6.S897/HST.956 Machine Learning for Healthcare

Lecture 16: Reinforcement Learning 1

Instructors: David Sontag, Peter Szolovits

1 Lecture Overview

Lecture 16 was taught by guest lecturer, Fredrik D. Johansson. Dr. Johansson is a postdoc in the Clinical Machine Learning group at MIT's Institute for Medical Engineering and Science (IMES), working with Professor David Sontag. In his lecture, Dr. Johansson covered the following topics:

- Overview of Treatment Policies and Potential Outcomes
- Introduction to Reinforcement Learning
- Decision Processes
- Reinforcement Learning Paradigms
- Learning from Off-Policy Data

Reinforcement learning in healthcare applications will be covered in detail in the following lecture.

2 Overview of Treatment Policies and Potential Outcomes

2.1 Lecture 15 Recap: Causal Effects

Causal Effects consider the following variables:

- Potential Effect, Y
- Covariate, X
- Treatment, T

The potential outcome for treatment T=1 is designated as Y(1). Similarly, the potential outcome for a control setting (T=0) is designated as Y(0). Using this notation, the Conditional Average Treatment Effect (CATE) is calculated as:

$$CATE(X) = \mathbb{E}[Y(1) - Y(0)|X] \tag{1}$$

where Y(1)-Y(0) represents the difference in potential outcomes and X represents the feature space.

In the healthcare setting, we may seek to design policies that recommend how to treat a patient based on his/her medical history or current state. Such policy may be designed to, for example, optimize for that patient's outcome. Furthermore, we can also think about much more complicated policies that are not just based on outcome, but also take into account legislation, cost, side effects, etc.

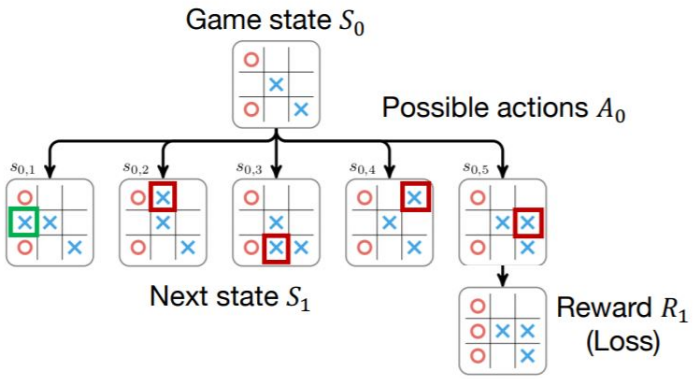


Figure 1: A simplified depiction of reinforcement learning to assess next move in the game Tic Tac Toe

3 Introduction to Reinforcement Learning

3.1 Reinforcement learning applications

Reinforcement learning is a technique which can be applied to find the optimal policy that maximizes reward. We have seen examples of reinforcement learning success stories in the gaming world, specifically with AlphaStar, AlphaGo, DQN Atari and Open AI Five. In these scenarios, reinforemcent learning assesses possible actions at a given state to influence the next state, with the goal of maximizing end reward. Figure 1 depicts how reinforecement learning is used to assess all possible actions at a given state to maximize overall reward.

Using the game analogy to apply reinforcement learning for treatment policy:

- Patient state at time S_t is like the game board
- Medical Treatments A_t are available actions
- Outcomes R_t are rewards

In the healthcare setting, a **policy** π recommends a treatment to a patient given his/her medical history or state. For a patient with medical history, x,

$$\pi(x) = \mathbb{I}[CATE(x) > 0] \tag{2}$$

3.2 Reinforcement learning for patient management

We turn to an example of Sepsis management. Sepsis is a life-threatening condition caused by infection that can lead to organ failure and death. For a septic patient, the clinician may need to determine whether or not to place that patient on a mechanical ventilator, as sepsis can often effect blood oxygen saturation or a patient's ability to breath. Subsequently, the clinician may need to determine whether to sedate the patient, and finally whether to administer vasopressors. The clinician should understand what will happen after each choice is made and use that knowledge to influence the next treatment plan. The states, actions, and responses present which influence the clinician's sequence of decisions is outlined in Figure 2.

