POLICY GRADIENT METHODS, CURVATURE, AND DISTRIBUTION SHIFT

SHAM KAKADE

These are notes from a talk given by Dr. Sham Kakade on November 2nd on some rather high-level concepts in reinforcement learning. The talk can be accessed **here**.

In this write-up, I'll provide a very brief conceptual overview of some ideas that Dr. Kakade deals with before diving into his talk.

1. What is reinforcement learning?

Reinforcement learning is an area of machine learning that is concerned with the behaviour of agents who are concerned with maximizing utility. In a sense, it is paradigmatically divergent from **supervised** & **unsupervised** learning, which focus upon recognizing patterns within a given domain.

Reinforcement learning is studied by scholars schooled in a multitude of different fields. Game theorists, control theorists, researchers in operations research, and statisticians, among others, study reinforcement learning.

Basic reinforcement is modeled as a Markov decision process (MDP), consisting of four elements:

- A set of environment and agent states, $\mathcal{S}$. Can be either finite or infinite. We assume it's finite or countably infinite.
- A set of actions $\mathcal{A}$ of an agent. Can be either finite or finite, but we typically assume it's finite.
- A transition function $P : \mathcal{S} \times \mathcal{A} \to \delta(\mathcal{S})$, where $\delta(\mathcal{S})$ is the space of probability distributions over $\mathcal{S}$. $P\left(s' \,\middle|\, s, a \right)$. In other words, it's the *probability simplex*.
- $P_a(s, s') = \Pr\left(\mathcal{S}_{t+1} = s' \,\middle|\, \mathcal{S}_t = s, a_t = a\right)$
- A feward function $r : \mathcal{S} \times \mathcal{A} \to [0, 1]$. This is the agent's reward for taking action a at state s.
- A discount factor $\gamma \in [0, 1)$, which defines a horizon for the problem. Intuitively, γ considers the tiem-scale of the pay-off (are we rewarding delayed gratification? In which case, γ will be large. Are we rewarding immediate gratification? In which case, γ will be small.)
- An initial state distribution $\mu \in \delta(\mathcal{S})$, which specifies how the initial state s_0 is generated.

A Markov decision process occurs over discrete steps of time t with can be characterized with a state s_t and an agent's action a_t. The goal of the agent is to learn a policy $\pi : A \times S \to [0, 1]$, $\pi(a, s) = \Pr\left(a_t = a \,\middle|\, s_t = s \right)$

Date: November 2, 2020.

2. What are policy methods?

In a given MDP $M = (\mathcal{S}, \mathcal{A}, P, r, \gamma, \mu)$, the agent interacts with the environment according to the following protocol: the agent starts at s_0 μ & at each time step $t = 0, 1, 2, \ldots$, the agent takes an action $a_t \in \mathcal{A}$, obtains the immediate reward $r_t(s_t, a_t)$, and observes the next state s_{t+1} sampled according to s_{t+1} $P\left(\cdot \,\middle|\, s_t, a_t \right)$

The interaction record τ at time t,

$$\tau_t = (s_0, a_0, r_1 \ldots s_t)5$$

is called the trajectory at time t and includes the state at time t, s_t.

A policy specifies a decision-making strategy in which the agent is informed by the history of their observations (in this sense, a Markov decision process should be considered conceptually different from a Markov chain in terms of agnosticity to history).

A policy is a (possibly randomized) mapping from a trajectory to an action. Thus,

$$\pi : \mathcal{H} \to \delta(\mathcal{A})$$

, where $\mathcal{H}$ is the set of all possible trajectories (of all lengths) and $\delta\mathcal{A}$ is the space of all probability distributions over $\mathcal{A}$. A stationary policy $\pi : \mathcal{S} \to \mathcal{A}$ specifies a decision making strategy based solely on the current state (that is $a_t \sim \pi\left(\cdot \,\middle|\, s_t \right)$). A determinstic stationary policy is of the form $\pi : \mathcal{S} \to \mathcal{A}$

Now, we define **values** for general policies. For a fixed policy and a starting state $s_{0=s}$, we define the value function $V_m^\pi : \mathcal{S} \to \mathbb{R}$ as the discounted sum of future rewards

$$V_m^\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \left(\gamma^t r(s_t, a_t) \,\middle|\, \pi, s_0 = s \right) \right] \tag{1}$$

Where the expectation is computed with respect to the randomness of the trajectory, that is, the randomness in state-transitions and stochasticity of π. Since $r(s, a)$ is bounded between 0 & 1, we have $0 \leq V_m^\pi(s) \leq 1/(1-\gamma)$

Similarly, the action-value (or Q-value) function $Q_M^\pi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as

$$Q_m^\pi = \mathbb{E}\left[\sum_{t=0}^{\infty} \left(r(s_t, a_t) \,\middle|\, \pi, s_0 = s, a_0 = a \right) \right] \tag{2}$$

, which is also bounded from below by 0 and above by $1/(1-\gamma)$

The optimization that the agent aims to solve is to maximize V_M^π by finding a policy π. The agent thus wants to maximize

$$\max_\pi V_M^\pi(s) \tag{3}$$

3. What was Dr. Kakade talking about?

Definition 1 (The state space & state variables)**.** *A* ***state variable*** *is one of the set of variables that are used to describe the "state" of a dynamical system – it describes enough of a system to determine its future behaviour in the absence of any external forces affecting the system.*

*The **state space** is the Euclidean space in which the variables on the axis are the state variables. It is a mathematical model of a physical system as a set of input, output, and state varibles related by first-order differential equations or difference equations.*

The tabular dynamic programming approach is commonly taken to describe a reinforcement learning problem:

State s	Action a	State-action value $Q^\pi(s,a)$
...	...	...
⋮	⋮	⋮
...	...	...

The third column consists of a policy and gives the reward. It's a one-step look-ahead (that is, it gives the reward for the next step after). If we could magically compute it for each row, we could update our policy to be greedy and it will converge quickly. Unfortunately, the table is very large.

The RL approach is to generalize; instead of querying the entire table, we can take a sampling approach in order to find an optimal policy. Policy gradient methods are one such approach, which easily deal with large state/action spaces. The policy is directly parameterized (for instance, through a neural network) with the gradient easily computed with simulation-based methods. You don't need to know the model. In ML, it's common to directly optimize the quantity of interest.

Why this is possible is an open-question: the gradients in supervised learning (SL) and reinforcement learning are very different. We don't care about non-convexity much in SL, which isn't the case in RL. In RL, the regions can have exponentially small gradients, with much higher-order regions being flat.

References

[1] A. Agarwal, N. Jiang, S.M. Kakade, W. Sun; Reinforcement Learning: Theory and Algorithms; accessed October 27, 2020; available at `https://rltheorybook.github.io/`