The Mathematics of Reinforcement Learning

Gilberto Batres-Estrada [*]

Keywords: Dynamic Programming, Bellman Equation, Reinforcement Learning, Markov Chains, AI.

I. Introduction

According to Haykin, Reinforcement Learning (RL) is a form of unsupervised learning, a form of behavioural learning. It is performed through interaction between an agent and its environment, where the agent tries to achieve a goal in the presence of uncertainties. Because Reinforcement Learning is a type of unsupervised learning, it is attractive for dynamic situations where it is costly or difficult to gather a set of input-output examples. There are two approaches to RL:

- **The classical approach**, learning is achieved through reward and punishment reaching a highly skilled behaviour.
- **The modern approach**, achieved with a mathematical technique known as dynamic programming with an emphasis on planning future stages without experiencing them.

These notes focus on dynamic programming. Dynamic programming deals with decisions made in stages with the outcome of each decision being predictable to some extent before next decision is made. These decisions cannot be made in isolation, as a consequence the desire of a low cost at the present must be balanced against high cost in the future. Optimal planning requires an efficient tradeoff between immediate and future costs, which is caught by dynamic programming. The question answered by Bellman's dynamic programming is as follows [2]:

How can an agent or decision maker improve its long-term performance in a stochastic environment when the attainment of this improvement may require having to sacrifice short-term performance?

With mathematical modelling we have to deal with two challenges:

- the realistic description of a given problem
- the power of analytic and computational methods to apply to the problem

[*]gilberto.batres-estrada@live.com

The focus of dynamic programming lies in decision making by an agent that operates in a stochastic environment. The framework of Markov decision processes is used to solve this problem. A Markov decision process provides a basis for choosing a sequence of decisions that will maximise the returns from an N-stage decision-making process. These notes follows closely [2].

II. Markov Decision Process

Consider an agent as in fig XX. The agent operates according to a finite-discrete-time Markovian decision process, [2]:

- The environment evolves *probabilistically*, occupying a finite set of discrete states. However the state does not contain past statistics even though these statistics could be useful to the agent.
- For each environment stage, there is a finite set of possible actions that may be taken by the agent.
- Every time the agent takes an action, a certain cost is incurred.
- States are observed, actions are taken, and costs are incurred at discrete times.

Definition 1: *The state of the environment is a summary of the entire past experience of an agent gained from its interaction with the environment, such that the information necessary for the agent to predict the future behaviour of the environment is contained in that summary.*

Let us introduce the following variables which are going to be used throughout the text:

Variable	Description
X_n	Random variable at time-step n
i_n	Outcome or actual state of X_n
$\mathfrak{H}$	Finite set of states
$\mathfrak{A} = \{a_{ik}\}$	Set of actions for state i, and action k
p_{ij}	Transition probability form state i to state j
$\mathbb{E}$	Expectation operator

It is worth to note that the complexity of the dynamic programming algorithm is quadratic in the dimension of state space and linear in the dimension of the action space. The transition from state i to a new state j due to action a_{ik} is probabilistic. The transition probability being Markovian, which means that the transition from state i to state j depends entirely on the current state i and the corresponding action a_{ik}. Let ASn be a random variable denoting the action taken by the agent at time-step n and the transition probability from state i to state j be p_{ij}. If the action responsible for that transition is $A_n = a$ then [2]

$$p_{ij}(a) = P(X_{n+1} = j | X_n = i, A_n = a) \qquad (1)$$

with the probabilities fulfilling the Kolmogorov axioms of probability theory for normality and positivity, see Appendix. The sequence of states resulting from the actions taken by the agent form a Markov chain, see Appendix.

Each transition made by the agent comes with a cost. The cost for a transition from state i to state j is denoted by $\gamma^n g(i, a_{ik}, j)$, where γ is a scalar discount factor and a_{ik} is the action taken by the agent. The discount factor is in the range $0 \leq \gamma < 1$.

A policy is a mapping of states into actions, [2]:

Definition 2: *Policy is a rule used by the agent to decide what to do, given knowledge of the current state of the environment.*

The policy is denoted by

$$\pi = \{\mu_0, \mu_1, \dots\} \tag{2}$$

where μ_n is a function that maps the state $X_n = i$ into an action $A_n = a$ at a time-step $n = 0, 1, 2 \dots$ This mapping is such that

$$\mu_n(i) \in \mathfrak{A}_i, \textit{ for all states } i \in \mathfrak{H}$$

where $\mathfrak{A}_i$ is the set of all possible actions taken by the agent in state i. Such policies are said to be admissible.

