Algorithms in Reinforcement Learning

Guannan Hu

March 19, 2018

Algorithm 1 Semi-gradient TD(λ) for estimating $\hat{v} \approx v_\pi$

Input: the policy π to be evaluated
Input: a differentiable function $\hat{v} : \mathcal{S}^+ \times \mathbb{R}^d \to \mathbb{R}$ such that $\hat{v}(terminal, \cdot) = 0$
Algorithm parameters: step size $\alpha > 0$, trace decay rate $\gamma \in [0, 1]$
Initialize value-function weights $\mathbf{w}$ arbitrarily (e.g., $\mathbf{w} = \mathbf{0}$)

loop for each episode:
  Initialize S
  $\mathbf{z} \leftarrow \mathbf{0}$
  repeat for each step of episode:
    Choose $A \sim \pi(\cdot|S)$
    Take action A, observe $R, S^{'}$
    $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + \nabla\hat{v}(S, \mathbf{w})$
    $\delta \leftarrow R + \gamma\hat{v}(S^{'}, \mathbf{w}) - \hat{v}(S, \mathbf{w})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$
    $S \leftarrow S^{'}$
  until $S^{'}$ is terminal
end loop

Algorithm 2 True Online $TD(\lambda)$ for estimating $\mathbf{w}^T\mathbf{x} \approx v_\pi$

Input: the policy π to be evaluated
Input: a feature function $\mathbf{x} : \mathcal{S}^+ \leftarrow \mathbb{R}^d$ such that $\mathbf{x}(terminal, \cdot) = \mathbf{0}$
Algorithm parameters: step size $\alpha > 0$, trace decay rate $\gamma \in [0, 1]$
Initialize value-function weights $\mathbf{w} \in \mathbb{R}^d$ (e.g., $\mathbf{w} = \mathbf{0}$)

loop for each episode:
  Initialize state and obtain initial feature vector $\mathbf{x}$
  $\mathbf{z} \leftarrow \mathbf{0}$
  $V_{old} \leftarrow 0$
  repeat for each step of episode:
    Choose $A \sim \pi$
    Take action A, observe $R, \mathbf{x}^{'}$ (feature vector of the next state)
    $V \leftarrow \mathbf{w}^T\mathbf{x}$
    $V^{'} \leftarrow \mathbf{w}^T\mathbf{x}$
    $\delta \leftarrow R + \gamma V^{'} - V$
    $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + (1 - \alpha\gamma\lambda\mathbf{z}^T\mathbf{x})\mathbf{x}$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha(\delta + V - V_{old})\mathbf{z} - \alpha(V - V_{old})\mathbf{x}$
    $V_{old} \leftarrow V^{'}$
    $\mathbf{x} \leftarrow \mathbf{x}^{'}$
  until $\mathbf{x}^{'} = \mathbf{0}$ (signaling arrival at the terminal state)
end loop

Algorithm 3 Sarsa(λ) with binary features and linear function approximation for estimating $\mathbf{w}^T\mathbf{x} \approx q_\pi$ or q_*

Input: a function $\mathcal{F}(s, a)$ returning the set of (indices of) active features for s, a
Input: a policy π (if estimating q_π)
Algorithm parameters: step size $\alpha > 0$, trace decay rate $\lambda \in [0, 1]$
Initialize: $\mathbf{w} = (w_1, ..., w_d)^T \in \mathbb{R}^d$ (e.g., $\mathbf{w} = \mathbf{0}$), $\mathbf{z} = (z_1, ..., z_d)^T \in \mathbb{R}^d$

loop for each episode:
    Initialize S
    Choose $A \sim \pi(\cdot|S)$ or ϵ-greedy according to $\hat{q}(S, \cdot, \mathbf{w})$
    $\mathbf{z} \leftarrow \mathbf{0}$
    loop for each step of episode:
        Take action A, observe $R, S^{'}$
        $\delta \leftarrow R$
        loop for i in $\mathcal{F}(S, A)$
            $\delta \leftarrow \delta - w_i$
            Set
$$z \leftarrow \begin{cases} z_i + 1 & \text{(accumulating traces)} \\ 1 & \text{(replaceing traces)} \end{cases}$$
        end loop
        if $S^{'}$ is terminal then **then**
            $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$
            Go to next episode
        end if
        Choose $A^{'} \sim \pi(\cdot|S^{'})$ or near greedily $\sim \hat{q}(S^{'}, \cdot, \mathbf{w})$
        loop for i in $\mathcal{F}(S^{'}, A^{'})$:
            $\delta \leftarrow \delta + \gamma w_i$
        end loop
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha\delta\mathbf{z}$
        $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z}$
        $S \leftarrow S^{'}; A \leftarrow A^{'}$
    end loop
end loop

