ECSE506 Term Project

Convergence of Stochastic Iterative Dynamic Programming Algorithms

Author

Imene ROMDHANE

Mcgill University

Electrical and Computer Engineering Department

May 1, 2020

Contents

Chapter 1

Introduction

This report is an overview of the work done on the convergence of stochastic iterative dynamic programming algorithms, mainly the Q-learning and temporal difference TD(λ) algorithms [1]. The Q-Learning is a reinforcement learning algorithm that learns an action-value function that gives the expected utility of taking a given action in a given state and following a fixed policy thereafter. Whereas the TD(λ) is a reinforcement learning algorithm that learns the value function through updating the estimate by only considering n steps into the future. In this chapter, we will start by briefly introducing reinforcement learning. Then, we present the contribution of the work presented in [1].

1.1 Reinforcement learning

Reinforcement learning is a machine learning approach. It is based on the existence of an agent that learns from its environment and takes actions with the aim of maximizing a total reward. The learning process of RL is mainly based on trial-and-error and is not based on any collected data. In many problems where the environment is dynamic and cannot be fully controlled, traditional machine learning techniques can not be applied, since solving such problems cannot be done using training methods based a priorly collected data. Thus, RL proves to be a promising candidate to solve such problems. RL has been used in many fields, such as robotics, wireless communications, gaming and business strategy planning. The RL system consists of four main elements: The policy, reward, value function, and model of the environment [2].

Policy π: The policy defines how the agent will act at any given time. The aim of the reinforcement learning algorithm is to find the optimal policy that maximizes the cumulative payoff.

Reward r: The reward function maps the couple $(state, action)$ to a value, called reward. It reflects the performance of the agent. The bigger the reward, the better the agent is performing. In a communication system, it is possible to calculate the reward by combining the throughput of a link, spectral efficiency, and power consumption.

Value function V: Unlike the reward that gives an immediate evaluation of the taken action, the value function gives a long-term reward. A current state can give a high immediate reward, but lead to a relatively low value if the next steps give low rewards. Thus, it can be wiser sometimes to opt for a state with a lower reward if it guarantees a higher value.

Model: In some reinforcement learning problems, the environment can be approximated to a model. This model can be used to either predict the reward or find the possible actions at a certain state. In a given RL problem, if policies are chosen according to a given model, it is called a "Model-based method". Otherwise, the policy is learned gradually through trial and error, which is known as "Model-free method".

If the transition probabilities and the immediate costs are known, then the problem can be solved directly using Dynamic Programming algorithms, such as Value iteration and Policy iteration. Otherwise, if the underlying system is not completely known, model-free methods such as Q-learning and $TD(\lambda)$ are needed. There are many cases in real life situations where it is impossible to define a model for the environment. For example, model-free RL algorithms became very popular in solving games and showed to have better performances than human beings. Moreover, this class of RL algorithms has been used in many wireless communication scenarios such as the multiple access wireless communications where the users' access to the system is randomly distributed. This hightlights the importance of model-free RL algorithms, such as Q-learning and $TD(\lambda)$.

1.2 Work contribution

In order to fully understand the Q-learning and the $TD(\lambda)$ algorithms, it is crucial to characterize their convergence. In other words, under what conditions they converge to correct predictions or optimal control policies. The stochastic nature of these algorithms immediately suggests the use of stochastic approximation theory to obtain the convergence results. But no approximation technique is available for problems involving the maximum norm, which is an essential element in the DP based learning algorithms.

The work in [1] extended the classical stochastic approximation theory formulation to obtain a class of converging processes involving the maximum norm. Then, it showed that Q-learning and both the on-line and batch versions of $TD(\lambda)$ are realizations of this class, and hence converge. The advantages of this approach is that it keeps the proofs of convergence simple and do no rely on specific constructions to particular algorithms.

Chapter 2

Q-learning

2.1 Formulation of Q-learning algorithm

Q-learning is an off-policy TD control algorithm. Here, we do not only consider the adopted policy, but also the alternative policies along the way. In other words, a value is assigned to each state-action pair instead of assigning one value to each state. The action-value function Q(s,a) is then updated using the maximum action-value function over all possible actions taken at state s. This can be expressed as,

$$Q(s,u) = \bar{c}_u + \gamma \sum_{s'} p_{ss'}(u) \max_{u'} Q(s', u') \tag{2.1}$$

where $\bar{c}_u$ is the average cost and $p_{ss'}(u)$ is the transition probability from state s to state s' by executing the action u. This equation can be solved iteratively by updating the Q-value at each step. This approach is called the value iteration method.

The values of $\bar{c}$ and $p_{ss'}$ are unknown but the observable quantity

$$\bar{c}_{s_t}(u_t) + \gamma \max_{u} Q(s_{t+1}, u) \tag{2.2}$$

is an unbiased estimate of the update used in value iteration. s_t and u_t are the state of the system and the action taken at time t respectively. The Q-Iearning algorithm is a relaxation method that is based on an iterative use of this estimate to update the current Q-values.

The convergence of Q-learning algorithms is due mainly to the contraction property of the Bellmann operator.