Policies can be stationary or non stationary. A non stationary policy varies with time as given in eq. 2. When the policy is time-independent it can be expressed as $\pi = \{\mu, \mu, \dots\}$ and is called stationary. For a stationary policy the underlying Markov chain can may be stationary or non stationary. It is not recommended to use a non stationary Markov chain for a stationary policy. If a stationary policy is used, then the states $\{X_n, n = 0, 1, 2, \dots\}$ forms a Markov chain with probabilities $p_{ij}(\mu(i))$, where $\mu(i)$ is an action.

A. Formulation of Dynamic Programming

A dynamic programming problem is considered to be of a finite-horizon kind if the cost accumulates over a finite number of stages. On the other hand in an infinete-horizon problem the cost accumulates over an infinite number of stages. Infinite-horizon problems can be used to approximate problems of finite, but very large number of stages.

A.1. Cost-to-go Function J^π

The total expected cost in an infinite-horizon problem under a policy $\pi = \{\mu_n\}$ from state $X_0 = i$ is

$$J^\pi = \mathbb{E}\Big[\sum_{n=0}^{\infty} \gamma^n g(X_n, \mu_n(X_n), X_{n+1}) \big| X_0 = i\Big] \tag{3}$$

where $g(X_n, \mu_n(X_n), X_{n+1})$ is the cost from state X_n to state X_{n+1} under policy $\mu_n(X_n)$, γ is the discount factor, and the expectation is taken with respect to the Markov chain $\{X_1, X_2, \dots\}$. The optimal value of the cost-to-go function J^π is

$$J^*(i) = \min_\pi J^\pi(i). \tag{4}$$

A greedy policy π, with respect to $J^*(i)$, tries to minimize the immediate next cost without taking into account actions in the near future events, which might result in less better alternatives in the future. When using a stationary policy we use $J^\mu(i)$ instead of $J^*(i)$ and say that μ is optimal if

$$J^\mu(i) = J^*(i), \text{ for all initial states } i \tag{5}$$

Summary of dynamic programming: Given a stationary Markov decision process describing the interaction between an agent and its environment, find a stationary policy $\pi = \{\mu, \mu, \dots\}$ that minimizes the cost-to-go function $J^\pi(i)$ for all initial states i.

III. Bellman's Optimality Criterion

We start with Bellman's principle of optimality:

Definition 3: *An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy starting from the state resulting from the first decision.*

The term "decision" is a choice of control at a particular time and "policy" is the entire control sequence or control function. Let the cost-to-go function of a finite-horizon problem be given by

$$J_0(X_0) = \mathbb{E}\left[g_K(X_K) + \sum_{n=0}^{K-1} g_n(X_n, \mu_n(X_n), X_{n+1})\right] \tag{6}$$

where K is the planning horizon (i.e., number of stages) and $g_K(X_K)$ is the terminal cost. The exception is with respect to $X_1, \dots, X_{K-1}$. The principle of optimality is defined as follows, [2]:

Definition 4: *Let $\pi^* = \{\mu_0^*, \mu_1^*, \dots, \mu_{k-1}^*\}$ be an optimal policy for the basic finite-horizon problem. Assume that when using the optimal policy π^*, a given state X_n occurs with positive probability. Consider the subproblem where the environment is in state X_n at time n, and suppose we wish to minimize the corresponding cost-to-go function*

$$J_n(X_n) = \mathbb{E}\left[g_K(X_K) + \sum_{n=0}^{K-1} g_n(X_n, \mu_n(X_n), X_{n+1})\right] \tag{7}$$

for $n = 0, 1, \dots, K-1$. Then the truncated policy $\{\mu_n^*, \mu_{n+1}^*, \dots, \mu_{K-1}^*\}$ is optimal for the subproblem.

An optimal policy for a complex multistage planning or control problem is constructed as follows, [2]

1. Construct an optimal policy for the "tail subproblem" involving only the last stage of the system.
2. Extend the optimal policy to the "tail subproblem" involving the last two stages of the system.
3. Continue the procedure in this fashion until the entire problem has been dealt with.