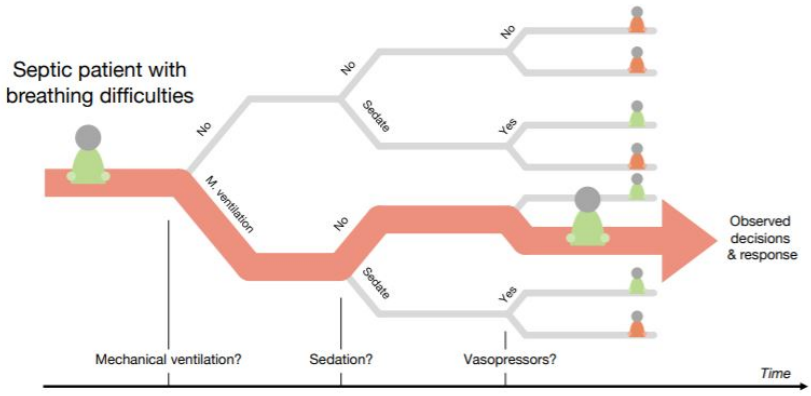


Figure 2: An illustration of a clinician's sepsis management decision tree for a given patient over time

4 Decision Processes

Definition 1 (Decision process)**.** *A* decision process *specifies how* S_t, A_t, *and* R_t *are distributed:* $p(S_0, \ldots, S_T, A_0, \ldots, A_T, R_0, \ldots, R_T)$.

Definition 2 (Behavior policy)**.** *A* behavior policy $\mu = p(A_t|\ldots)$ *specifies how an agent takes actions depending on the environment. Which environment variables show up in the conditional depends on the type of agent.*

4.1 Examples of decision processes

A **Markov decision process** (MDP) is a well-known type of decision process, where the states follow the Markov assumption that the state transitions, rewards, and actions depend only on the most recent state-action pair. See Figure 3(a) for an illustration. Algebraically, this means the states, actions and reward spaces have the following properties respectively:

$$p(S_t|S_0, \ldots, S_{t-1}, A_0, \ldots, A_{t-1}) = p(S_t|S_{t-1}, A_{t-1}) \tag{3}$$
$$p(A_t|S_0, \ldots, S_{t-1}, A_0, \ldots, A_{t-1}) = p(A_t|S_t) = \mu \tag{4}$$
$$p(R_t|S_0, \ldots, S_{t-1}, A_0, \ldots, A_{t-1}) = p(R_t|S_t, A_t) \tag{5}$$

Contextual bandits are another type of decision process following an additional stronger assumption, that the states are independent. See Figure 3(b) for an illustration, and notice the edges that have been removed from Figure 3(a). We can clearly see that this is equivalent to many single-step problems, and is in fact analogous to the potential outcomes estimation problem from last week: if we treat the S_t's as i.i.d patients, each A_t as the treatment, and R_t as the outcome of interest, then the reinforcement learning problem is essentially measuring the causal effect for each patient.

4.2 Value maximization

The goal of reinforcement learning is to understand the effect of many actions A_t on future rewards. Previously, in causal effect estimation, we were interested in the effect of a one-time treatment (action) on one

Figure 3: (a) Markov decision process. Red arrows highlight the dependencies from the Markov assumption. $A_0 \to A_T$ is an example of an edge that would violate the Markov assumption. (b) Contextual bandit. Each time step graph is equivalent to the simple causal effect graph we assumed in lectures 14 and 15.

outcome (reward). Therefore reinforcement learning, where we have multiple time steps and a more complex structure, is at least as hard as measuring the causal effect. Once the decision processes are learned, most reinforcement learning algorithms take a value-based approach, focusing on maximizing the expected cumulative reward or the value V_π for some π. Other approaches are discussed in section 5.

Definition 3 (Value of a policy)**.** *Let $G_t = \sum_{s=t}^{T} R_s$ represent the return at time step t, that is the total of the future rewards after and including t. Then the **value** $V_\pi = \mathbb{E}_{A_t \sim \pi}[G_0]$ represents the expected total rewards under the policy π. Here the expectation is taken w.r.t. possible actions under our learned policy.*

Consider the following toy example we might encounter in a healthcare setting, illustrated in Figure 4. We observe data from several patients, where binary actions at three timesteps are taken according to some policy π. Each patient has a different decision sequence, but their total return is calculated in the same way, by summing the returns at each time point. Since we don't know the actual decision process generating the actions A_t under π, we can approximate the expected total rewards by using our sample of n patients as $V_n \approx \frac{1}{n} \sum_{i=1}^{n} G^n$.