2.2 Convergence of Q-learning

The Q-learning algorithm is a stochastic process, so the techniques of stochastic approximation are usually applicable. Yet, the traditional formulation of stochastic approximation does not include the maximum norm existing in the update equation of Q-learning algorithm. Thus, this formulation needs to be extended to prove the convergence of Q-learning. This can be accomplished via the following theorem.

Theorem 2.1: A random iterative process

$$\Delta_{n+1}(x) = (1 - \alpha_n(x))\Delta_n(x) + \beta_n(x)F_n(x) \tag{2.3}$$

converges to zero w.p.l under the following assumptions:

1. The state space is finite
2. $\sum_n \alpha_n(x) = \infty$, $\sum_n \alpha_n^2(x) < \infty$, $\sum_n \beta_n(x) = \infty$, $\sum_n \beta_n^2(x) < \infty$, and $E[\beta_n(x)|P_n] \leq E[\alpha_n(x)|P_n]$ uniformly w.p.1.
3. $||E[F_n(x)|P_n]||_w \leq \gamma||\Delta_n||_w$, where $\gamma \in (0, 1)$
4. $Var[F_n(x)|P_n] \leq C(1 + ||\Delta_n||_w)^2$, where C is some constant.

where $P_n = [\Delta_n, \Delta_{n-1}, ..., F_{n-1}, ..., \alpha_{n-1}, ..., \beta_{n-1}, ...]$ is the past at step n, $F_n(x), \alpha_n(x)$, and $\beta_n(x)$ may depend on the past and the norm $||.||_w$ denotes some weighted maximum norm.

The term Δ_n usually represents the difference between a stochastic process of interest and some optimal value. Thus, the formulation of the theorem requires some knowledge about the optimal solution to the learning problem before verifying the convergence of the algorithm. For the Q-learning algorithm, the required knowledge is available in the Bellman's equation. This gets us to the second theorem that we deduce from theorem 1.

Theorem 2.2: The Q-learning algorithm given by

$$Q_{t+1}(s_t, u_t) = (1 - \alpha_t(s_t, u_t))Q_t(s_t, u_t) + \alpha_t(s_t, u_t)[c_{s_t}(u_t) + \gamma V_t(s_{t+1})] \quad (2.4)$$

converges to the optimal $Q^*(s, u)$ values if:

1. The state and action spaces are finite.
2. $\sum_t \alpha_t(s_t, u_t) = \infty$, $\sum_n \alpha_t^2(s_t, u_t) < \infty$ uniformly w.p.1.
3. $Var[c_s(u)]$ is bounded.
4. if $\gamma = 1$, all policies lead to a cost free terminal state w.p.1.

Proof: Starting from the equation (2.3) in theorem 1, we define
$\Delta_t(s, u) = Q_t(s, u) - Q_t^*(s, u)$
$F_t(s, u) = c_s(u) + \gamma V_t(s_{next}) - Q^*(s, u)$
$\beta_t(s, u) = \alpha_t(s, u)$.

This brings the learning rule of the Q-learning algorithm defined for theorem 2. We are left with verifying that he defined $F_t(s, u)$ has the required properties.We first start by showing that it is a contraction mapping with respect to a maximum norm.

$$\begin{aligned} \max_u |E[F_t(i, u)]| &= \gamma \max_u |\sum_j p_{ij}(u)[V_t(j) - V^*(j)]| \\ &\leq \gamma \max_u \sum_j p_{ij}(u) \max_v |Q_t(j, v) - Q^*(j, v)| \qquad (2.5) \\ &\leq \gamma \max_j \max_v |Q_t(j, v) - Q^*(j, v)| \end{aligned}$$

When $\gamma < 1$, the contraction property is verified. If $\gamma = 1$ but the chain is absorbing and all policies lead to the terminal state w.p.1, the property still holds because there exists still a weighted maximum norm with respect to which it is a contraction mapping. The variance of $F_t(s, u)$ given the past is within the bounds of Theorem 1 as it depends on $Q_t(s, u)$ at most linearly and the variance of $c_s(u)$ is bounded.

Chapter 3

Temporal Difference Learning

3.1 Formulation of $TD(\lambda)$ algorithm

Temporal difference (TD) learning is a model-free reinforcement learning algorithm. While Q-learning algorithms include decision making tasks, TD learning focuses more on predicting future costs of an evolving system. It is based on what is called bootstrapping techniques. In other words, instead of waiting until the end of the episode, a TD learning algorithm updates the estimate in a faster way by considering only n steps into the future. The definition of a TD algorithm is joined with a parameter λ ranging between 0 and 1. The parameter lambda quantifies how much the future affects the reward function. If $\lambda = 0$, then the future states have no effect is defining the current reward, and if $\lambda = 1$, then the future states effects strongly the current reward. We start by the simplest one, TD(0), where the algorithm looks one step only into the future. The one step look-ahead is given as

$$V_t^{(1)}(i) = c_{t+1} + \gamma V(i_{t+1}) \tag{3.1}$$

The update is thus,

$$V_{t+1}(i_t) = V_t(i_t) + \alpha_t[V_t^{(1)}(i) - V(i_t)] \tag{3.2}$$

For non zero λ, the algorithm combines all n-step look-head terms V_t^ns weighted by λ as follows,

$$V_t^{\lambda}(i) = (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} V_t^{(n)}(i) \tag{3.3}$$

where $V_t^{(n)}(i)$ is the n step look-ahead prediction. The TD(λ) update can then be described by the following equation,

$$V_{t+1}(i_t) = V_t(i_t) + \alpha_t[V_t^{\lambda}(i_t) - V(i_t)] \tag{3.4}$$

The convergence of the algorithm is mainly due to the statistical properties of the $V_t(i)^{\lambda}$ estimates.