A. Dynamic-Programming Algorithm

The algorithm starts backwards in time from period $N-1$ to period 0. Let $\pi = \{\mu_0, \mu_1, \ldots, \mu_{X-1}\}$ be an admissible policy. For each $n = 0, 1, \ldots, K-1$, let $\pi^n = \{\mu_n, \mu_{n+1}, \ldots, \mu_{K-1}\}$ and let $J_n^*(X_n)$ be the optimal cost for the $(K-n)$-stage problem that starts at state X_n and time n and ends at time X, then start with

$$J_n^*(X_n) = \min_{\pi^n} \underset{(X_{n+1},\ldots,X_{K-1})}{\mathbb{E}} \left[g_K(X_K) + \sum_{k=n}^{K-1} g_k(X_k, \mu_k(X_k), X_{k+1}) \right]. \tag{8}$$

Using $\pi^n = (\mu_n, \pi^{n+1})$ and partial expansion of the summation

$$\begin{aligned} J_n^*(X_n) &= \min_{(\mu_n,\pi^{n+1})} \underset{(X_{n+1},\ldots,X_{K-1})}{\mathbb{E}} \left[g_n(X_n, \mu_n(X_n), X_{n+1}) + g_K(X_K) + \sum_{k=n+1}^{K-1} g_k(X_k, \mu_k(X_k), X_{k+1}) \right] \\ &= \min_{\mu_n} \underset{X_{n+1}}{\mathbb{E}} \left\{ g_n(X_n, \mu_n(X_n), X_{n+1}) + \min_{\pi^{n+1}} \underset{(X_{n+2},\ldots,X_{K-1})}{\mathbb{E}} \left[g_K(X_K) + \sum_{k=n+1}^{K-1} g_k(X_k, \mu_k(X_k), X_{k+1}) \right] \right\} \\ &= \min_{\mu_n} \underset{X_{n+1}}{\mathbb{E}} \left[g_n(X_n, \mu_n(X_n), X_{n+1}) + J_{n+1}^*(X_{n+1}) \right] \end{aligned} \tag{9}$$

where in the last line the equation for $J_n^*(X_n)$ has been used with $n+1$ in place of n. This result in the following equation

$$J_n(X_n) = \min_{\mu_n} \underset{X_{n+1}}{\mathbb{E}} \left[g_n(X_n, \mu_n(X_n), X_{n+1}) + J_{n+1}(X_{n+1}) \right] \tag{10}$$

Dynamic-Programmin algorithm

Definition 5: *For every initial state X_0, the optimal cost $J^*(X_0)$ of the basic finite-horizon problem is equal to $J_0(X_0)$, where the function J_0 is obtained from the last step of the algorithm*

$$J_n(X_n) = \min_{\mu_n} \underset{X_{n+1}}{\mathbb{E}} \left[g_n(X_n, \mu_n(X_n), X_{n+1}) + J_{n+1}(X_{n+1}) \right] \tag{11}$$

which runs backward in time, with $J_K(X_K) = g_K(X_K)$ Furthermore, if μ_n^* minimizes the right-hand side 11 for each X_n and n, then the policy $\pi^* = \{\mu_0^*, \mu_1^*, \ldots, \mu_{K-1}^*\}$ is optimal.

B. Bellman's Optimality Equation

The dynamic-programming algorithm is defined for the finite-horizon case. To recast it to an infinite-horizon case the following is done

1. Reverse the time index in the algorithm
2. Define the cost $g_n(X_n, \mu_n(X_n), X_{n+1})$ as

$$g_n(X_n, \mu_n(X_n), X_{n+1}) = \gamma^n g(X_n, \mu_n(X_n), X_{n+1}) \tag{12}$$

This leads to the following form for the dynamic-programming algorithm

$$J_{n+1}(X_0) = \min_{\mu} \mathop{\mathbb{E}}_{X_1} \Big[g(X_0, \mu(X_0), X_1) + \gamma J_n(X_1) \Big] \tag{13}$$

where X_0 is the initial state, X_1 is the new state due to policy μ, γ is the discount factor and where the initial condition $J_0(X) = 0$ for all X. Equation

$$J^*(i) = \lim_{K \to \infty} J(K),\ \forall i \tag{14}$$

equates the finite-horizon to the infinite-horizon discounted problem. In the equation $J^*(i)$ is the optimal infinite-horizon cost, or the limit of the K-stage optimal cost $J_K(i)$ as $K \to \infty$. We can rewrite the above equation with $n + 1 = K$ and $X_0 = i$ as

$$J^*(i) = \min_{\mu} \mathop{\mathbb{E}}_{X_1} \Big[g(i, \mu(i), X_1) + \gamma J^*(X_1) \Big] \tag{15}$$

We can simplify further by the following procedure:

1. Compute the expectation of $g(i, \mu(i), X_1)$ with respect to X_1:

$$\begin{aligned} \mathbb{E}[g(i, \mu(i), X_1)] &= \sum_{j=1}^{N} p_{ij} g(i, \mu(i), j) \\ &= c(i, \mu(i)) \end{aligned} \tag{16}$$

   where Eq. 16 is called immediate expected cost incurred at state i, N is the number of states of the environment and as usual p_{ij} is the transition probability from state $X_0 = i$ to the new state $X_1 = j$.
2. Compute the expectation of $J^*(X_1)$ with respect to X_1. If we know the cost $J^*(X_1)$ for each state X_1 then we may determine the expectation of $J^*(X_1)$ in terms of the transition probabilities of the underlying Markov chain:

$$\mathbb{E}[J^*(X_1)] = \sum_{j=1}^{N} p_{ij} J^*(j) \tag{17}$$

We the help of the equations in the list above with p_{ij} as a function of the policy μ we get:

$$J^*(i) = \min_{\mu} \left(c(i, \mu(i)) + \gamma \sum_{j=1}^{N} p_{ij}(\mu) J^*(j) \right) \text{ for } i = 1, 2, \ldots, N \tag{18}$$

This derivation leads to Bellman's optimality equation: **Bellman's optimality equation:**

$$\boxed{J^*(i) = \min_{\mu} \left(c(i, \mu(i)) + \gamma \sum_{j=1}^{N} p_{ij}(\mu) J^*(j) \right) \text{ for } i = 1, 2, \ldots, N} \tag{19}$$

IV. Policy Iteration

We introduce the Q-factor as follows. If μ is an existing policy for a known cost-to-go function $J^\mu(i)$ for all states i, then the Q-factor for each state $i \in \mathfrak{H}$ and action $a \in \mathfrak{A}_i$ is defined as the immediate cost plus the sum of the discounted costs of all successor states under policy μ, i.e.

$$Q^\mu(i, a) = c(i, a) + \gamma \sum_{j=1}^{n} p_{ij}(a) J^\mu(j) \tag{20}$$

where we made use of $a = \mu(i)$. Note that $J^*(i) = \min_\mu Q^\mu(i, a)$. The policy μ is greedy with respect to the cost-to-go function $J^\mu(i)$ if for all states, $\mu(i)$ is an action that satisfies the condition:

$$Q^\mu(i, \mu(i)) = \min_{a \in \mathfrak{A}_i} Q^\mu(i, a) \text{ for all } i. \tag{21}$$

1. It is possible for more than one action to minimize the set of Q-factors for some state, in which case there can be more than one greedy policy with respect to the pertinent cost-to-go function.
2. A policy can be greedy with respect to many different cost-to-go functions.

The following result is basic to all dynamic-programming:

$$Q^{\mu*}(i, \mu^*(i)) = \min_{a \in \mathfrak{A}_i} Q^{\mu*}(i, a) \tag{22}$$

where μ^* is an optimal policy.

With the aid of the notion Q-factors and greedy policy, policy iteration goes as follows:

1. a policy evaluation step, in which the cost-to-go function for some current policy and the corresponding Q-factor are computed for all states and actions.
2. a policy improvement step, in which the current policy is updated in order to be greedy with respect to the cost-to-go function computed in step 1.

Appendix A. Appendix

Appendix A. Probability Theory

Kolmogorov gave a rigourous mathematical definition of probability as follows [1]. Let $(\Omega, \mathfrak{F}, P)$ be a probability space for a sample space Ω and a collection of events $\mathfrak{F}$ also called a σ-algebra of measurable subsets of Ω, see [1]. Moreover P is a measure with the following properties, [1]

- For any $A \in \mathfrak{F}$, then there exists a $P(A)$, the probability of A satisfying $P(A)$.
- $P(\Omega) = 1$.
- Let $\{A_n, n \geq 1\}$ be a collection of pairwise disjoint events, and let A be their union, then $P(A) = \sum_{n=1}^{\infty} P(A_n)$.

Appendix B. Markov Chains

REFERENCES

[1] Allan Gut. *An intermediate course in probability.* ISBN 978-1-4419-0161-3; DOI 10.1007/978-1-4419-0162-0; Second edition, Springer Dordrecht Heidelberg London New York 2009.

[2] Simon Haykin. *Neural networks and learning machines.* ISBN-13: 978-0-13-147139-9; SBN-10: 0-13-147139-2; Third edition, Pearson, Prentice Hall, 2009.