Practical Deep Reinforcement Learning

Eilif Solberg

TEK5040/TEK9040

Outline

Section 1

Introduction

Sample efficiency

Sample efficiency

- May be expensive to generate data

Sample efficiency

- May be expensive to generate data
- Can we use data more than once?

Stability

Problem:

Stability

Problem:

- Bad updates impact the data we see

Stability

Problem:

- Bad updates impact the data we see
- Stability is difficult due to changes in distribution of observations and rewards

Stability

Problem:

- Bad updates impact the data we see
- Stability is difficult due to changes in distribution of observations and rewards
- Targets often depend on the output of the network
  - Targets may be changing even if distribution is not, e.g. with TD-learning.
  - State *aliasing* may lead prediction updates to also update target

Stability

Problem:

- Bad updates impact the data we see
- Stability is difficult due to changes in distribution of observations and rewards
- Targets often depend on the output of the network
  - Targets may be changing even if distribution is not, e.g. with TD-learning.
  - State *aliasing* may lead prediction updates to also update target

Goal:

Stability

Problem:

- Bad updates impact the data we see
- Stability is difficult due to changes in distribution of observations and rewards
- Targets often depend on the output of the network
    - Targets may be changing even if distribution is not, e.g. with TD-learning.
    - State *aliasing* may lead prediction updates to also update target

Goal:

- Would like algorithms that *works most of the time*

Stability

Problem:

- Bad updates impact the data we see
- Stability is difficult due to changes in distribution of observations and rewards
- Targets often depend on the output of the network
  - Targets may be changing even if distribution is not, e.g. with TD-learning.
  - State *aliasing* may lead prediction updates to also update target

Goal:

- Would like algorithms that *works most of the time*
- Would like algorithms that work across environments with minimal adjustment of hyperparameters

Stability

Problem:

- Bad updates impact the data we see
- Stability is difficult due to changes in distribution of observations and rewards
- Targets often depend on the output of the network
  - Targets may be changing even if distribution is not, e.g. with TD-learning.
  - State *aliasing* may lead prediction updates to also update target

Goal:

- Would like algorithms that *works most of the time*
- Would like algorithms that work across environments with minimal adjustment of hyperparameters

Will not look at: "Normalizing environments"

Section 2

Deep Q-Networks (DQN)

Papers

- Playing atari with deep reinforcement learning [1].
- Human-level control through deep reinforcement learning [2].
- Deep reinforcement learning with double q-learning [3].

Atari 2600

▸ Atari 2600 spill

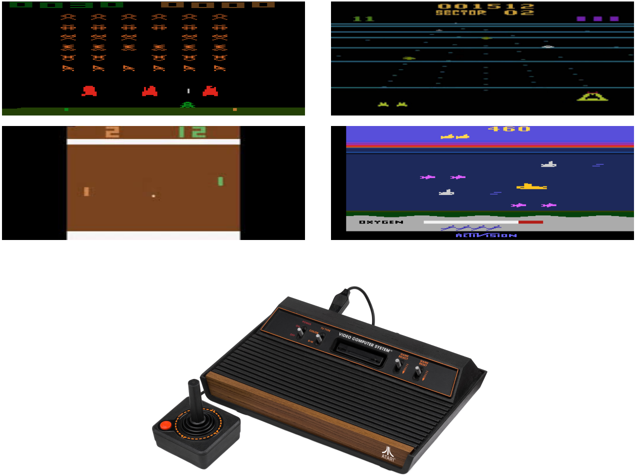

Figure: Atari 2600

Base update

DQN is a q-learning algorithm. We will start with our basic q-learning update and introduce the proposed additions one at a time.

Let q_η be our current estimate of the optimal action-value function q_*. Our *base update* is given by

$$\eta \leftarrow \eta + \alpha\big((r_{t+1} + \gamma \max_{a'} q_\eta(s_{t+1}, a')) - q_\eta(s_t, a_t)\big)\nabla_\eta q_\eta(s_t, a_t)$$

where we call $(s_t, a_t, r_{t+1}, s_{t+1})$ a *transition*.

