Lecture 13 - On-policy Control with Approximation

Instructor: GuangBing Yang, PhD

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Introduction to On-policy Control with Approximation

- With parametric approximation of action-value function $\hat{q}(s, a, W) \approx q_*(s, a)$
- The semi-gradient Sarsa algorithm—the natural extension of semi-gradient TD(0)—to action value and to on-policy control will be studied this lecture
- Based on on-policy GPI, using ϵ-greedy for action selection
- n-step linear Sarsa on the Mountain Car problem
- continuing case for the average-reward case with differential values.

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Episodic Semi-gradient Control

- The approximation action-value function $\hat{q} \approx q_\pi|_W$
- Consider examples of the form $S_t, A_t \rightarrow U_t$:
  - The update target U_t can be any approximation of $q_\pi(S_t, A_t)$, including the usual backed-up values such as the full Monte Carlo return (G_t) or any of the n-step Sarsa returns
- The general gradient-descent update for action-value prediction is:
  - $W_{t+1} = W_t + \alpha[U_t - \hat{q}(S_t, A_t, W_t)] \nabla \hat{q}(S_t, A_t, W_t)$
- For example, the update for the one-step Sarsa method is:

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Episodic semi-gradient one-step Sarsa

- For example, the update for the one-step Sarsa method is:
- $W_{t+1} = w_t + \alpha[R_{t+1} - \gamma\hat{q}(S_t, A_t, w_t) - \hat{q}(S_t, A_t, w_t)] \nabla \hat{q}(S_t, A_t, w_t)$
- For a constant policy, this method converges in the same way that TD(0) does, with the same kind of error bound
- The control method works with this action-value prediction for policy improvement and action selection
- Suitable techniques for continuous actions, or to actions from large discrete sets, are a topic of ongoing research with as yet no clear resolution

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Episodic semi-gradient one-step Sarsa

- If the action set is discrete and not too large, for each possible action a available in the current state S_t
- Compute $\hat{q}(S_t, a, w_t)$ and then find the greedy action $A_t^* = \text{argmax}_a \hat{q}(S_t, a, w_t)$.
- Policy improvement is then done by changing the estimation policy to a soft approximation of the greedy policy such as the ϵ-greedy policy. Actions are selected according to this same policy.
- Pseudocode for the complete algorithm is given next

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Episodic semi-gradient one-step Sarsa

Episodic Semi-gradient Sarsa for Estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

Loop for each episode:
  $S, A \leftarrow$ initial state and action of episode (e.g., ε-greedy)
  Loop for each step of episode:
    Take action A, observe R, S'
    If S' is terminal:
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
      Go to next episode
    Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\big[R + \gamma\hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})\big]\nabla\hat{q}(S, A, \mathbf{w})$
    $S \leftarrow S'$
    $A \leftarrow A'$

Cited from Sutton and Barto, 1998; 2017;Introduction to Reinforcement Learning

Example: Mountain Car Task

- The task is driving an underpowered car up a steep mountain road. Shown to the right.
- The problem is:
  - the gravity is stronger than the car's engine
  - the car cannot accelerate up the steep slope even with full throttle
- The only solution is to first move away from the goal and up the opposite slope on the left.
- Then, by applying full throttle the car can build up enough inertia to carry it up the steep slope even though it is slowing down the whole way.

Cited from Sutton and Barto, 1998; 2017;Introduction to Reinforcement Learning

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Example: Mountain Car Task

- This is a simple example of a continuous control task where things have to get worse in a sense (farther from the goal) before they can get better.
- The reward in this problem is -1 on all time steps until the car moves past its goal position at the top of the mountain, which ends the episode.
- There are three possible actions:
  - full throttle forward (+1),
  - full throttle reverse (-1), and
  - zero throttle (0).
  - The car moves according to a simplified physics. Its position, x_t, and velocity, $\dot{x}_t$
  - are updated by the gravity is stronger than the car's engine the car cannot accelerate up the steep slope even with full throttle

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Example: Mountain Car Task

- $X_{t+1} = \text{bound}[x_t + \dot{x}_{t+1}], x_{t+1} = \text{bound}[x_t + 0.001A_t - 0.0025\cos(3x_t)]$,
- where the bound operation enforces $-1.2 \leq x_{t+1} \leq 0.5$ and $-0.07 \leq \dot{x}_{t+1} \leq 0.07$
- when x_{t+1} reached the left bound, $\dot{x}_{t+1} = 0$, reset to zero.
- When it reached the right bound, the goal was reached and the episode was terminated.
- Each episode started from a random position $x_t \in [-0.6, -0.4)$ and zero velocity.

