FGDQN Implementation for Average Rewards

Tejas Pagare

January 18, 2022

$$
\begin{aligned}
\theta_{n+1} = \theta_n - a(n) \Big(\overline{r(X_n, U_n) + \max_v Q(X_{n+1}, v; \theta_n) - f(Q;\theta_n) - Q(X_n, U_n; \theta_n)} \Big) \times \\
\Big(\nabla_\theta Q(X_{n+1}, v_n; \theta_n) - \nabla_\theta f(Q;\theta_n) - \nabla_\theta Q(X_n, U_n; \theta_n) \Big) \qquad (1)
\end{aligned}
$$

For simplicity assume the current transition is (X, U, R, X').
Now, from the replay buffer sample all the transitions with this fixed state action pair as (X, U). Let's say we get $(X, U, R_1, X'_1), (X, U, R_2, X'_2)$ and (X, U, R_3, X'_3).

Let's denote

$$
\begin{aligned}
target_n &= R_n + \max_v Q(X'_n, v; \theta) - f(Q;\theta) \\
pred_n &= Q(X, U; \theta)
\end{aligned}
$$

Now, we compute $avg\_part = (target_1 - pred_1) + (target_2 - pred_2) + (target_3 - pred_3)$ in Line 137. Here, we have computed avg_part inside the `torch.no_grad()` block which implies PyTorch will not calculate gradient w.r.t the above avg_part tensor automatically after doing `.backward()`.

Now we compute

$$
\begin{aligned}
target &= R + \max_v Q(X', v; \theta_n) - f(Q;\theta) \\
pred &= Q(X, U; \theta)
\end{aligned}
$$

for the current transition (X, U, R, X'). Now, we do $target - pred$, note here this tensor does have `requires_grad` attribute True which implies PyTorch will calculate gradient w.r.t this tensor automatically after doing `.backward()`.

Now, we do `avg_part = avg_part+(target-pred).detach()` in Line 158. This new avg_part tensor have its `requires_grad` attribute False since we have used `.detach()` which essentially makes a new tensor with the same data value but is detached from the computation graph.

Finally we do `avg_part = torch.mean(avg_part)` in Line 160 which is the term average with the overline which is

$$
\overline{r(X_n, U_n) + \max_v Q(X_{n+1}, v; \theta_n) - f(Q;\theta_n) - Q(X_n, U_n; \theta_n)}
$$

where X_n, U_n is X, U and this average term includes $target - pred$ for the iterations $(X, U, R, X'), (X, U, R_1, X'_1), (X, U, R_2, X'_2)$ and (X, U, R_3, X'_3) i.e.

$$
\frac{\Big(R + \max_v Q(X', v; \theta) - f(Q;\theta) - Q(X, U; \theta) \Big) + \Big(R_1 + \max_v Q(X'_1, v; \theta) - f(Q;\theta) - Q(X, U; \theta) \Big) + \cdots}{4}
$$

Now, we finally do `loss = torch.mul(avg_part,target-pred)` in Line 163 which is

$$loss = \frac{\Big(R + \max_v Q(X', v; \theta) - f(Q;\theta) - Q(X, U; \theta)\Big) + \Big(R_1 + \max_v Q(X'_1, v; \theta) - f(Q;\theta) - Q(X, U; \theta)\Big) + \cdots}{4} \times \Big(R + \max_v Q(X', v; \theta) - f(Q;\theta) - Q(X, U; \theta)\Big)$$

Here, first term have `requires_grad` attribute False and the other one has True, since while calculating the first we used `.detach()`.

Now, when we calculate gradient of this w.r.t parameters θ, we get the following,

$$\nabla loss = \frac{\Big(R + \max_v Q(X', v; \theta) - f(Q;\theta) - Q(X, U; \theta)\Big) + \Big(R_1 + \max_v Q(X'_1, v; \theta) - f(Q;\theta) - Q(X, U; \theta)\Big) + \cdots}{4} \times \Big(\nabla Q(X', v; \theta) - \nabla f(Q;\theta) - \nabla Q(X, U; \theta)\Big)$$

This resembles our FGDQN Eq.(1), now when we do `loss.backward()`, we get our FGDQN iterate as follows

$$\theta' \leftarrow \theta - \gamma(\nabla loss)$$

We should note that, our actual loss is MSE Loss calculated in Line 170 using `actual_loss = F.mse_loss(pred,target)`.