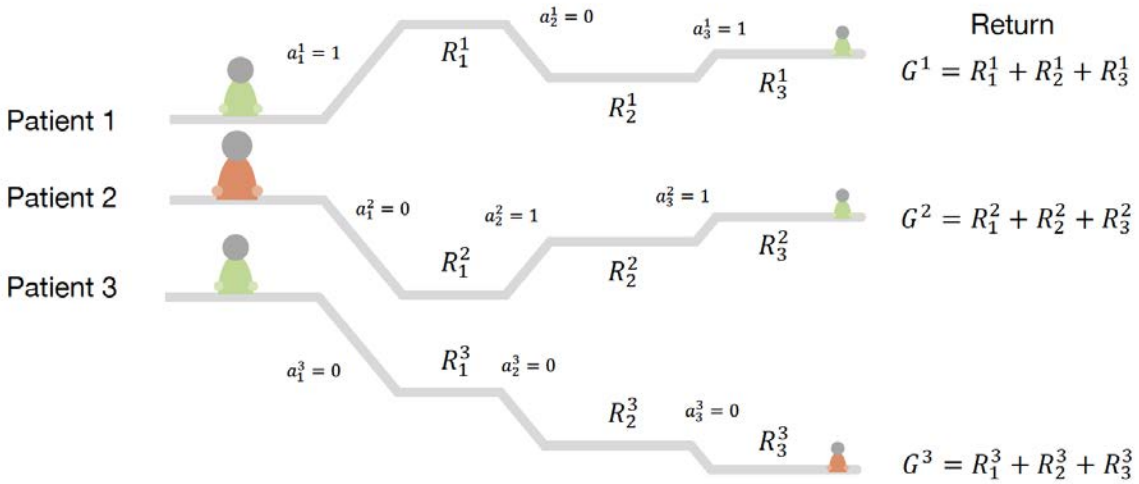


Figure 4: Example trajectories for patients observed under some policy π.

Though this example is simple, it is representative of healthcare setting where a policy is already in place and illustrates how we can evaluate it without actually proposing our own. In the following section, we will

see a more complex example, where we examine a game and attempt to find an optimal policy π.

5 Reinforcement Learning Paradigms

5.1 Three existing paradigms

There are three common paradigms in reinforcement learning:

- Model-based reinforcement learning
- Value-based reinforcement learning
- Policy-based reinforcement learning

In lecture 16, we focused on value-based reinforcement learning, which measures the overall value/return:

$$p(G_t|S_t, A_t) \tag{6}$$

The dominant method to use in value-based reinforcement learning is **Q-learning**.

5.2 Dynamic programming

Q-learning is an example of dynamic programming. The idea of dynamic programming for reinforcement learning is to recursively learn the best action/value in a previous state given the base action/value in future state.

For example, take the matrix shown in Figure 5. We want to find the best path to get us from the "Start" state to the optimal end state, terminal +1. The following rules apply:

- Can move up, left or right one cell at a time
- p(Move up given A="up")=0.8
- The reward for reaching terminal [4,3]=+1
- The reward for reaching terminal [4,2]=-1
- The cost of one step is -.04

Individually, one might manually find a method to determine a (seemingly) optimal solution. However, this problem can also be solved recursively using Q-Learning, which will be discussed further below.

5.3 Q-Learning for value based reinforcement learning

Recall that the value of a state, s, under a policy $\pi$is:

$$\nu_\pi(S) := \mathbb{E}_\pi[G_t|S_t = s] \tag{7}$$

The value of a state-action pair (s,a) is:

$$q_\pi(s, a) := \mathbb{E}_\pi[G_t|S_t = s, A_t = a] \tag{8}$$

Q-learning attempts to estimate q_π with a function Q(s,a) such that $\pi$is the deterministic policy

$$Q*(s, a) := \max_\pi q_\pi(s, a) := q*(s, a) \tag{9}$$

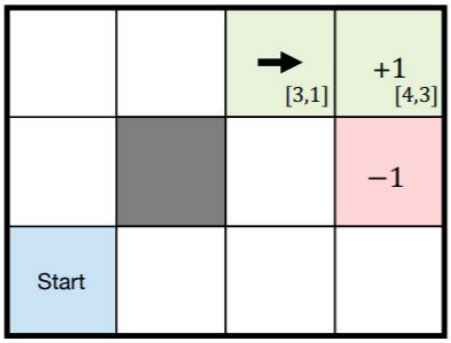


Figure 5: Peter Bodk's "Robot in a Room" reinforcement learning problem. The goal of this example is to identify optimal path from the start cell (bottom left) to the optimal terminal cell, +1 (top right)