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

9

9

Example: Mountain Car Task

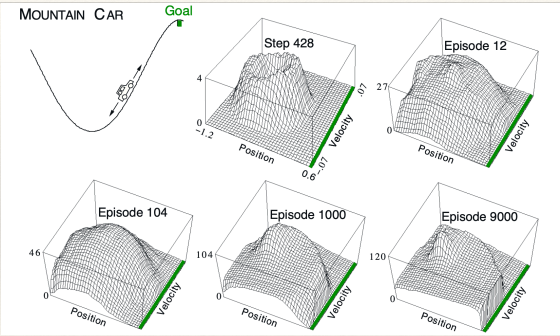
- Use grid-tilings to convert the two continuous state variables to binary features.
- In this example, use 8 tilings, with each tile covering 1/8th of the bounded distance in each dimension and asymmetrical offsets.
- The feature vectors x(s,a) created by tile coding were then combined linearly with the parameter vector to approximate the action-value function
- $\hat{q}(s, a, W) = W^T X(s, a) = \sum_{i=1}^{d} w_i x_i(s, a) \quad \forall s, a$
- When it reached the right bound, the goal was reached and the episode was terminated.
- Several learning curves for semi-gradient Sarsa on this problem is given as:

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

10

10

Example: Mountain Car Task



The Mountain Car task and the cost-to-go function (-max_a q(s,a, w)) learned during one run. Cited from Sutton and Barto, 1998; 2017;Introduction to Reinforcement Learning

11

11

Semi-gradient n-step Sarsa

- Use an n-step return as the update target in the semi-gradient Sarsa update equation to obtain an n-step version of episodic semi-gradient Sarsa.
- $G_{t:t+n} = R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n \hat{q}(S_{t+n}, A_{t+n}, W_{t+n-1}), t + n < T, ,$
  - with: $G_{t:t+n} = G_t$ if $t + n \geq T$.
- The n-step update equation is:
  - $W_{t+n} = w_{t+n-1} + \alpha[G_{t:t+n} - \hat{q}(S_t, A_t, w_{t+n-1}] \nabla \hat{q}(S_t, A_t, w_{t+n-1}), 0 \leq t < T$

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

12

12

Episodic Semi-gradient n-step Sarsa

Episodic semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_*$ or q_π

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Input: a policy π (if estimating q_π)
Algorithm parameters: step size $\alpha > 0$, small $\varepsilon > 0$, a positive integer n
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
All store and access operations (S_t, A_t, and R_t) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim \pi(\cdot|S_0)$ or ε-greedy wrt $\hat{q}(S_0, \cdot, \mathbf{w})$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$:
    If $t < T$, then:
      Take action A_t
      Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
      If S_{t+1} is terminal, then:
        $T \leftarrow t+1$
      else:
        Select and store $A_{t+1} \sim \pi(\cdot|S_{t+1})$ or ε-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
    $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
    If $\tau \geq 0$:
      $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n, T)} \gamma^{i-\tau-1} R_i$
      If $\tau + n < T$, then $G \leftarrow G + \gamma^n \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w})$ $(G_{\tau:\tau+n})$
      $\mathbf{w} \leftarrow \mathbf{w} + \alpha\,[G - \hat{q}(S_\tau, A_\tau, \mathbf{w})]\,\nabla \hat{q}(S_\tau, A_\tau, \mathbf{w})$
  Until $\tau = T - 1$

Cited from Sutton and Barto, 1998; 2017;Introduction to Reinforcement Learning

Average reward: a new problem setting for continuing tasks

- The problems exist in the interaction between agent and environment without termination or start states when applying the average reward setting in continuing problems.
- There is no discounting—the agent cares about delayed rewards and immediate reward.
- The discounted setting does not work well for function approximation
- To replace the average-reward setting:
  - the quality of a policy π is defined as the average rate of reward, or simply average reward, while following that policy, which we denote as r(π)

$$r(\pi) = \lim_{h\to\infty} \frac{1}{h} \sum_{t=1}^{h} \mathbb{E}[R_t \,|\, S_0, A_{0:t-1} \sim \pi]$$

  - $r(\pi) = \lim_{h\to\infty} \mathbb{E}[R_t \,|\, S_0, A_{0:t-1} \sim \pi] = \sum_s \mu_\pi(s) \sum_a \pi(a \,|\, s) \sum_{s',r} p(s', r \,|\, s, a) r$

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Average reward: a new problem setting for continuing tasks

- The steady state distribution: $\sum_s \mu_\pi(s) \sum_a \pi(a \,|\, s) p(s' \,|\, s, a) = \mu_\pi(s')$
- The return of average-reward is given as:
  - $G_t = R_{t+1} - r(\pi) + R_{t+2} - r(\pi) + R_{t+3} - r(\pi) + \ldots$
  - This is known as the *differential return*, and the corresponding value functions are known as *differential value functions*.