Batch updates

Store several transitions and make batch update

$$\eta \leftarrow \eta + \alpha \frac{1}{N} \sum_{i=1}^{N} \left((r^{(i)} + \gamma \max_{a'} q_\eta(s'^{(i)}, a')) - q_\eta(s^{(i)}, a^{(i)}) \right) \nabla_\eta q_\eta(s^{(i)}, a^{(i)})$$

- N is our minibatch size and (s, a, r, s') is a transition in an episode

Batch updates

Store several transitions and make batch update

$$\eta \leftarrow \eta + \alpha \frac{1}{N} \sum_{i=1}^{N} \left(\left(r^{(i)} + \gamma \max_{a'} q_\eta(s'^{(i)}, a') \right) - q_\eta(s^{(i)}, a^{(i)}) \right) \nabla_\eta q_\eta(s^{(i)}, a^{(i)}$$

- N is our minibatch size and (s, a, r, s') is a transition in an episode

Motivation:

- Batches often improve stability

Batch updates

Store several transitions and make batch update

$$\eta \leftarrow \eta + \alpha \frac{1}{N} \sum_{i=1}^{N} \left(\left(r^{(i)} + \gamma \max_{a'} q_\eta(s'^{(i)}, a') \right) - q_\eta(s^{(i)}, a^{(i)}) \right) \nabla_\eta q_\eta(s^{(i)}, a^{(i)})$$

- N is our minibatch size and (s, a, r, s') is a transition in an episode

Motivation:

- Batches often improve stability
- Better utilization of GPU

Replay buffer

- Store transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ in *replay buffer* $\mathcal{D}$.
- At each iteration we sample a minibatch from $\mathcal{D}$ which we make updates based on.
- Discard older experience as it becomes out-of-date.

$$\eta \leftarrow \eta + \alpha \frac{1}{N} \sum_{i=1}^{N} \left((r^{(i)} + \gamma \max_{a'} q_\eta(s'^{(i)}, a')) - q_\eta(s^{(i)}, a^{(i)}) \right) \nabla_\eta q_\eta(s^{(i)}, a^{(i)}$$

Now $(s^{(i)}, a^{(i)}, r^{(i)}, s'^{(i)}) \sim \mathcal{D}$, no longer consecutive experience.

Replay buffer

- Store transitions $(s_t, a_t, r_{t+1}, s_{t+1})$ in *replay buffer* $\mathcal{D}$.
- At each iteration we sample a minibatch from $\mathcal{D}$ which we make updates based on.
- Discard older experience as it becomes out-of-date.

$$\eta \leftarrow \eta + \alpha \frac{1}{N} \sum_{i=1}^{N} \left((r^{(i)} + \gamma \max_{a'} q_\eta(s'^{(i)}, a')) - q_\eta(s^{(i)}, a^{(i)}) \right) \nabla_\eta q_\eta(s^{(i)}, a^{(i)})$$

Now $(s^{(i)}, a^{(i)}, r^{(i)}, s'^{(i)}) \sim \mathcal{D}$, no longer consecutive experience. Serves two purposes:

- Sample efficiency: Several updates from the same experience
- Stability: Get less correlated data sampling from a larger dataset

"Fixed" target Q-network

Problem: Risk of state *aliasing* when using function approximators.

- Features q_η extracts from consecutive states s and s' may be almost identical.

"Fixed" target Q-network

Problem: Risk of state *aliasing* when using function approximators.

- Features q_η extracts from consecutive states s and s' may be almost identical.
- Recall prediction and targets are of the form

$$q_\eta(s, a), \qquad r + \gamma \max_{a'} q_\eta(s', a')$$

Updating $q(s, a)$ may affect $q(s', a')$ for different actions a'.

- Targets are moving - may end up chasing our own tail.

"Fixed" target Q-network

Problem: Risk of state *aliasing* when using function approximators.

- Features q_η extracts from consecutive states s and s' may be almost identical.
- Recall prediction and targets are of the form

$$q_\eta(s,a), \qquad r + \gamma \max_{a'} q_\eta(s',a')$$

Updating $q(s,a)$ may affect $q(s',a')$ for different actions a'.

- Targets are moving - may end up chasing our own tail.

Solution:

- Keep a separate target network q_{η^-}

"Fixed" target Q-network

Problem: Risk of state *aliasing* when using function approximators.

