Lecture 5: Policy Gradient Methods

SUMS 707 - Basic Reinforcement Learning

Gabriela Moisescu-Pareja and Viet Nguyen

McGill University, Mila

February 18, 2021

Recap: Model-free control

- The generalized policy iteration blueprint:
  - Policy evaluation: TD or MC
  - Policy improvement: ϵ-greedy
- We perform computations for Q instead of V, as from the Bellman optimality equations:

$$T^*V(s) = \sup_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a)} \left[V(s') \right] \right\}$$

$$T^*Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s' \sim \mathbb{P}(\cdot|s,a)} \left[\sup_{a' \in \mathcal{A}} Q(s', a') \right]$$

- The application of the Bellman optimality operator can be expressed as an *expectation*.

Model-free control

- We saw that TD policy evaluation + ϵ-greedy = default SARSA algorithm
- We also saw the Q-learing algorithm and its inherent off-policy nature, allowing us to evaluate the greedy policy while playing the ϵ-greedy policy
- These algorithms converge in the tabular setting

Recap: Iterative value function approximation and control

Big idea: approximate the true value function V^π by a differentiable parametric function $V(\cdot;\theta) \approx V^\pi$.

- Define loss:

$$J(\theta) = \mathbb{E}_\pi \left[\left(V^\pi(s) - V(s;\theta) \right)^2 \right]$$

- Update direction:

$$\Delta\theta = \alpha \mathbb{E}_\pi \left[\left(V^\pi(s) - V(s;\theta) \right) \nabla_\theta V(s;\theta) \right]$$

- SGD updates essentially sample from the above:

$$\Delta\theta = \alpha \left(V^\pi(s) - V(s;\theta) \right) \nabla_\theta V(s;\theta)$$

We substitute V^π with the target of our choice (MC, TD, TD(λ)). *When controlling, estimate Q^π instead.*

Batch RL

- Our incremental methods are not very sample efficient
- We now seek the best fitting value function given the agent's experience (we imagine the agent to remember everything that it has done up until the current timestep)
- Recycle past experiences

Least Squares

Given our function approximation $V(s;\theta)$, we have a hypothetical dataset:

$$\mathcal{D} = \{(s_t, V_t^\pi) : t \in [T]\}$$

We want to find θ such that our estimate V best fits this dataset. This is exactly the problem setup of *least-squares*, we minimize the sum of squared errors:

$$LS(\theta) = \sum_{t=1}^{T} (V_t^\pi - V(s_t;\theta))^2$$

Finding a θ such $LS(\theta)$ is minimized is the idea behind Batch RL.

Experience replay

We store a dataset $\mathcal{D}$ as defined previously,

- We sample $(s, V^{\pi}) \sim \mathcal{D}$
- We update with stoch. grad. descent:

$$\Delta\theta = \alpha(V^{\pi} - V(s;\theta))\nabla_{\theta} V(s, \theta)$$

From theory: this process converges to the least squares solution:

$$\theta^{\pi} = \arg\min_{\theta} LS(\theta)$$

DQN

- Initialize Q to be a neural network. It's job is to approximate the true Q^*.
- Take action a_t according to ϵ-greedy policy
- Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $\mathcal{D}$
- Sample a *batch* $S = \{(s, a, r, s')\} \sim \mathcal{D}$
- Compute Q-learning targets w.r.t. old, fixed parameters θ^-
- Optimize MSE between Q-network and Q-learning targets:

$$MSE(\theta_i) = \mathbb{E}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i)\right)^2\right]$$

The expression $r + \gamma \max_{a'} Q(s', a'; \theta_i^-)$ is the *target* in our Q-learning algorithm!