LINEAR FUNCTION APPROXIMATION FOR Q-LEARNING

ANTHONY WONG

1. Introduction. Q-learning is one of the oldest and most well-studied reinforcement learning algorithms. The main idea is to store Q(s,a) $\forall a \in A, s \in S$, which provides an estimate of "value" of a state. The value attempts to quantify how good a state is by approximating the expected rewards the agent will eventually get by choosing action a at state s, and then choosing the optimal policy afterward.

As the agent goes through more and more states and actions, it collects information from the environment in the form of rewards. It then updates Q(s,a) iteratively after each observation as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left(r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right)$$

- $Q(s,a)$: Q-value for state s and action a
- α: Learning rate
- r: Immediate reward received after taking action a in state s
- γ: Discount factor for future rewards
- s': Next state after taking action a
- a': Possible actions in the next state s'

Using a Q-table to store Q(s,a) has three significant drawbacks and limitations, according to [3]:

1. If we do not visit a state s and take an action a, we have no estimate of $Q(s,a)$. This is the case even if we have visited states very similar to s and have taken the action a in other states. Furthermore, good estimates often require many visits to Q(s,a).
2. A table of size $|A| \times |S|$ can become very computationally expensive for many applications.
3. A table can only hold discrete states where $|A|$ and $|S|$ are finite sets. For continuous states and actions represented by real numbers, a table simply cannot store all the values for Q(s,a).

Linear functon approximation, the primary focus of this project, is a solution to these issues in Q-learning. Instead of storing a Q-table for Q(s,a), function approximation uses a parameterized linear function to approximate Q(s,a).

1.1. Project Goals. I first examine the mathematical framework presented by Melo et. al [2], considering the necessity, practicality, and significance of the key assumptions and results. In particular, we analyze the novel applications of function spaces and stochastic estimation to the linear approximation problem. I provide key mathematical statements and an original proof to interpret the necessity of their assumptions

Then, I dive into Carvalho et. al [1] to analyze their new proposed linear function approximation algorithm that wold converge under more general conditions. I question why this new algorithm might not need conditions as strict as the last paper. At the end, I analyze the practicality of both papers, judging whether their assumptions can be safely made in real world settings.

2. Background Mathematical Framework. The problem that reinforcement learning seeks to solve is formalized as a Markov Decision Process (MDP), which includes:

- States: represented by χ, a compact subset of $\mathbb{R}^p$, representing the the set of possible states
- Actions (A): The set of legal actions the agent can take when encountering a state
- Transition Probability (P): The probability of moving from one state to another, given the action that the agent has taken:

$$\mathbb{P}[X_{t+1} \in U \mid X_t = x, A_t = a] = P_a(x, U),$$

  where U is any measurable subset of $\mathcal{X}$.
- Reward Function (R): The immediate reward received after transitioning between states following an action The agent aims to choose the sequence of actions $\{A_t\}$ that maximizes the expected total discounted reward, which exponentially decreases weight for future rewards:

$$V(\{A_t\}, x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = x\right],$$

2.1. Solving MDPs:. To solve an MDP, we introduce value functions and Q-functions, both of which are critical for generating an optimal policy. A value function V represents the expected return an agent can achieve given a state, quantifying the "desirability" of each state. The optimal value function V^* is defined for each state $x \in \mathcal{X}$ as

$$V^*(x) = \max_{\{A_t\}} V(\{A_t\}, x) = \max_{\{A_t\}} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(X_t, A_t) \mid X_0 = x\right]$$

$$= \max_{a \in \mathcal{A}} \int_{\mathcal{X}} \left[r(x, a, y) + \gamma V^*(y)\right] P_a(x, dy),$$

The last recursive expression for the optimal value function is significant because it expresses V^* of the form the well-known Bellman optimality equation.