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Differential value functions

- They have Bellman equations:

$$v_\pi(s) = \sum_a \pi(a \,|\, s) \sum_{r,s'} p(s', r \,|\, s, a)[r - r(\pi) + v_\pi(s')],$$

$$q_\pi(s, a) = \sum_{r,s'} p(s', r \,|\, s, a)[r - r(\pi) + \sum_{a'} \pi(a' \,|\, s') q_\pi(s', a')],$$

$$v_*(s) = \max_a \sum_{r,s'} p(s', r \,|\, s, a)[r - max_\pi r(\pi) + v_*(s')],$$

$$q_*(s, a) = \sum_{r,s'} p(s', r \,|\, s, a)[r - max_\pi r(\pi) + max_{a'} q_*(s', a')]$$

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Differential form of the TD Errors

- There is also a differential form of the two TD errors:

$$\delta_t = R_{t+1} - \bar{R}_t + \hat{v}(S_{t+1}, w_t) - \bar{v}(S_t, w_t),$$

  - $\delta_t = R_{t+1} - \bar{R}_t + \hat{q}(S_{t+1}, A_{t+1}, w_t) - \bar{q}(S_t, A_t, w_t),$
  - where $\bar{R}_t$ is an estimate at time t of the average reward r(π)
- With these alternate definitions, most of our algorithms and many theoretical results carry through to the average-reward setting without change
- The average reward version of semi-gradient Sarsa is defined with the differential version of the TD error:
  - $W_{t+1} = w_t + \alpha\delta_t \nabla \bar{q}(S_t, A_t, w_t)$, with δ_t given by above two formula

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Differential semi-gradient Sarsa

Differential semi-gradient Sarsa for estimating $\hat{q} \approx q_*$

Input: a differentiable action-value function parameterization $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$
Algorithm parameters: step sizes $\alpha, \beta > 0$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Initialize average reward estimate $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., $\bar{R} = 0$)

Initialize state S, and action A
Loop for each step:
  Take action A, observe R, S'
  Choose A' as a function of $\hat{q}(S', \cdot, \mathbf{w})$ (e.g., ε-greedy)
  $\delta \leftarrow R - \bar{R} + \hat{q}(S', A', \mathbf{w}) - \hat{q}(S, A, \mathbf{w})$
  $\bar{R} \leftarrow \bar{R} + \beta\delta$
  $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S, A, \mathbf{w})$
  $S \leftarrow S'$
  $A \leftarrow A'$

Cited from Sutton and Barto, 1998; 2017;Introduction to Reinforcement Learning

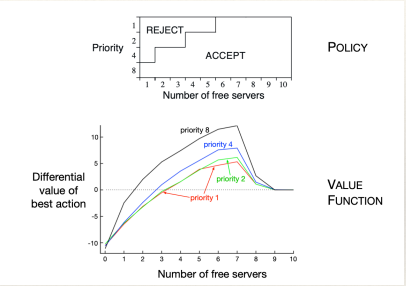
Example: An Access-Control Queuing Task

- The **problem**: Make a decision task involving access control to a set of 10 servers.
- **Settings**:
  - Customers of four different priorities arrive at a single queue.
  - The customers pay a reward of 1, 2, 4, or 8 to the server,
  - With higher priority customers paying more.
  - In each time step, the customer at the head of the queue is either accepted (assigned to one of the servers) or rejected (removed from the queue, with a reward of zero).
  - In either case, on the next time step the next customer in the queue is considered.
  - The queue never empties, and the priorities of the customers in the queue are equally randomly distributed.
  - Always reject customer if there is no free server.
  - Each busy server becomes free with probability p = 0.06 on each time step.
  - Assume the statistics of arrivals and departures are unknown.
- The **task:**
  - decide on each step whether to accept or reject the next customer,
  - maximize long-term reward without discounting based on the customer's priority and the number of free servers

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Example: An Access-Control Queuing Task

- Thus this is a problem of differential action- value estimate for each pair of state (number of free servers and priority of the customer at the head of the queue) and action (accept or reject).
- Figure on the right shows the solution found by differential semi-gradient Sarsa with parameters $\alpha = 0.01, \beta = 0.01, \epsilon = 0.1$. The initial action values and $\bar{R} = 0$.