Algorithm 4 True Online Sarsa(λ) for estimating $\mathbf{w}^T\mathbf{x} \approx q_\pi$ or q_*

Input: a feature function $\mathbf{x} : \mathcal{S}^+ \times \mathcal{A} \leftarrow \mathbb{R}^d$ such that $\mathbf{x}(terminal, \cdot) = \mathbf{0}$
Input: a policy π (if estimating q_π)
Algorithm parameters: step size $\alpha > 0$, trace decay rate $\lambda \in [0, 1]$
Initialize: $\mathbf{w} \in \mathbb{R}^d$ (e.g., $\mathbf{w} = \mathbf{0}$)

loop for each episode:
  Initialize S
  Choose $A \sim \pi(\cdot|S)$ or near greedily from S using $\mathbf{w}$
  $\mathbf{w} \leftarrow \mathbf{x}(S, A)$
  $\mathbf{z} \leftarrow \mathbf{0}$
  $Q_{old} \leftarrow 0$
  repeat for each step of episode:
    Take action A, observe $R, S^{'}$
    Choose $A^{'} \sim \pi(\cdot|S^{'})$ or near greedily from $S^{'}$ using $\mathbf{w}$
    $\mathbf{x}^{'} \leftarrow \mathbf{x}(S^{'}, A^{'})$
    $Q \leftarrow \mathbf{w}^T\mathbf{x}$
    $Q^{'} \leftarrow \mathbf{w}^T\mathbf{x}^{'}$
    $\delta \leftarrow R + \gamma Q^{'} - Q$
    $\mathbf{z} \leftarrow \gamma\lambda\mathbf{z} + (1 - \alpha\gamma\lambda\mathbf{z}^T\mathbf{x})\mathbf{x}$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha(\delta + Q - Q_{old})\mathbf{z} - \alpha(Q - Q_{old})\mathbf{x}$
    $Q_old \leftarrow Q^{'}$
    $\mathbf{x} \leftarrow \mathbf{x}^{'}$
    $A \leftarrow A^{'}$
  until $S^{'}$ is terminal
end loop

Algorithm 5 REINFORCE: Monte-Carlo Policy-Gradient Method (episodic) estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differential policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Algorithm parameter: step size $\alpha > 0$
Initializa policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ (e.g., to $\mathbf{0}$)

loop forever (for each episode):
  Generate an episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
  loop for each step of the episode $t = 0, 1, ..., T-1$:
    $G \leftarrow \sum_{k=t+1}^{T} R_k$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\gamma^t G \nabla \ln\pi(A_t|S_t, \boldsymbol{\theta})$
  end loop
end loop

Algorithm 6 REINFORCE with Baseline (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s, \mathbf{w})$
Algorithm parameters: step sizes $\alpha^{\boldsymbol{\theta}} > 0, \alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)

loop forever (for each episode):
    Generate an episode $S_0, A_0, R_1, ..., S_{T-1}, A_{T-1}, R_T$, following $\pi(\cdot|\cdot, \boldsymbol{\theta})$
    loop for each step of episode $t = 0, 1, ..., T-1$:
        $G \leftarrow \sum_{k=t+1}^{T} R_k \qquad (G_t)$
        $\delta \leftarrow G - \hat{v}(S_t, \mathbf{w})$
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} \gamma^t \nabla \hat{v}(S_t, \mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} \gamma^t \delta \nabla \ln \pi(A_t|S_t, \boldsymbol{\theta})$
    end loop
end loop

