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
2: Initialize the learning rate of $\mathbf{A}$ as $\alpha = 1 \times 10^{-3}$ [w_0 are drawn from Glorot uniform initialization]
3: Initialize the minibatch size of DQN as $k = 32$
4: Initialize the total number of training episodes $N = 110$, and the number of episodes at each seperate phase of training $N_0 = 10$ [It would be a wise choice to seperate the warm-up phase, the antoencoder update phase, and the LSTD update phase]
5: [Initialization of training hyperparameters]
6: Initialize the number of maximum time step per episode $T = 5000$
7: Initialize the weights θ_0 for linear approximation function $y = f(phi(s))$ which will be used for LSTD [w_0 are drawn from a zero-mean Gaussian distribution with $\sigma = 1$ and a fixed seed 42]
8: Initialize a replay memory buffer $\mathcal{D}$ with a capacity $N \times T$
9: Initialize a discount factor $\gamma = 0.9$
10: Initialize a small constant $\lambda = 1 \times 10^{-4}$ to initialize A^{-1} for LSTD
11: Add global variables $A^{-1} = \lambda^{-1} I$ and $b = 0$ to store the tensors that suffice $\theta_t = A_t^{-1} b_t$ at each time step t [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* $e = 1$ to N_0 **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 $\epsilon = 0.3$ [The learning curves of Game B of Assignment 1 corroborate that a relatively high ϵ could improve Q more efficiently when an e does not end prematurely in the majority of cases]
        Execute a_t and obtain reward r_t and new state s_{t+1}
        Store the transition of the current t, i.e. (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
        Encode state s_t using $\mathbf{A}$ to get latent embedding $\phi(s_t)$
        Select action a_t using ϵ-greedy policy with $\epsilon = 0.3$ [The learning curves of Game B of Assignment 1 corroborate that a relatively high ϵ could improve Q more efficiently when an e does not end prematurely in the majority of cases]
        Execute a_t and obtain reward r_t and new state s_{t+1}
        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

[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
Initialize an Adam optimizer o with a learning rate α for $\mathbf{A}$
for *episode* $e = 1$ to N_0 **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' of d as input to $\mathbf{A}$
        First encode, then decode d using $\mathbf{A}$
        Calculate the loss L of s' by comparing the input and output of $\mathbf{A}$
        Update w with o as per L
    end for
end for

[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* $e = 1$ to N_0 **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 $A^{-1} = A^{-1} - \frac{A^{-1}\phi(i)v^T}{1+v\phi(s)}$
        Update $b = b + r\phi(s)$
        Update state $s = s'$
    end for
end for

[1]

Algorithm 5 Training procedure

Run *Initialization*
Run *Warm-up phase*
while *episode* $e \leq N$ **do**
    Run *Autoencoder update phase*
    $e = e + N_0$
    Run *LSTD update phase*
    $e = e + N_0$
end while

[1]