The optimal Q function Q^* represents the expected value of taking a particular action a in a state x, followed by acting optimally:

$$Q^*(x, a) = \int_{\mathcal{X}} \left[r(x, a, y) + \gamma V^*(y)\right] P_a(x, dy)$$

The key idea for a Q-learning solution to an MDP is that given a the optimal Q-function Q^*, we can define a optimal policy. We define this optimal mapping $\pi^* : \mathcal{X} \to \mathcal{A}$ by

$$\pi^*(x) = \arg\max_{a \in \mathcal{A}} Q^*(x, a),$$

Heuristically, the policy says that given a state, we take the action that maximizes our long term returns (gives us the highest Q value). Mathematically, resulting control process $A_t = \pi^*(X_t)$ is optimal because

$$V(\{A_t\}, x) = V^*(x),$$

Thus, the policy π^* is optimal because the agent "attains the maximum" in the sense that the value of each state ($V^*(x)$) is fully realized. These mathematical definitions are standard for Q-learning, and define exactly what it means to have an "optimal" policy, the "value" of any given state, and the relative "goodness" of each action.

2.2. The Q-learning Algorithm. Given functions $v : \mathcal{X} \to \mathbb{R}$ and $q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$, define the following operators

$$(Tv)(x) = \max_{a \in \mathcal{A}} \int_{\mathcal{X}} [r(x, a, y) + \gamma v(y)] \, P_a(x, dy)$$

and

$$(Hq)(x, a) = \int_{\mathcal{X}} \left[r(x, a, y) + \gamma \max_{u \in \mathcal{A}} q(y, u) \right] P_a(x, dy).$$

The functions V^* and Q^* from above are fixed-points of the Bellman operators T and H. These operators are defined carefully, such that they are contractions in corresponding norms, so a fixed-point iteration would return V^* and Q^*.

However, if the transition probabilities P and the reward function r are not known, we instead approximate Q^* using the Q-learning algorithm, which is defined by the update rule

$$Q_{k+1}(x, a) = (1 - \alpha_k) Q_k(x, a) + \alpha_k \left[R(x, a) + \gamma \max_{u \in \mathcal{A}} Q_k(X(x, a), u) \right] \tag{2.1}$$

where $Q_k(x, a)$ is the kth estimate of $Q^*(x, a)$. Here, $R(x, a)$ and $X(x, a)$ are observations or simulations rather than the true functions. With minor constraints on the learning rates α_t, estimates of Q_k converge with high probability, as found by Watkins and Dayan [4].

These definitions give the mathematical structure and approach for Q-learning. From here, the two papers differ. We first examine Melo et. al, which establishes convergence and error guarantees of linear function approximation in the presence of conditions on the underlying markov chain, the exploration process, and algorithmic step sizes. Carvalho et. al propose a new linear function approximation algorithm that converges under more general conditions, and deals with more modern environments like bootstrapping and sample replay.

3. Novel Framework for Linear Function Approximation (Melo et al.). In order to tackle the problem of Q-function estimation in infinite-dimensions as described in the background section, Melo et al. introduces linear function spaces. Denote a linear family of functions Q_θ defined by the linearly independent functions $\xi_i : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. Despite the fact that we are working with functions and states rather, the approach is heavily motivated by linearly regression. ξ_i can be interpreted as the "features" extracted from the state data, and θ are the linear coefficients as in regression.

The linear function space Q_θ is therefore the linear span of the feature functions, and any $Q_\theta(x, a) \in Q_\theta$ can be represented as

$$Q_\theta(x, a) = \sum_{i=1}^{M} \xi_i(x, a) \theta(i) = \xi^\top(x, a) \theta,$$

where $\xi(x, a)$ is a vector in $\mathbb{R}^M$ with ith component given by $\xi_i(x, a)$, and where $\theta(i)$ and $\theta \in \mathbb{R}^M$. We let Ξ represent the chosen feature functions, $\Xi = \{\xi_i, i = 1, \ldots, M\}$. Here, the paper introduces key conditions and a new projection operator that are essential to setting up their algorithm, but glosses over the motivation and implications of such conditions. I will first present and define these mathematical statements, then I will explain their significance and give an interpretable analysis on

