DM887 Assignment2 Q1

Jiawei Zhao

19.03.2024

Implementation of Least-Squares Temporal Differences (LSTD) Deep Q-Learning with Nonlinear Feature Extraction that maps a state to lower-dimensional latent embedding. While the action-value function should be a linear function of the output of the feature extractor.

Algorithm 1 Initialization

---

1. Initialize a variational autoencoder $\mathbf{A}$ as the feature extraction network with initial weights w_0 [w_0 are drawn from Glorot uniform initialization]
2. Initialize the learning rate of $\mathbf{A}$ as α
3. Initialize an Adam optimizer o with a learning rate α for $\mathbf{A}$
4. Initialize the minibatch size of DQN as k
5. Initialize the total number of training episodes N, the number of episodes at each cycle consisting of three phases of training N_0, ensure that $N \mod N_0 = 0$
6. Set the number of episodes at the warm-up phase as N_1, at the autoencoder update phase as N_2, and at the LSTD weight update phase as N_3, ensure that $(N_0 - N_1) \mod (N2 + N3) = 0$ [It would be a wise choice to seperate the warm-up phase, the antoencoder update phase, and the LSTD update phase at each cycle]
7. Initialize the number of maximum time step per episode T
8. Initialize the weights θ_0 for linear approximation function randomly between $(0, 1)$ which will be used for LSTD
9. Initialize a replay memory buffer $\mathcal{D}$ with a capacity $N \times T$
10. Initialize a relatively discount factor, e.g. $\gamma = 0.9$
11. Initialize a small constant λ, e.g. 1×10^{-3} to initialize A^{-1} for LSTD
12. Given the number of actions as N_a and embedding dimension N_e, initialize a matrix θ with shape $N_a \times N_e$
13. Initialize a tensor $A^{-1} = \lambda^{-1} I$ with shape $N_a \times N_e \times N_e$ and a tensor b with shape $N_a \times N_e$ full of zeros to store the tensors that suffice $\theta_a = A_a^{-1} b_a, \forall a \in A$ [A relatively large discount factor encourages long-term planning and faster convergence during training]

[1]

Algorithm 2 Warm-up phase

Freeze the weights of $f(phi(s))$, i.e. θ
Freeze the weights of $\mathbf{A}$, i.e. w
for *episode* $e_0 = 1$ to N_1 **do**
    Initialize state s
    Preprocess s into s_0 to adapt it as input of $\mathbf{A}$ [preprocessing of high-dimensional states is necessary w.r.t. autoenconders]
    for each time step $t = 1$ to T **do**
        Encode state s_t using $\mathbf{A}$ to get latent embedding $\phi(s_t)$
        Select action a_t using ϵ-greedy policy with $\epsilon = 0.2$ with an ϵ decay
        Execute a_t and obtain reward r_t and new state s_{t+1}
        if $s_{t+1} \notin S$ **then**
            break
        end if
        Store the transition of the current t, i.e. (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    end for
end for
$e = e + N_1$

[1]

Algorithm 3 Autoencoder update phase

Freeze the weights of $f(phi(s))$, i.e. θ
Unfreeze the weights of $\mathbf{A}$, i.e. w
for *episode* $e_0 = 1$ to N_2 **do**
    Repeat the same steps at the warm-up phase
    for each time step $t = 1$ to T **do**
        Repeat the same steps at the warm-up phase
[Start updating the autoencoder using minibatches]
        Sample a minibatch of transitions d from $\mathcal{D}$ with a batchsize k
        Use the s_t of d as input to $\mathbf{A}$
        First encode, then decode d using $\mathbf{A}$
        Calculate the loss L of s_t by comparing the input and output of $\mathbf{A}$
        Update w with o as per L
    end for
end for
$e = e + N_2$
[1]

Algorithm 4 LSTD update phase

Freeze the weights of $f(phi(s))$, i.e. θ
Unfreeze the weights of $\mathbf{A}$, i.e. w
for *episode* $e_0 = 1$ to N_3 **do**
    Repeat the same steps at the warm-up phase
    for each time step $t = 1$ to T **do**
        Repeat the same steps at the warm-up phase
[Start using the online LSTD algorithm to update θ]
        Calculate $\tau = \phi(s_t) - \gamma\phi(s_{t+1})$
        Calculate $v = \tau^T A^{-1}$
        Update $A_a^{-1} = A_a^{-1} - \frac{A_a^{-1}\phi(s)v^T}{1+v\phi(s)}$
        Update $b_a = b_a + r\phi(s)$
        Given the action a of the current time step, update $\theta_a = A_a^{-1}b_a$
        Update state $s_t = s_{t+1}$
    end for
end for
$e = e + N_3$
[1]

Algorithm 5 Training procedure

Run *Initialization*
for each training cycle $c = 1$ to $\frac{N}{N_0}$ **do**
    Run *Warm-up phase*
    for each intra-c round $r = 1$ to $\frac{N_0 - N_1}{N_2 + N_3}$ **do**
        Run *Autoencoder update phase*
        Reset A^{-1}, b, θ to the default value at the initialization phase
        Run *LSTD update phase*
    end for
end for

[1]