- Features q_η extracts from consecutive states s and s' may be almost identical.
- Recall prediction and targets are of the form

$$q_\eta(s, a), \qquad r + \gamma \max_{a'} q_\eta(s', a')$$

Updating $q(s, a)$ may affect $q(s', a')$ for different actions a'.

- Targets are moving - may end up chasing our own tail.

Solution:

- Keep a separate target network q_{η^-}
- Only occasionally update η^- to match η

"Fixed" target Q-network

Problem: Risk of state *aliasing* when using function approximators.

- Features q_η extracts from consecutive states s and s' may be almost identical.
- Recall prediction and targets are of the form

$$q_\eta(s, a), \qquad r + \gamma \max_{a'} q_\eta(s', a')$$

Updating $q(s, a)$ may affect $q(s', a')$ for different actions a'.

- Targets are moving - may end up chasing our own tail.

Solution:

- Keep a separate target network q_{η^-}
- Only occasionally update η^- to match η

$$\eta \leftarrow \eta + \alpha \frac{1}{N} \sum_{i=1}^{N} \left((r^{(i)} + \gamma \max_{a'} q_{\eta^-}(s'^{(i)}, a')) - q_\eta(s^{(i)}, a^{(i)}) \right) \nabla_\eta q_\eta(s^{(i)}, a^{(

Bias-reduction of Q

Problem: Targets are too optimistic

- Value estimate is $\max_a q_\eta(s,a) = q_\eta(s, \mathrm{argmax}_a q_\eta(s,a))$.

Bias-reduction of Q

Problem: Targets are too optimistic

- Value estimate is $\max_a q_\eta(s,a) = q_\eta(s, \text{argmax}_a q_\eta(s,a))$.
- The reason that an action is chosen, is often because it is too optimistic! (Winner's curse)
- For a state s assume $q_\pi(s,a)$ are zero for all a, and assume we have an equal number of values $q_\eta(s,a)$ that are positive and negative. Then $q_\eta(s, \text{argmax}_a q_\eta(s,a)) > 0$.
- So if $a' = \text{argmax}_a q_\eta(s,a)$. Often
  - $q_\eta(s,a') > q_\pi(s,a')$, even
  - $q_\eta(s,a') > q_\pi(s, \text{argmax}_a q_\pi(s,a))$

Bias-reduction of Q

Problem: Targets are too optimistic

- Value estimate is $\max_a q_\eta(s,a) = q_\eta(s, \text{argmax}_a q_\eta(s,a))$.
- The reason that an action is chosen, is often because it is too optimistic! (Winner's curse)
- For a state s assume $q_\pi(s,a)$ are zero for all a, and assume we have an equal number of values $q_\eta(s,a)$ that are positive and negative. Then $q_\eta(s, \text{argmax}_a q_\eta(s,a)) > 0$.
- So if $a' = \text{argmax}_a q_\eta(s,a)$. Often
  - $q_\eta(s,a') > q_\pi(s,a')$, even
  - $q_\eta(s,a') > q_\pi(s, \text{argmax}_a q_\pi(s,a))$
- Note: Happens even though $q_\eta(s,a)$ is not too optimistic in general.
- This is not just a problem to function approximation, but q-learning in general.

Bias-reduction of Q II

Solution:

Bias-reduction of Q II

Solution:

- Choose the action from our current policy network q_η

Bias-reduction of Q II

Solution:

- Choose the action from our current policy network q_η
- Still get value from evaluating target network q_{η^-}.

Bias-reduction of Q II

Solution:

- Choose the action from our current policy network q_η
- Still get value from evaluating target network q_{η^-}.

$$a'^{(i)} = \text{argmax}_a q_\eta(s'^{(i)}, a)$$

$$\eta \leftarrow \eta + \alpha \frac{1}{N} \sum_{i=1}^{N} \left((r^{(i)} + \gamma q_{\eta^-}(s'^{(i)}, a'^{(i)})) - q_\eta(s^{(i)}, a^{(i)}) \right) \nabla_\eta q_\eta(s^{(i)}, a^{(i)})$$