them (which is notably missing from Melo et al). Condition 3.2 on the set of feature vectors Ξ states that for all i, ξ_i must be bounded functions satisfying

$$\sum_i |\xi_i(x,a)| \leq 1$$

$\forall (x,a) \in \mathcal{X} \times \mathcal{A}$. Furthermore, the paper also imposes Condition 3.3, that the Ξ must verify

$$\|\xi_i\|_\infty = 1$$

This condition is quite strict, but quite important for the projection definitions, as I will explain. Furthermore, Condition 3.3 also provides stability of the iterative stochastic approximation algorithm, which helps allow for convergence guarantees. Melo et al. remarks that if Conditions 3.2 and 3.3 both hold true, then Ξ are linear independent. Here are some important observations that prove this fact.

First, notice that for each ξ_i there is a point (x_i, a_i) such that $|\xi_i(x,a)| = 1$. This follows by definition of the infinity norm being 1 in Condition 3.3. By condition 3.2, we have that for the pair (x_i, a_i), we get $\xi_j(x_i, a_i) = 0$ for all $j \neq i$. Thus, the functions are linearly independent. These points (x_i, a_i) turn out to be extremely important in the definition of the projection operator.

3.1. Formulation: Projection operator. The projection operator is defined in Melo et al. as follows. For each function $\xi_i \in \Xi$, we take the point (x_i, a_i) such that $|\xi_i(x_i, a_i)| = 1$. We let I denote the set obtained of all M such points, one for each $\xi_i \in \Xi$. Let B an infinite dimensional function space, where for $\in B$, $f : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$. B represents the set of all possible Q functions, in which we want to find the optimal Q function. Now, the projection operator is defined as $\varphi : B \to \mathbb{R}^M$

$$(\varphi f)(i) = f(x_i, a_i)$$

where $f \in B$ and $(\varphi f)_i$ is the ith component of the vector φf.

3.2. Results: Projection Operator Interpretation. In this section, I present the purpose of the projection operator, and explain why we need the mathematical conditions I've thus far presented. Melo et al. simply defines the operator using these conditions without explanation. I will provide the interpretation for these conditions and the projection operator, along with an original proof for my mathematical claims.

Finding an optimal Q-function in the infinite dimensional function space B is intractable. The purpose of the projection operator φ is that to project each $f \in B$ down to a finite dimensional subspace. In particular, for linear function approximation, we want to project the function down to the feature space, i.e. the space spanned by the feature vectors ξ_i. The projection φ maps each $f \in B$ to a an M dimensional vector, representing the function projected down to the finite dimensional subspace. This idea is motivated by linear regression, where the true response vector Y is projected down onto the feature space of X to produce $\hat{Y}$.

Consider the set $\Xi = \{\xi_1, \ldots, \xi_M\}$. This set Ξ naturally spans an M-dimensional subspace $V \subseteq B$:

$$V = \{f_\theta : f_\theta(x,a) = \sum_{i=1}^{M} \theta_i \xi_i(x,a) \mid \theta \in \mathbb{R}^M\}.$$

THEOREM 3.1. *φ provides a linear isomorphism between V and $\mathbb{R}^M$ such that the combination of the chosen functions $\{\xi_i\}$ and evaluation points (x_i, a_i) provides a concrete coordinate system (a basis) for representing functions in V. Further, applying φ to an arbitrary $f \in B$ projects the function down to the subspace V.*

Proof:. First, consider the action of φ on a function $f_\theta \in V$. By definition of φ, we get:

$$(\varphi f_\theta) = \big(f_\theta(x_1, a_1), \ldots, f_\theta(x_M, a_M)\big).$$