Cited from Sutton and Barto, 1998; 2017;Introduction to Reinforcement Learning

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Deprecating the discounted setting

- For the tabular case, in which the returns from each state can be separately identified and averaged, the continuing discounted problem formulation has been very useful.
- But it is not easy to use continuing discounted formulation in the approximate cases.
- For example, consider an infinite sequence of returns with no beginning or end, and no clearly identified states.
- The states might be represented only by feature vectors, which may do little to distinguish the states from each other.
- As a special case, all of the feature vectors may be the same.
- Thus one really has only the reward sequence (and the actions), and performance has to be assessed purely from these.
  - One way is by averaging the rewards over a long interval—this is the idea of the average-reward setting.
  - By measuring the discounted return for each time step, the discounting can be used, but have to average them over a sufficiently large time interval because some returns would be small and some big

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Deprecating the discounted setting

- In the continuing setting there are no starts and ends, and no special time steps, so there is nothing else that could be done.
- However, doing this makes the average of the discounted returns being proportional to the average reward.
- In fact, for policy π, the average of the discounted returns is always $r(\pi)/(1-\gamma)$,
- That is, it is essentially the average reward, $\gamma(\pi)$.
- In particular, the ordering of all policies in the average discounted return setting would be exactly the same as in the average-reward settings.
- The discount rate γ thus has no effect on the problem formulation. It could in fact be zero and the ranking would be unchanged

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Differential semi-gradient n-step Sarsa

- In order to generalize to n-step bootstrapping, we need an n-step version of the TD error.
- generalizing the n-step return to its differential form, with function approximation:
- $G_{t:t+n} = R_{t+1} - \bar{R}_{t+n-1} + \ldots + R_{t+n} - \bar{R}_{t+n-1} + \hat{q}(S_{t+n}, A_{t+n}, w_{t+n-1})$
- Where $\bar{R}$ is an estimate of r(π), $n \geq 1$,and $t+n < T$. if $t+n \geq T, G_{t:t+n} = G_t$
- The n-step TD error is then: $\delta_t = G_{t:t+n} - \bar{q}(S_t, A_t, w)$

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Differential semi-gradient n-step Sarsa

Differential semi-gradient n-step Sarsa for estimating $\hat{q} \approx q_\pi$ or q_*

Input: a differentiable function $\hat{q} : \mathcal{S} \times \mathcal{A} \times \mathbb{R}^d \to \mathbb{R}$, a policy π
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)
Initialize average-reward estimate $\bar{R} \in \mathbb{R}$ arbitrarily (e.g., $\bar{R} = 0$)
Algorithm parameters: step size $\alpha, \beta > 0$, a positive integer n
All store and access operations (S_t, A_t, and R_t) can take their index mod $n+1$

Initialize and store S_0 and A_0
Loop for each step, $t = 0, 1, 2, \ldots$:
  Take action A_t
  Observe and store the next reward as R_{t+1} and the next state as S_{t+1}
  Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$, or ε-greedy wrt $\hat{q}(S_{t+1}, \cdot, \mathbf{w})$
  $\tau \leftarrow t - n + 1$ (τ is the time whose estimate is being updated)
  If $\tau \geq 0$:
    $\delta \leftarrow \sum_{i=\tau+1}^{\tau+n} (R_i - \bar{R}) + \hat{q}(S_{\tau+n}, A_{\tau+n}, \mathbf{w}) - \hat{q}(S_\tau, A_\tau, \mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \beta\delta$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\nabla\hat{q}(S_\tau, A_\tau, \mathbf{w})$

Cited from Sutton and Barto, 1998; 2017;Introduction to Reinforcement Learning

Recap

- In this lecture we discussed an approach for applying the ideas of parameterized function approximation and semi-gradient descent to on-policy control approximation problem.
- The approach for the episodic case is straightforward from the prediction to the control.
- But for continuing cases, maximizing the average reward setting per time step has to be used to formulate the problem of approximation of the control.

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Recap

- Discounting concepts no longer exist in the approximation of the control in continuing cases.
- Most policies cannot be represented by a value function. To rank the policies, the scalar average reward r(π) provides an effective way to do this.
- The average reward formulation involves new differential versions of value functions, Bellman equations, and TD errors, but all of these parallel the old ones, and the conceptual changes are small. There is also a new parallel set of differential algorithms for the average-reward case.

yguangbing@gmail.com, Guang.B@chula.ac.th Apr 27 2021

Questions and Lab