Algorithm 7 One-step Actor Critic (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameter $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s, \mathbf{w})$
Algorithm parameter: step size $\alpha^{\boldsymbol{\theta}} > 0, \alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)

loop forever (for each episode):
    Initialize S (first state of episode)
    $I \leftarrow 1$
    loop while S is not terminal (for each time step):
        $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
        Take action A, observe S', R
        $\delta \leftarrow R + \gamma \hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$ (if S' is terminal, then $\hat{v}(S', \mathbf{w}) \doteq 0$)
        $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}} I \delta \nabla \hat{v}(S, \mathbf{w})$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}} I \delta \nabla \ln \pi(A|S, \boldsymbol{\theta})$
        $I \leftarrow \gamma I$
        $S \leftarrow S'$
    end loop
end loop

Algorithm 8 Actor-Critic with Eligibility Traces (episodic), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\hat{v}(s, \mathbf{w})$
Input: a differentiable state-value function parameterization $\hat{v}(s, \mathbf{w})$
Algorithm parameters: trace-decay rate $\lambda^{\boldsymbol{\theta}} \in [0,1], \lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0, \alpha^{\mathbf{w}} > 0$
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$)

loop forever (for each episode):
  initialize S (first state of episode)
  $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
  $\mathbf{z}^{\mathbf{w}} \leftarrow \mathbf{0}$ (d'-component eligibility trace vector)
  $I \leftarrow 1$
  loop while S is not terminal (for each time step):
    $A \in \pi(\cdot|S, \boldsymbol{\theta})$
    Take action A, observe S', R
    $\delta \leftarrow \gamma\hat{v}(S', \mathbf{w}) - \hat{v}(S, \mathbf{w})$ (if S' is terminal, then $\hat{v}(S', \mathbf{w})$)
    $\mathbf{z}^{\mathbf{w}} \leftarrow \gamma\lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + I\nabla\hat{v}(s, \mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \gamma\lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + I\nabla\hat{v}(s, \boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}\delta\mathbf{z}^{\mathbf{w}}}$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha^{\theta}\delta\mathbf{z}^{\boldsymbol{\theta}}$
    $I \leftarrow \gamma I$
    $S \leftarrow S'$
  end loop
end loop

Algorithm 9 Actor-Critic with Eligibility Traces (continuing), for estimating $\pi_{\boldsymbol{\theta}} \approx \pi_*$

Input: a differentiable policy parameterization $\pi(a|s, \boldsymbol{\theta})$
Input: a differentiable state-value function parameterization $\hat{v}(s, \mathbf{w})$
Algorithm parameters: trace-decay rates $\lambda^{\boldsymbol{\theta}} \in [0,1], \lambda^{\mathbf{w}} \in [0,1]$; step sizes $\alpha^{\boldsymbol{\theta}} > 0, \alpha^{\mathbf{w}} > 0, \eta > 0$
Initialize $\bar{R} \in \mathbb{R}$ (e.g., to 0)
Initialize policy parameter $\boldsymbol{\theta} \in \mathbb{R}^{d'}$ and state-value weights $\mathbf{w} \in \mathbb{R}^{d}$ (e.g., to $\mathbf{0}$) $\doteq 0$

loop forever (for each time step):
    $A \sim \pi(\cdot|S, \boldsymbol{\theta})$
    Take Action A, observe $S^{'}, R$
    $\delta \leftarrow R - \bar{R} + \hat{v}(S^{'}, \mathbf{w}) - \hat{v}(S, \mathbf{w})$
    $\bar{R} \leftarrow \bar{R} + \eta\delta$
    $\mathbf{z}^{\mathbf{w}} \leftarrow \lambda^{\mathbf{w}}\mathbf{z}^{\mathbf{w}} + \nabla\hat{v}(S, \mathbf{w})$
    $\mathbf{z}^{\boldsymbol{\theta}} \leftarrow \lambda^{\boldsymbol{\theta}}\mathbf{z}^{\boldsymbol{\theta}} + \nabla \ln \pi(A|S, \boldsymbol{\theta})$
    $\mathbf{w} \leftarrow \mathbf{w} + \alpha^{\mathbf{w}}\delta\mathbf{z}^{\mathbf{w}}$
    $\mathbf{w} \leftarrow \boldsymbol{\theta} + \alpha^{\boldsymbol{\theta}}\delta\mathbf{z}^{\boldsymbol{\theta}}$
    $S \leftarrow S^{'}$
end loop