Substituting $f_\theta(x, a) = \sum_{j=1}^{M} \theta_j \xi_j(x, a)$, we have:

$$(\varphi f_\theta)(i) = \sum_{j=1}^{M} \theta_j \xi_j(x_i, a_i).$$

This can be viewed in matrix form:

$$\varphi f_\theta = A\theta,$$

where A is the $M \times M$ matrix whose entries are $A_{ij} = \xi_j(x_i, a_i)$. Notice that if the chosen points (x_i, a_i) and the functions ξ_i are selected so that the resulting matrix A is nonsingular (invertible), then:

- Every function f_θ in the subspace V can be uniquely identified by the vector of its values at these chosen points (x_i, a_i).
- Conversely, given the vector of evaluations φf_θ, you can solve for θ via $\theta = A^{-1}(\varphi f_\theta)$.

Thus, it suffices to show A is invertible to show that that φ is a linear isomorphism.

This is where conditions 3.2 and 3.3 come in. These conditions are necessary for φ to be a linear isomorphism. First, apply condition (3.2) to the point (x_i, a_i). Since $|\xi_i(x_i, a_i)| = 1$, if there were any $j \neq i$ for which $|\xi_j(x_i, a_i)| > 0$, then we would have$1 = |\xi_i(x_i, a_i)| + \sum_{j \neq i} |\xi_j(x_i, a_i)| > 1$, which would contradict (3.2). Thus, the only way to satisfy the inequality is:

$$|\xi_i(x_i, a_i)| = 1 \quad \text{and} \quad |\xi_j(x_i, a_i)| = 0$$

for all $j \neq i$.

Therefore, at the chosen point (x_i, a_i), the vector $(\xi_1(x_i, a_i), \ldots, \xi_M(x_i, a_i))$ has a 1 in the i-th position (up to a possible sign if $\xi_i(x_i, a_i) = -1$, which can be handled by considering $-\xi_i$ if necessary) and zeros in all other positions. By construction, the matrix $\mathbf{A}$ whose entries are $A_{ij} = \xi_j(x_i, a_i)$ is such that the i-th row of $\mathbf{A}$ is all zeros except for a single nonzero entry at column i, which equals $\xi_i(x_i, a_i)$ with magnitude 1. In other words, $\mathbf{A}$ is a matrix whose rows form a permutation of the rows of a diagonal matrix with nonzero diagonal entries. More explicitly, after reordering indices or adjusting signs, $\mathbf{A}$ looks like:

$$A = \begin{pmatrix} \xi_1(x_1, a_1) & 0 & \cdots & 0 \\ 0 & \xi_2(x_2, a_2) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \xi_M(x_M, a_M) \end{pmatrix}$$

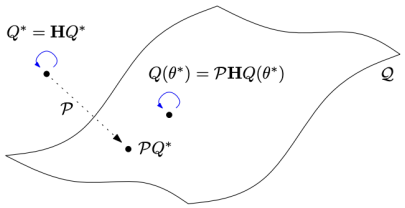

FIG. 1. *Melo et al. projection algorithm*

with each $|\xi_i(x_i, a_i)| = 1$. Since each $\xi(x_i, a_i) \neq 0$, this matrix A becomes an upper triangular matrix with nonzero diagonal entries.

$$\det(A) = \prod_{i=1}^{M} \xi(x_i, a_i) \neq 0.$$

Therefore, A is invertible.

Now, consider the action of φ to an arbitrary $f \in B$. We have already shown that the vectors produced φ work as a basis coordinate system of V. We know outline exactly how φ projects f into the finite dimensional basis of V. From above we know that for some $v \in V$ given the vector φv, we can recover $\boldsymbol{\theta}$ uniquely by:

$$\boldsymbol{\theta} = A^{-1}(\varphi v).$$

To project a general function $f \in \mathcal{B}$ onto V, we proceed as follows:

1. First, evaluate f at the chosen sample points to get φf.
2. Solve $\boldsymbol{\theta} = A^{-1}(\varphi f)$ to find the coefficients $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_M)$.
3. Form the projection $\mathcal{P}(f)$ by equation (3.5) below:

$$\mathcal{P}(f)(x, a) = \sum_{j=1}^{M} \theta_j \xi_j(x, a)$$

In other words, φ extracts the "sampled coordinates" of f at the chosen points. Then, using the known structure of $\{\xi_i\}$ and the invertible matrix A, we map these coordinates back into the subspace spanned by the $\{\xi_i\}$. The combined operation—apply φ, then invert A, and reconstruct from $\{\xi_i\}$—constitutes the projection onto V. This concludes the proof.