Pseudocode

Algorithm 1 Deep Q-learning with Experience Replay

Initialize (round-robin) replay memory $\mathcal{D}$ (partially) up to capacity N
Initialize action-value function q_η with random weights.
Initialize target action-value function $q_{\eta-}$ with weights $\eta^- = \eta$.
Let h_t denote the history so far $(o_0, a_0, r_1, o_1, \dots, r_t, o_t)$.
for episode $= 1, M$ **do**
  Initialize sequence with $s_0 = f(o_0)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a q_\eta(s_t, a)$
    Execute action a_t in emulator and observe reward r_{t+1} and observation o_{t+1}
    Set $s_{t+1} = f(h_{t+1})$
    Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $\mathcal{D}$.
    Sample random minibatch of transitions $\left(s_j, a_j, r_{j+1}, s_{j+1}\right)$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_{j+1} & \text{for terminal } s_{j+1} \\ r_{j+1} + \gamma q_{\eta^-}(s_{j+1}, \operatorname{argmax}_{a'} q_\eta(s_{j+1}, a')) & \text{for non-terminal } s_{j+1} \end{cases}$
    Perform a gradient descent step on $\left(y_j - q_\eta(s_j, a_j)\right)^2$ with respect to the network parameters η.
    Every C steps, set $\eta^- = \eta$.
  end for
end for

Section 3

Proximal Policy Optimization (PPO)

Papers

- Trust region policy optimization [4].
- Proximal policy optimization algorithms [5].

Advantage

We define the *advantage* function d_π as

$$d_\pi(s,a) := q_\pi(s,a) - v_\pi(s)$$

Note that for a given state s the expected advantage is always 0

$$\begin{aligned} E_{A\sim\pi(s)}[d_\pi(s,A)] &= E_{A\sim\pi(s)}[q_\pi(s,A) - v_\pi(s)] \\ &= E_{A\sim\pi(s)}[q_\pi(s,A)] - v_\pi(s) \\ &= \sum_a \pi(a|s) q_\pi(s,a) - v_\pi(s) \\ &= v_\pi(s) - v_\pi(s) = 0 \end{aligned}$$

Possible approximatations are e.g.

- $G_t - v_\eta(s_t)$
- $R_{t+1} + \gamma v_\eta(S_{t+1}) - v_\eta(s_t)$
- $q_\nu(s_t, a_t) - v_\eta(s_t)$

Actor-critic

Policy-gradient update:

$$\theta \leftarrow \theta + \alpha \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\tau^{(i)}-1} g_t^{(i)} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Actor-critic

Policy-gradient update:

$$\theta \leftarrow \theta + \alpha \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\tau^{(i)}-1} g_t^{(i)} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Actor-critic update:

$$\theta \leftarrow \theta + \alpha \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\tau^{(i)}-1} \hat{d}_t^{(i)} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Actor-critic

Policy-gradient update:

$$\theta \leftarrow \theta + \alpha \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\tau^{(i)}-1} g_t^{(i)} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

Actor-critic update:

$$\theta \leftarrow \theta + \alpha \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\tau^{(i)}-1} \hat{d}_t^{(i)} \nabla_\theta \log \pi_\theta(a_t^{(i)} | s_t^{(i)})$$

- $\hat{d}_t \approx q_{\pi_\theta}(s_t, a_t) - v_{\pi_\theta}(s_t)$, i.e. estimation of *advantage* of taking action a_t from state s_t.

Returns of a policy in terms of another

For two policies π and $\tilde{\pi}$

$$E_{\tilde{\pi}}[G_0] = E_{\pi}[G_0] + E_{\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t d_{\pi}(S_t, A_t)]$$

Returns of a policy in terms of another

For two policies π and $\tilde{\pi}$

$$E_{\tilde{\pi}}[G_0] = E_{\pi}[G_0] + E_{\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t d_{\pi}(S_t, A_t)]$$

- G_0: return of the episode, i.e. $G_0 = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$.

Returns of a policy in terms of another

For two policies π and $\tilde{\pi}$

$$E_{\tilde{\pi}}[G_0] = E_{\pi}[G_0] + E_{\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t d_{\pi}(S_t, A_t)]$$

- G_0: return of the episode, i.e. $G_0 = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$.
- Optimize left-hand side by optimizing $E_{\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t d_{\pi}(S_t, A_t)]$ with respect to $\tilde{\pi}$.