Algorithm 10 Double DQN with proportional prioritization

Input: minibatch k, step-size η, replay period K, and size N, exponents α and β, budget T.
Initialize replay memory $\mathcal{H} = \Phi, \Delta = 0, p_1 = 1$
Observe S_0 and choose $A_0 \sim \pi_\theta(S_0)$
for $t = 1 \to T$ **do**
  observe S_t, R_t, γ_t
  Store transition $(S_{t-1}, A_{t-1}, R_t, \gamma_t, S_t)$ with $\mathcal{H}$ with maximal priority $p_t = \max_i p_i$
  if $t \equiv 0 \mod K$ **then**
    for $j = 1 \to k$ **do**
      Sample transition $j \sim P(j) = \frac{p_j^\alpha}{\max_i w_i}$
      Compute importance-sampling weight $w_j = \frac{(N \cdot P(j))^{-\beta}}{\max_i w_i}$
      Compute TD-error $\delta_j = R_i + \gamma_j Q_{target}(S_j, \text{argmax}_a Q(S_j, a)) - Q(S_{j-1}, A_{j-1})$
      Update transition priority $p_j \leftarrow |\delta_j|$
      Accumulate weight change $\Delta \leftarrow \Delta + w_i \cdot \delta_j \cdot \nabla_\theta Q(S_{j-1}, A_{j-1})$
    end for
    Update weights $\theta \leftarrow \theta + \eta \cdot \Delta$, reset $\Delta = 0$
    From time to time copy weights into target network $\theta_{target} \leftarrow \theta$
  end if
  Choose action $A_t \sim \pi_\theta(S_t)$
end for

Algorithm 11 Distributed Prioritized Experience Replay

procedure ACTOR(B, T)
 $\theta_0 \leftarrow$ LEARNER.PARAMETERS()
 $s_0 \leftarrow$ ENVIRONMENT.INITIALIZE()
 for $t = 1 \to T$ **do**
  $a_{t-1} \leftarrow \pi_\theta(s_{t-1})$
  $(r_t, \gamma_t, s_t) \leftarrow$ ENVIRONMENT.STEP(a_{t-1})
  LOCALBUFFER.ADD($(s_{t-1}, a_{t-1}, r_t, \gamma_t)$)
  if LOCALBUFFER.SIZE() $\geq B$ **then**
   $\tau \leftarrow$ LOCALBUFFER.GET(B)
   $p \leftarrow$ COMPUTERPRIORITIES(τ)
   REPLAY.ADD(τ, p)
  end if
  PERIODICALLY($\theta_t \leftarrow$ LEARNER.PARAMETERSOP())
 end for
end procedure

procedure LEARNER(T)
 $\theta_0 \leftarrow$ INITIALIZENETWORK()
 for $t = 0 \to T$ **do**
  $id, \tau \leftarrow$ REPLAY.SAMPLE()
  $l_t \leftarrow$ COMPUTELOSS($\tau; \theta_t$)
  $\theta_{t+1} \leftarrow$ UPDATEPARAMETERS($l_t; \theta_t$)
  $p \leftarrow$ COMPUTEPRIORITIES()
  REPLAY.SETPRIORITY(id, p)
  PERIODICALLY(REPLAY.REMOVETOFIT())
 end for
end procedure

Algorithm 12 Policy iteration algorithm guaranteeing non-decreasing expected return η

Initialize π_0

for $i = 0, 1, 2, ...$ until convergence **do**

Compute all advantage values $A_{\pi_i}(s, a)$.

Solve the constrained optimization problem

$\pi_{i+1} = \arg\max_\pi [L_{\pi_i}(\pi) - C D_{KL}^{max}(\pi_i, \pi)]$

where $C = 4\epsilon\gamma/(1-\gamma)^2$

and $L_{\pi_i}(\pi) = \eta(\pi_i) + \sum_s \rho_{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$

$$\pi_{i+1} = \arg\max_\pi [L_{\pi_i}(\pi) - C D_{KL}^{max}(\pi_i, \pi)]$$

$$\text{and } L_{\pi_i}(\pi) = \eta(\pi_i) + \sum_s \rho_{\pi_i}(s) \sum_a \pi(a|s) A_{\pi_i}(s, a)$$

end for