3.2 Convergence of $TD(\lambda)$

Here, the convergence of the algorithm can be proven using the theorem 2.1 defined in the previous chapter. The parameters $\alpha_n(i)$ can be replaced by the learning rate parameters α_n since they satisfy the conditions $\sum_n \alpha_n(i) = \infty$ and $\sum_n \alpha_n^2(i) < \infty$ uniformly w.p.1.

Theorem 3.1: For any finite absorbing Markov chain, for any distribution of starting states with no inaccessible states, and for any distributions of the costs with finite variances the $TD(\lambda)$ algorithm given by

1. $V_{n+1}(i) = V_n(i) + \alpha_n(i)\sum_{t=1}^{m}[c_{i_t} + \gamma V_n(i_{t+1}) - V_n(i_t)]\sum_{k=1}^{t}(\gamma\lambda)^{t-k}\chi_i(k)$,

   $\sum_n \alpha_n(i) = \infty$ and $\sum_n \alpha_n^2(i) < \infty$ uniformly w.p.1.

2. $V_{t+1}(i) = V_t(i) + \alpha_t(i)[c_{i_t} + \gamma V_t(i_{t+1}) - V_t(i_t)]\sum_{k=1}^{t}(\gamma\lambda)^{t-k}\chi_i(k)$,

   $\sum_n \alpha_t(i) = \infty$ and $\sum_n \alpha_t^2(i) < \infty$ uniformly w.p.1 and within sequences $\alpha_t(i)/max_{t\in S}\alpha_t(i) \to 1$ uniformly w.p.1.

converges to the optimal predictions w.p.1 provided $\gamma, \lambda \in [0,1]$ with $\gamma\lambda < 1$.

Proof:

1) Here the learning rule used is

$$V_{n+1}(i) = V_n(i) + \alpha_n(i)[G_n(i) - \frac{m(i)}{E[m(i)]}V_n(i)] \tag{3.5}$$

$$G_n(i) = \frac{1}{E[m(i)]}\sum_{k=1}^{m(i)} V_n^{\lambda}(i;k) \tag{3.6}$$

where $V_n^{\lambda}(i;k)$ is an estimate calculated at the k^{th} occurrence of state i in a sequence. For mathematical convenience, we make $\alpha_n(i) \rightarrow E[m(i)]\alpha_n(i)$, where $m(i)$ is the number of times the state i has been visited.

Similarly to the proof of Theorem 2.2, we start from the equation (2.3), and we take

$\Delta_t(i) = V_t(i) - V_t^*(i)$,

$\alpha_n(i) = \alpha_n(i)m(i)/E[m(i)]$,

$\beta_n(i) = \alpha_n(i)$,

and $F_n(i) = G_n(i) - V^*(i)m(i)/E[m(i)]$

to reach the equation defined in the first part of this theorem. We are only left with proving that $F_n(i)$ satisfy the conditions defined in Theorem 2.1. It can be shown that

$$max_i|E[F_n(i)|V_i]| \leq \gamma \max_i |V_n(i) - V^*(i)| \tag{3.7}$$

using the fact that $V_n^{\lambda}(i)$ is a contraction mapping independent of possible discounting. The variance of $F_n(i)$ can be seen to be bounded by $E[m^4]\max_i |V_n(i)|^2 \leq C(k)\max_i |V_n(i)|^2$ implying that the variance of $F_n(i)$ is within the bounds of Theorem 2.1. Thus, the batch version of $TD(\lambda)$ convergence to the optimal predictions w.p.1.

Chapter 4

Conclusion

In this report, we presented two model-free RL algorithms, which are the Q-learning and the $TD(\lambda)$ algorithms. We also discussed their convergence and, based on the work done in [1], and it has been proved that they both converge to the optimal prediction w.p.1. The prove has been done by extending the results from stochastic approximation theory to cover asynchronous relaxation processes that have a contraction property with respect to some maximum norm. The proofs presented in [1] are simple and uses only high-level statistical properties of the estimates and do not rely on constructions specific to the algorithms.

Bibliography

[1] Tommi Jaakkola, Michael I Jordan, and Satinder P Singh. Convergence of stochastic iterative dynamic programming algorithms. In *Advances in neural information processing systems*, pages 703–710, 1994.

[2] Richard S Sutton and Andrew G Barto. Reinforcement learning: An introduction. 2011.