The theorem above mathematically explains the necessity for conditions 3.2 and 3.3. I've provided this interpretation and mathematical elaboration because it is missing from Melo et al, which make these conditions seem arbitrary. But in fact, they are necessary for creating a projection operator that is a linear isomorphism. This is significant because is provides a finite dimensional basis and coordinate system to work with, which makes the problem tractable for the convergence algorithm.

3.3. Formulation: Convergence Algorithm. We now introduce the main algorithm of Melo et. al. The idea is to carry out the fixed point iteration of the Bellman operator exclusively on the feature space by using the projection operator, see Fig 1. Recall that the original derivation uses fixed point iteration in the infinite dimensional space, as mentioned in section 2.2.

The resulting $\theta^* \in \mathbb{R}^M$ satisfies the fixed-point recursion

$$\theta^* = \varphi \mathcal{H} Q(\theta^*) \tag{3.1}$$

where $\mathcal{H}$ is the operator defined in section 2.2.

Here, I present another interpretation for why Condition 3.2 is necessary. In order for the above fixed point to be properly defined and unique, we need $\varphi\mathcal{H}$ to be a contraction map. $\mathcal{H}$ is already a contraction in the maximum norm. Since Condition 3.2 says $\sum_i |\xi_i(x,a)| \leq 1$, we get that $||\varphi f||_\infty \leq ||f||_\infty$, and thus φ does not amplify the norm of any f, and $\varphi\mathcal{H}$ is a contraction norm as desired.

Now, we re-express 3.1 in terms of the operators $\mathcal{H}$ and $\mathcal{P}$ to get

$$\theta^*(i) = \int_{\mathcal{X}} \delta_{(x_i,a_i)}(x,a) \int_{\mathcal{X}} \left[r(x,a,y) + \gamma \max_u \xi^\top(y,u)\theta^* \right] P_a(x,dy)\mu(x,a) \tag{3.2}$$

where μ is the probability measure on $\mathcal{X} \times \mathcal{A}$ and $\delta_{(x_i,a_i)}$ is the Dirac delta.

Suppose that $\{x_t\}$, $\{a_t\}$, and $\{r_t\}$ are sampled trajectories from the MDP $(\mathcal{X}, \mathcal{A}, P, r, \gamma)$ using policy π. Then, given any initial estimate θ_0, we generate an iterative sequence $\{\theta_t\}$ using

$$\theta_{t+1}(i) = \theta_t(i) + \alpha_t g_\varepsilon(x_i, a_i; x_t, a_t) \left[r_t + \gamma \max_{u \in \mathcal{A}} \xi^\top(x_{t+1}, u)\theta_t - \xi^\top(x_t, a_t)\theta_t \right],$$

where $\{\varepsilon_t\}$ is a sequence such that $\varepsilon_{t+1} = (1-\beta_t)\varepsilon_t$. g_ε be a smooth Dirac approximation, with conditions such that g_ε approaches the dirac delta as $\epsilon \to 0$.

We can also express this algorithm as

$$\varepsilon_{t+1} = \varepsilon_t + \beta_t h(\varepsilon_t),$$

where h is chosen so that the ODE $\dot{x}_t = h(x_t)$ has the right globally stable equilibrium. The resulting optimal function Q_{θ^*} satisfies the fixed point iteration

$$Q_{\theta^*}(x,a) = (\mathcal{P}HQ_{\theta^*})(x,a)$$

where $\mathcal{P} : \mathcal{B} \to \mathcal{Q}$ denotes the operator given by

$$(\mathcal{P}Q)(x,a) = \xi^\top(x,a) \odot Q.$$

3.4. Results: Interpreting the algorithm. The Dirac delta is used to express the locality in the original Q-update equation 2.1. The idea is that after every state action pair, we only want to update the Q-function at that state action pair. The Dirac delta is an appropriate tool for isolating one point, and assigning no weight to other points when updating. However, for a continuous state space, we cannot strictly update one point. Instead, we update a small ε neighborhood around the state action pair, which is expressed in the smooth approximation g_ε.