For the optimal Q-function, q*, "Bellman Optimality" holds true. The Bellman equations states:

$$q^*(s,a) := \mathbb{E}_\pi[R_t + \gamma \max_{a'} q^*(S_{t+1}, a') | S_t = s, A_t = a] \tag{10}$$

When Q-states are discrete, Q-learning can be solved using dynamic programming using the following recursion steps:

- Initialize a table of Q(s,a)
- Repeat

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_t + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)] \tag{11}$$

Figure 6 depicts how recursion is used over a series of 6 discrete steps to identify the optimal direction to move in each of the available cells.

When the number of states K is large or S_t is not discrete, a table cannot be maintained for Q(s,a). Instead, Q(s,a) may be represented as a functional approximation, Q_{theta}:

$$R(Q_\theta) = \mathbb{E}_\pi[(R + \gamma \max_{a'} \hat{Q}(S', A') - Q_\theta(S, A))^2] \tag{12}$$

where old estimate of Q =

$$\hat{Q}(S', A') \tag{13}$$

and new estimate of Q =

$$Q_\theta(S, A) \tag{14}$$

In the scenario where there are no future cases, finding q(s,a) is the same as finding expected potential outcome:

$$R(Q_\theta) = \mathbb{E}_\pi[(R_t - Q_\theta(S, A))^2] \tag{15}$$

6 Learning from Off-Policy Data

In section 4 we saw an example of observing realized trajectories $(s_1, a_1, r_1) \ldots (s_T, a_T, r_T)$ from some behavior policy μ. Suppose now we want to determine the value V_π of some other policy π. As we discussed in the same section, learning policies based on this kind of data (**off-policy learning**) is at least as hard as causal effect estimation from observational data.

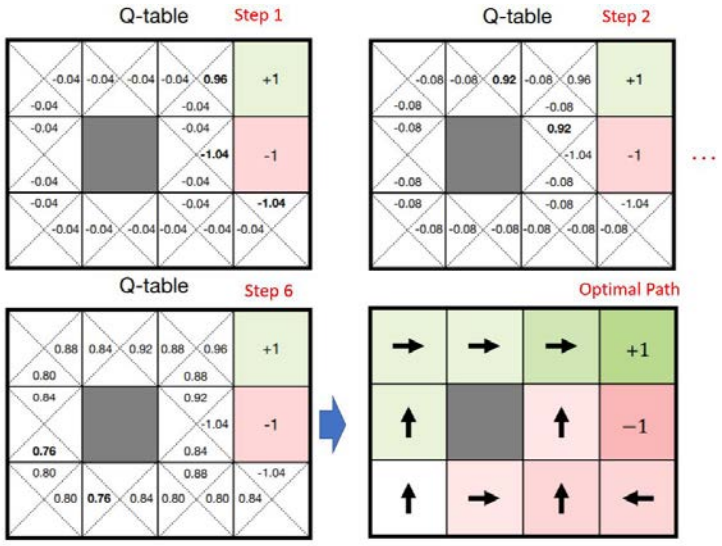


Figure 6: A depiction of dynamic programming to solve optimal path between start cell [1,1] and optimal terminal cell [4,3]

6.1 Assumptions

We continue drawing a parallel between these two approaches by discussing the assumptions that go into off-policy reinforcement learning:

- In the single-step case, we assumed we had overlap: $p(T = t|X = x) > 0 \quad \forall x, t$. Now, in the sequential case, we assume **positivity**: $p(A_t = a|\bar{S}_t, \bar{A}_{t-1}) > 0 \quad \forall a, t$, i.e. all actions are possible at all times.
- In the single-step case, we needed to assume ignorability: $Y(0), Y(1) \perp\!\!\!\perp T|X$, i.e. there are no hidden confounders. In the sequential case, we assume **sequential randomization**: $G(\ldots) \perp\!\!\!\perp A_t|\bar{S}_t, \bar{A}_{t-1}$, i.e. the reward is independent of policy given the past history. Here $\bar{S}_t$ represents all states up to t.

