Q-learning Algorithm

Justin Sirignano

University of Illinois at Urbana-Champaign

October 17, 2019

Deep Reinforcement Learning

- Algorithms for training deep learning models to estimate an optimal control.
- Successes: Atari games, Go, Starcraft.
- Potential applications: robotics, driverless cars, UAVs.
- Relatively few mathematical guarantees.

Markov Decision Problem

- We consider a Markov decision problem defined on finite state space $\mathcal{X} \subset \mathbb{R}^{d_x}$.
- In each state, an action $a \in \mathcal{A} \subset \mathbb{R}^{d_a}$ can be taken.
- The probability transition function is $p(z|x,a) = \mathbb{P}[x_{j+1} = z | x_j = x, a_j = a]$.
- The reward function is $r(x,a)$.
- The objective is to determine a policy $a^* : \mathcal{X} \to \mathcal{A}$ that maximizes

$$W(x) = \mathbb{E}\left[\sum_{j=0}^{\infty} \gamma^j r(x_j, a_j) | x_0 = x\right], \tag{1}$$

where $a_j = a^*(x_j)$.

Bellman Equation

$$\begin{aligned} 0 &= r(x,a) + \gamma \sum_{z \in \mathcal{X}} \max_{a' \in \mathcal{A}} V(z,a') p(z|x,a) - V(x,a), \\ a^*(x) &= \arg\max_{a \in \mathcal{A}} V(x,a). \end{aligned} \tag{2}$$

- In principle, the Bellman equation can be solved to find the optimal control.
- However, the transition probability function $p(z|x,a)$ (i.e., the state dynamics) may not be known.
- Even if they are known, the state space may be too high-dimensional for standard numerical methods to solve (2) due to the curse-of-dimensionality.

Reinforcement Learning

- Reinforcement learning approximates the solution to the Bellman equation with a function approximator $Q(x, a; \theta)$ such as a (deep) neural network.

- The parameters of the neural network, denoted by θ, are estimated using the Q-learning algorithm.

- The neural network in Q-learning is referred to as a “Q-network".

Q-learning

The Q-learning algorithm attempts to minimize the objective function

$$L(\theta) = \sum_{x,a \in \mathcal{X} \times \mathcal{A}} \left[\big(Y(x,a) - Q(x,a;\theta) \big)^2 \right] \pi(x,a), \tag{3}$$

where π is a probability measure which is positive for every (x, a) and the "target" Y is

$$Y(x,a) = r(x,a) + \gamma \sum_{x' \in \mathcal{X}} \max_{a' \in \mathcal{A}} Q(x', a'; \theta) p(x'|x,a). \tag{4}$$

- If $L(\theta) = 0$, then $Q(x, a; \theta)$ is a solution to the Bellman equation.
- In practice, the hope is that the Q-learning algorithm will learn a model Q such that $L(\theta)$ is small and therefore $Q(x, a; \theta)$ is a good approximation for the Bellman solution $V(x, a)$.

$$L(\theta) = \sum_{x,a \in \mathcal{X}\times\mathcal{A}} \left[\big(Y(x,a) - Q(x,a;\theta) \big)^2 \right] \pi(x,a),$$

$$Y(x,a) = r(x,a) + \gamma \sum_{x' \in \mathcal{X}} \max_{a' \in \mathcal{A}} Q(x',a';\theta) p(x'|x,a). \tag{5}$$

- Take the "gradient" of $L(\theta)$ but treat Y as a constant.

$$\theta_{k+1} = \theta_k - \alpha \sum_{x,a} \left[\big(Y(x,a) - Q(x,a;\theta) \big) \right] \nabla_\theta Q(x,a;\theta) \pi(x,a). \tag{6}$$

- To make the update (6) computationally efficient, use the stochastic approximation:

$$\begin{aligned} \theta_{k+1} &= \theta_k + \alpha G_k, \\ G_k &= \left(r(x_k,a_k) + \gamma \max_{a' \in \mathcal{A}} Q(x_{k+1},a';\theta_k) - Q(x_k,a_k;\theta_k) \right) \\ &\times \nabla_\theta Q(x_k,a_k;\theta_k), \end{aligned} \tag{7}$$

where, for example, (x_k, a_k) is an ergodic Markov chain with a unique stationary distribution $\pi(x,a)$.

Deep Q-learning

A "$Q-$network" $Q(x,a;\theta)$ is trained using the Q-learning algorithm

$$\begin{aligned} \theta_{k+1} &= \theta_k + \alpha G_k, \\ G_k &= \Big(y_k - Q(x_k, a_k; \theta_k)\Big) \nabla_\theta Q(x_k, a_k; \theta_k), \end{aligned} \tag{8}$$

where

$$y_k = r(x_k, a_k) + \gamma \max_{a' \in \mathcal{A}} Q(x_{k+1}, a'; \theta_k) \tag{9}$$

is considered to be the "target" for iteration k.

- Unlike in the stochastic gradient descent algorithm, the Q-learning update directions G_k are not unbiased estimates of a descent direction for the objective function $L(\theta)$.
- The Q-learning algorithm takes the derivative of $L(\theta)$ *while treating the target Y as a constant*. Therefore,

$$\mathbb{E}\big[G_k | \theta_k, x_k, a_k\big] \neq \nabla_\theta \left[\Big(Y(x_k, a_k) - Q(x_k, a_k; \theta_k) \Big)^2 \right]. \tag{10}$$

Q-learning: mathematical challenges

- Unlike in the stochastic gradient descent algorithm, the Q-learning update directions G_k are not unbiased estimates of a descent direction for the objective function $L(\theta)$.

- It is not necessarily clear that an update step will lead to the objective function decreasing.

- Many of the mathematical challenges of analyzing Q-learning arise from this fact.

A $Q-$network is trained using the Q-learning algorithm

$$\theta_{k+1} = \theta_k + \Big(y_k - Q(x_k, a_k; \theta_k)\Big) \nabla_\theta Q(x_k, a_k; \theta_k), \tag{11}$$

where

$$y_k = r(x_k, a_k) + \gamma \max_{a' \in \mathcal{A}} Q(x_{k+1}, a'; \theta_k) \tag{12}$$

is considered to be the target for iteration k.

The Q-learning algorithm estimates an approximation to the Bellman equation using stochastic gradient descent (but where the gradient is not taken with respect to the target y_k).