ε_t and β_t represent step sizes for the algorithm. Notice that if ε is fixed, the algorithm will converges to a neighborhood of the optimal function. Once the estimates reach this neighborhood, we decrease ε to reach a smaller neighborhood. In this manner the algorithm repeatedly and slowly decreases ε.

The projection $\mathcal{P}$ follows from the interpretation of φ I provided earlier, and projects arbitrary functions down to feature space as seen in Figure 1.

3.5. Results: Proof of Convergence. Under the assumption that the underlying Markov chain $(\mathcal{X}, P_\pi)$ is ergodic and that the step-sizes α_t and β_t meet certain conditions, this algorithm converges. The idea is to apply Hirsh's lemma to show that the stochastic trajectory of the algorithm closely follow those of an associated ODE which has equilibrium at θ^*. Thus, θ_t converges w.h.p to θ^*. The detailed proof can be found in the appendix of Melo et al. I provide a more intuitive proof sketch, which shows hueristically why these assumptions are required.

We define the the sampled temporal-difference error to be:

$$\Delta_t = R_t + \gamma \max_a Q_\theta(X_{t+1}, a) - Q_\theta(X_t, A_t).$$

Under geometric ergodicity, we have that the every region of the state-action space is visited infinitely often. Mathematically, this ensures that:

$$\mathbb{E}[\Delta_t \mid X_t, A_t] \to \mathbb{E}[R_t + \gamma \max_a Q_\theta(X_{t+1}, a) - Q_\theta(X_t, A_t)],$$

This implies that the expected value of the temporal difference Δ_t, conditioned on the current state-action pair, converges to the expected Bellman error. The update rule now becomes

$$\theta_{t+1} = \theta_t + \alpha_t \left[\mathbb{E}[R_t + \gamma \max_a Q_\theta(X_{t+1}, a) - Q_\theta(X_t, A_t)] + \eta_t\right],$$

where η_t captures the stochastic noise (the deviation of the update from the expectation). As $t \to \infty$, the step size $\alpha_t \to 0$, and the contribution of η_t diminishes. Thus, intuitively, these conditions reduce the stochastic updates to a deterministic approximation, an ODE where the update is always the expectation.

$$\theta_{t+1} \approx \theta_t + \alpha_t \left[\mathbb{E}[H(\theta)] - \theta\right],$$

where $H(\theta)$ is the Bellman operator. We can now derive fixed point corresponding to the equilibrium solution θ^*, for the above ODE:

$$\theta_{t+1} \approx \theta_t + \alpha_t \mathbb{E}[H(\theta_t) - \theta_t]$$

Thus, we obtain $\theta^* = \mathbb{E}[H(\theta^*)]$. Mathematically, the intuition is that the stochastic updates are unbiased in the limit, with noise averaging out due to ergodicity and diminishing step sizes.

4. New Proposed Algorithm in Carvalho et al.. Carvalho et al. proposes a convergent Q-learning linear function approximation algorithm in a setting that allows replay buffers and non ergodicity. At a high level, this algorithm operates in a much more general setting that introduces bootstrapping and off-policy learning. In off-policy algorithms, the learning policy (which updates the Q function) is decoupled from the behavior policy (which generates the data). Bootstrapping algorithms model future estimates of the Q-function using previous estimates. In the past, off-policy and bootstrapping algorithms have been shown to diverge, since Q-value over-estimation errors would propagate repeatedly across iterations. Modern techniques allow the algorithm to still converge with non-ideal features, whereas the Melo et al. requires a near-perfect set of features. By now, we have done most of the grunt work for the mathematical framework, as Carvalho et al. have a very similar set of assumptions as we have already outlined.