Returns of a policy in terms of another

For two policies π and $\tilde{\pi}$

$$E_{\tilde{\pi}}[G_0] = E_{\pi}[G_0] + E_{\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t d_{\pi}(S_t, A_t)]$$

- G_0: return of the episode, i.e. $G_0 = \sum_{t=0}^{\infty} \gamma^t R_{t+1}$.
- Optimize left-hand side by optimizing $E_{\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t d_{\pi}(S_t, A_t)]$ with respect to $\tilde{\pi}$.
- Or? We will rewrite and simplify problem.

Visitation frequencees

- Assume discrete state and action spaces

Visitation frequencees

- Assume discrete state and action spaces
- Let ρ_π be the unnormalized discounted visitation frequencies

$$\rho_\pi(s) = P_\pi(S_0 = s) + \gamma P_\pi(S_1 = s) + \gamma^2 P_\pi(S_2 = s) + \dots$$

Visitation frequencees

- Assume discrete state and action spaces
- Let ρ_π be the unnormalized discounted visitation frequencies

$$\rho_\pi(s) = P_\pi(S_0 = s) + \gamma P_\pi(S_1 = s) + \gamma^2 P_\pi(S_2 = s) + \ldots$$

- $S_0 \sim \rho_0$
- Actions are chosen according to π.

Visitation frequencees

- Assume discrete state and action spaces
- Let ρ_π be the unnormalized discounted visitation frequencies

$$\rho_\pi(s) = P_\pi(S_0 = s) + \gamma P_\pi(S_1 = s) + \gamma^2 P_\pi(S_2 = s) + \dots$$

- $S_0 \sim \rho_0$
- Actions are chosen according to π.
- This function often also called (discounted) occupancy measure.

Rewrite objective

$$
\begin{aligned}
E_{\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t d_\pi(S_t, A_t)] &= \sum_{t=0}^{\infty} \sum_{s} \sum_{a} P_{\tilde{\pi}}(S_t = s, A_t = a) \gamma^t d_\pi(s, a) \\
&= \sum_{t=0}^{\infty} \sum_{s} \sum_{a} P_{\tilde{\pi}}(S_t = s) \tilde{\pi}(a|s) \gamma^t d_\pi(s, a) \\
&= \sum_{t=0}^{\infty} \sum_{s} P_{\tilde{\pi}}(S_t = s) \sum_{a} \tilde{\pi}(a|s) \gamma^t d_\pi(s, a) \\
&= \sum_{s} \sum_{t=0}^{\infty} \gamma^t P_{\tilde{\pi}}(S_t = s) \sum_{a} \tilde{\pi}(a|s) d_\pi(s, a) \\
&= \sum_{s} \rho_{\tilde{\pi}}(s) \sum_{a} \tilde{\pi}(a|s) d_\pi(s, a)
\end{aligned}
$$

Rewrite objective

$$
\begin{aligned}
E_{\tilde{\pi}}[\sum_{t=0}^{\infty} \gamma^t d_\pi(S_t, A_t)] &= \sum_{t=0}^{\infty} \sum_s \sum_a P_{\tilde{\pi}}(S_t = s, A_t = a) \gamma^t d_\pi(s, a) \\
&= \sum_{t=0}^{\infty} \sum_s \sum_a P_{\tilde{\pi}}(S_t = s) \tilde{\pi}(a|s) \gamma^t d_\pi(s, a) \\
&= \sum_{t=0}^{\infty} \sum_s P_{\tilde{\pi}}(S_t = s) \sum_a \tilde{\pi}(a|s) \gamma^t d_\pi(s, a) \\
&= \sum_s \sum_{t=0}^{\infty} \gamma^t P_{\tilde{\pi}}(S_t = s) \sum_a \tilde{\pi}(a|s) d_\pi(s, a) \\
&= \sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) d_\pi(s, a)
\end{aligned}
$$

Increasing $\tilde{\pi}(a|s)$ for positive advantages $d_\pi(s, a)$ leads to improvement?