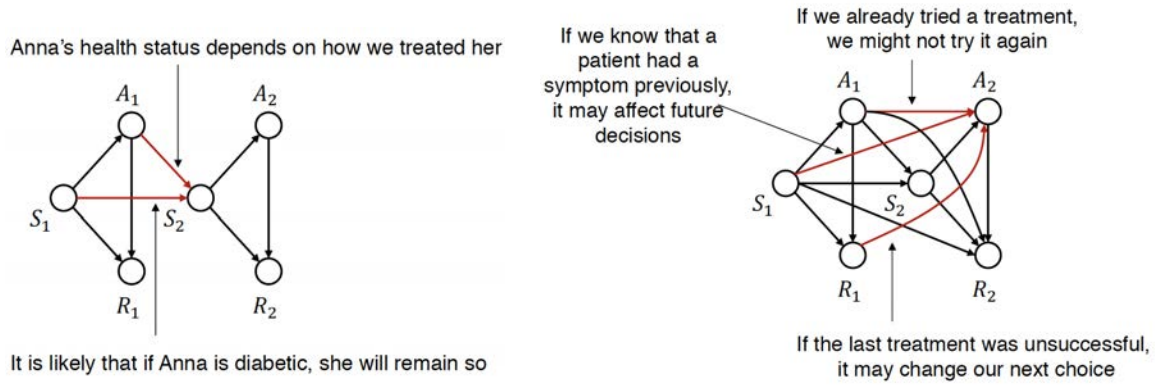
Together, these form sufficient conditions to identify the value function. Note that the assumption of sequential randomization is generally the harder of the two to satisfy, even though it is weaker than the Markov assumption, and is not always realistic in a healthcare setting.

6.2 In practice

To illustrate the challenge of reinforcement learning under the above assumptions, consider the example of treating a single patient, Anna, over time.

- In the case of contextual bandits, where the states at each time point are completely independent of each other, we could reduce the problem again to that of learning potential outcomes. However, Anna's state at time t is likely dependent on the action at the previous time step A_{t-1} and her previous state S_{t-1}. See Figure 7 for examples. Adding in these dependencies still satisfies Markov assumptions.
- In addition, Anna's future outcomes may depend on past treatments and states; more problematic is that her future treatments also likely depend on past treatments, states, and outcomes. See Figure 7 for examples. Adding in the latter dependencies violates our assumption.

In order to work around this sequential randomization problem, then, we encompass Anna's past history into the state variable, and ignorability is preserved. However, this too has its challenges in practice: history



(a) (b)

Figure 7: (a) Anna's current state may be dependent on past state and past treatment. (b) The current treatment is often influenced by past treatments (consistency or experimentation), past state, and past outcomes.

grows with time, and available history may differ among patients. One way to deal with this is to keep only the states and actions of the last k time steps so that the history captures a sliding window. Another approach is to learn a function that takes as input all past history and outputs into the current state what is relevant for the reinforcement learning problem, for example by using a RNN.

6.3 Contrast with successes

We concluded this lecture by revisiting the success stories in reinforcement learning we discussed at the start. With our new understanding, we see there are major differences between the gaming and healthcare applications that contribute to the difference in success:

- Full observability: everything important to the optimal action in a game is observed. In healthcare settings, we almost always have unobserved confounders, either affecting the outcome in the same time step or linked to future outcomes in a more complex way.
- Markov dynamics: in general we don't need to track history in a game - all information relevant to the future outcome is captured on the current board or state.
- Limitless exploration: any policy can be tested and resultant outcomes observed, if we just play the game enough; whereas in a healthcare setting you often have only observational data of a past study dote on a limited number of patients.
- Noiseless state/outcome: in games the we can deterministically evaluate the state and outcome, while in healthcare problems these are noisy.

Further challenges and approaches for reinforcement learning in healthcare applications will be discussed in the upcoming lecture 17.

References

[PCC+17] Niranjani Prasad, Li-Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara E Engelhardt. A reinforcement learning approach to weaning of mechanical ventilation in intensive care units.

In *Proceedings of the Conference on Uncertainty in Artificial Intelligence*. AUAI, 2017.

6.S897 / HST.956 Machine Learning for Healthcare

Spring 2019