4.1. Formulation: New Assumptions and Convergence Algorithm. Replay buffers store previous pairs of state, action, reward and next pair (x_t, a_t, r_t, x'_t) so that they can be sampled later for function approximation. This experience replay technique relaxes assumptions about the sampled data, making the approach applicable to scenarios with limited exploration or non-ergodic environments. The tuples are independently sampled from a replay buffer B, following a fixed distribution μ. This distribution ensures in the tuple used for the next update, x'_t is generated by the MDP's transition probability $P(\cdot|x_t, a_t)$, and the reward satisfies $\mathbb{E}_\mu[r_t|x_t, a_t] = R(x_t, a_t)$. The features must still be linearly indpendent, and scaled such that $|\xi_i(x, a)|_\infty \leq 1$. However, the very strict condition that $\sum_i |\xi_i(x, a)| \leq 1 \forall (x, a) \in \mathcal{X} \times \mathcal{A}$ is dropped.

The detailed algorithm and proof of convergence can be found in the paper. The main ideas are as follows:

The algorithm approximates the Q-function using two sets of parameters:

- Main network (Q_v): Represents the "current" Q-function, updated on a faster timescale.
- Target network (Q_u): Serves as a reference or stabilizer, updated on a slower timescale.

Both networks approximate the Q-value as:

$$Q_v(x, a) = \xi(x, a)^\top v \quad \text{and} \quad Q_u(x, a) = \xi(x, a)^\top u$$

Q_v updates using a temporal difference (TD) rule similar to traditional Q-learning:

$$v_{t+1} = v_t + \beta_t \xi(x_t, a_t)\delta_t \tag{4.1}$$

The target network updates more slowly, ensuring stability:

$$u_{t+1} = u_t + \alpha_t \left(\xi(x_t, a_t)Q_v(x_t, a_t) - u_t\right) \tag{4.2}$$

Critically, we set the step sizes $\alpha_t \ll \beta_t$. This way, target network updates slowly relative to the main network, mimicking the behavior of the target network in Deep Q-network learning.

4.2. Results: Analysis/Interpretation of the New Conditions. Replay buffers are a less restrictive than forcing a sampling method that creates ergodicity as Melo et al. had done. Critically, experience replay allows for independent sampling of data points. This helps avoid the highly correlated updates that can occur when training directly from sequential Markov chain samples (since states are often heavily correlated). We now interpret the update functions for each network. For the main network, we have that the TD error, measuring the difference between the target value (using the target network, Q_u) and the current estimate from Q_v.

$$\delta_t = r_t + \gamma \max_{a'} Q_u(x'_t, a') - Q_v(x_t, a_t)$$

The idea is that in the update 4.1, if the Q function is far from the target value, we update it less. But what is the interpretation of the target network update? Q_v actually serves as a stabilized version of the Q-function, so that the current estimates do not stray too far away.

The target network aligns itself with the projection of the main network's outputs, ensuring consistency between the two networks. The target network's update 4.2

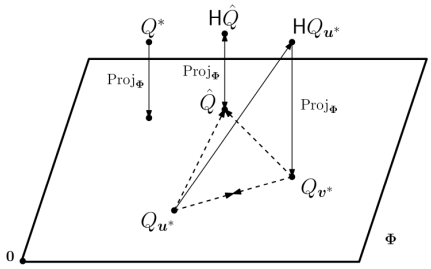

FIG. 2. *Relationship of Q Networks and Projections*

can be intrepreted as a projection. Rewriting $Q_v(x,a) = (x,a)^\top v$. Then, this gets projected to:

$$\xi(x_t,a_t)Q_v(x_t,a_t) = \xi(x_t,a_t)\left(\xi(x_t,a_t)^\top v_t\right)$$

This projection serves to let (u_t) remain in the subspace defined by Ξ, which prevents arbitrary or unbounded updates away from the feature space (and towards Q^*) which would otherwise lead to divergence.

