Terminal Estimation

sjf

June 07, 2020

In general cases:

$$Q(s,a) = \sum_{s' \neq \widetilde{s}} p(s'|s,a)V(s') + p(\widetilde{s}\,|s,a)V(\widetilde{s})$$

where $\widetilde{s}$ means terminal state
so we can set:

$$\begin{aligned}
\hat{Q}(s_t, a_t; w_q) &= \sum_{s' \neq \widetilde{s}} p(s'|s,a)V(s') \\
f(s_t, a_t; w_t) &= p(\widetilde{s}\,|s,a)V(\widetilde{s}) \\
p_f(s_t, a_t; w_p) &= \sum_{s} p(s'|s,a)\mathbb{I}(s' = \widetilde{s}) \\
Q(s_t, a_t) &= \sum_{s} p(s'|s,a)V(s') = f(s_t, a_t; w_t) * p_f(s_t, a_t) + \hat{Q}(s_t, a_t; w_q) * (1 - p_f(s_t, a_t))
\end{aligned}$$

For terminal estimation function, it can effectively reuse experience buffer because of independence of policy. And terminal prob function trained as DQN.

In network, all output is random and doesn't equal zero. for some cases: if $p(s'|s,a) = 0$, $f(s,a)$ should be zero always and if $p(s'|s,a) = 1$, $\hat{Q}(s,a)$ should be zero always, too.

so for all cases

$$\begin{aligned}
\hat{Q}(s_t, a_t) &= \alpha * \hat{Q}(s_t, a_t) + (1 - \alpha) * (R_{t+1} - \gamma Q(s_{t+1}, a_{t+1})) \\
f(s_t, a_t) &= \beta * f(s_t, a_t) + (1 - \beta) * \widetilde{R}_{t+1}
\end{aligned}$$

It has a shrinking effect in network and correct the former bias. And it works well in experiment. The terminal-estimation are more robust in choose of hyperparameters , especially in weight decay coefficient.

- More precise value estimation.
- Effectively reuse experience buffer.

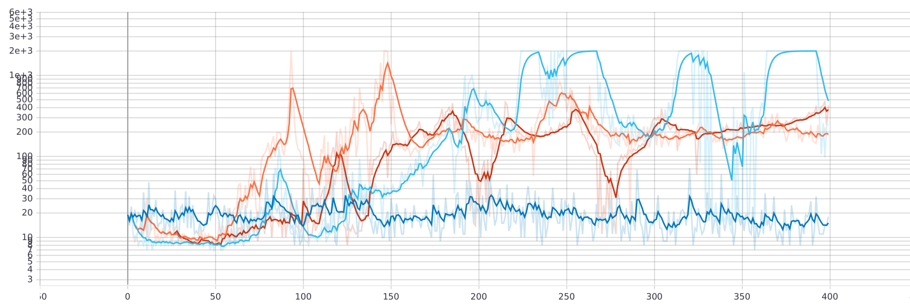

Figure 1: Performance at task CartPole-v0, All of these has the same hyperparameters except weight decay. Bottom deep blue line is baseline with weight decay 1e-3, light blue line is TE AC with weight decay 1e-2 , orange line with weight decay 1e-3 and red line with weight decay 1e-4. It shows that TE AC are more rubust with hyperparameters