction would be to use general

[20] [1] provides a survey of many such algorithms

multi-agent formulations to develop RL algorithms which are more robust to environment specifications.

Acknowledgments

First, I would like to thank my thesis advisor, Professor Robert Manning, and my brother, Andrew Scharf, for their feedback and notes throughout this process. I would also like to thank my parents, family, and friends for their love and support these last four years. Finally, I would like to thank the wonderful faculty at Haverford College for all I have learned in my time here.

References

[1] R. Babuska L. Busoniu and B. De Schutter. *Multi-agent reinforcement learning: An overview*. 2010.

[2] Michael L. Littman. *Markov games as a framework for multi-agent reinforcement learning*. 1994.

[3] John F. Nash. "Equilibrium points in n-person games". In: *Proceedings of the National Academy of Sciences* 36.1 (1950), pp. 48–49. DOI: 10.1073/pnas.36.1.48. eprint: https://www.pnas.org/content/36/1/48.full.pdf.

[4] S. Paternain and M. Calvo-Fullana. *Penn ESE 680-5 Lecture Notes*. 2019.

[5] A. B. Piunovskiy. *Examples in Markov decision processes*. Imperial College Press, 2013.

[6] S. M. Ross. *Introduction to Stochastic Dynamic Programming*. Academic Press, 1983.

[7] T. Sandholm. *Perspectives on multiagent learning*. 2007.

[8] David Silver et al. "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play". In: *Science* 362.6419 (2018), pp. 1140–1144. DOI: 10.1126/science.aar6404. eprint: https://science.sciencemag.org/content/362/6419/1140.full.pdf.

[9] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. Second. The MIT Press, 2018.

[10] John N. Tsitsiklis. *Asynchronous Stochastic Approximation and Q-Learning*. 1994.

[11] Chao Yu, Jiming Liu, and Shamim Nemati. *Reinforcement Learning in Healthcare: A Survey*. 2019. arXiv: 1908.08796 [cs.LG].