The relationship of the two networks can be seen in Figure 2. The target network can be interpreted as a slowly moving target, which maybe ahead or behind the main network. The main network rapidly updates towards Q^*, but is balanced with the target network. This balance is key because it allows the algorithm to go fast enough to converge, but not fast enough that over-estimation errors will propogate. The details for proving convergence mostly draw upon theory for two time-scale stochastic approximation, and again uses an ODE. In particular, linear independence and Condition 3.3 are still needed to ensure the projection is a contraction, and to provide lipschitz continuity to the mean field functions.

So how is this algorithm able to converge in a more general setting? To answer this question, we interpret the impact of several key algorithmic steps. First, the use of independent replay sampling means that we no longer need ergodicity to converge. Mathematically, ergodicity reduces of the temporal correlations of the states as the Markov chain converges to its stationary distribution. Convergence in stochastic approximation requires that the noise terms are centered (zero-mean) and have bounded variance. Both ergodicity and independence guarantee these conditions:

- Ergodicity relies on the Markov chain's mixing properties to ensure noise centering over time.
- Independence guarantees this directly since samples are already i.i.d. from μ:

$$\mathbb{E}[m_t \mid \mathcal{F}_{t-1}] = 0 \quad \text{and} \quad \mathbb{E}[m_t^2] < \infty.$$

Similarly, we no longer need the strict Condition 3.2, which had previously ensured that the features were small so that Q-value over-estimations would be held in check. In this new algorithm, the main network is allowed to handle features and overrestimate values, but will be held in check by the target network so that errors do not increase exponentially.

4.3. Conclusion/Results: Practicality. One major practical issue is that both convergence algorithms can only find the optimal solution on given a set of features Ξ. If the chosen basis poorly represents the true Q-function, the approximation error could render the solution ineffective. Not only do the two papers give

no insight into feature selection, but they also impose significant restrictions on the choice of features. However, as we've demonstrated, much of these conditions were necessary to get key mathematical properties like projection, contraction, and stability. For example, to make the analysis feasible, it was necessary to simplify the problem to one subspace, rather than a space of all possible features.

Specifically, consider Conditions 3.2 and 3.3. Condition 3.3 can be achieved simply by scaling down each feature by its suprenum, but Condition 3.2 is quite strict. Given an arbitrary set of features, there is no clear way to satisfy Condition 3.2. Instead, Condition 3.2 could likely only be met using extremely contrived features or settings; it would be extremely hard to construct such features in practice.

Furthermore, while these issues provide asymptotic guarantees, this does not directly translate to finite-time performance, which is critical in practical applications. The algorithms also require strict conditions for when they can applied, like ergodicity and full state-action visitation. While key assumptions for these frameworks were mathematically necessary, they were often very theoretical, making it hard to satisfy (or even to check whether these conditions are met) in practice.

REFERENCES

[1] D. S. Carvalho, F. S. Melo, and P. A. Santos, *A new convergent variant of Q-learning with linear function approximation*, in Proceedings of INESC-ID & Instituto Superior Técnico, Lisbon, Portugal, 2020. {diogo.s.carvalho, pedro.santos}@tecnico.ulisboa.pt, fmelo@inesc-id.pt.

[2] F. S. Melo and M. I. Ribeiro, *Q-learning with linear function approximation*, Tech. Report RT-602-07, Instituto de Sistemas e Robótica, Pólo de Lisboa, March 2007.

[3] T. Miller, *About the author and acknowledgements*. Online Notes, 2023. Maintained by the University of Queensland, Brisbane/Meanjin, Australia. https://www.uq.edu.au.

[4] C. J. Watkins and P. Dayan, *Q-learning*, Machine Learning, 8 (1992), pp. 279–292, https://doi.org/10.1007/BF00992698.