Policy iteration revisited

If for all states s

$$\sum_a \tilde{\pi}(a|s) d_\pi(s,a) >= 0$$

we are indeed guranteed that $\tilde{\pi} \geq \pi$.

Policy iteration revisited

If for all states s

$$\sum_a \tilde{\pi}(a|s) d_\pi(s,a) >= 0$$

we are indeed guranteed that $\tilde{\pi} \geq \pi$. Note that our derivations imply the policy iteration theorem, where we defined our new policy as

$$\tilde{\pi}(s) := \text{argmax}_a q_\pi(s,a) = \text{argmax}_a d_\pi(s,a)$$

Policy iteration revisited

If for all states s

$$\sum_a \tilde{\pi}(a|s) d_\pi(s,a) >= 0$$

we are indeed guranteed that $\tilde{\pi} \geq \pi$. Note that our derivations imply the policy iteration theorem, where we defined our new policy as

$$\tilde{\pi}(s) := \text{argmax}_a q_\pi(s,a) = \text{argmax}_a d_\pi(s,a)$$

We will here look at *stochastic parametrized* families of policies $\pi_\theta, \theta \in \Theta$.

Ignoring change in state-visitation frequencies

Optimizing

$$\sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) d_\pi(s, a)$$

is too difficult due to complex effect of change in state-visitation frequences.

Ignoring change in state-visitation frequencies

Optimizing

$$\sum_s \rho_{\tilde{\pi}}(s) \sum_a \tilde{\pi}(a|s) d_\pi(s,a)$$

is too difficult due to complex effect of change in state-visitation frequences. Thus we define the simpler function

$$L(\tilde{\pi}) = \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) d_\pi(s,a)$$

Optimizable II

$$
\begin{aligned}
L(\tilde{\pi}) &:= \sum_s \rho_\pi(s) \sum_a \tilde{\pi}(a|s) d_\pi(s,a) \\
&= \sum_s \sum_{t=0}^{\infty} \gamma^t P_\pi(S_t = s) \sum_a \tilde{\pi}(a|s) d_\pi(s,a) \\
&= \sum_{t=0}^{\infty} \sum_s P_\pi(S_t = s) \sum_a \tilde{\pi}(a|s) \gamma^t d_\pi(s,a) \\
&= \sum_{t=0}^{\infty} \sum_s P_\pi(S_t = s) \sum_a \pi(a|s) \frac{\tilde{\pi}(a|s)}{\pi(a|s)} \gamma^t d_\pi(s,a) \\
&= \sum_{t=0}^{\infty} \sum_s \sum_a P_\pi(S_t = s) \pi(a|s) \frac{\tilde{\pi}(a|s)}{\pi(a|s)} \gamma^t d_\pi(s,a) \\
&= E_\pi\left[\sum_{t=0}^{\infty} \frac{\tilde{\pi}(A_t|S_t)}{\pi(A_t|S_t)} \gamma^t d_\pi(S_t, A_t) \right]
\end{aligned}
$$

Approximation

Can we optimize $L(\tilde{\pi})$?

Approximation

Can we optimize $L(\tilde{\pi})$?

Approximate with sample

$$L(\tilde{\pi}) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\tau^{(i)}-1} \frac{\tilde{\pi}(a_t^{(i)}|s_t^{(i)})}{\pi(a_t^{(i)}|s_t^{(i)})} \gamma^t \hat{d}_t^{(i)}$$

Approximation

Can we optimize $L(\tilde{\pi})$?

Approximate with sample

$$L(\tilde{\pi}) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\tau^{(i)}-1} \frac{\tilde{\pi}(a_t^{(i)}|s_t^{(i)})}{\pi(a_t^{(i)}|s_t^{(i)})} \gamma^t \hat{d}_t^{(i)}$$

- Let $\hat{E}$ denote the empirical distribution, then the problem may be restated as

$$\max_{\tilde{\pi}} \hat{E}\left[\frac{\tilde{\pi}(a_t|s_t)}{\pi(a_t|s_t)} \gamma^t \hat{d}_t\right]$$

Approximation

Can we optimize $L(\tilde{\pi})$?
Approximate with sample

$$L(\tilde{\pi}) \approx \frac{1}{N} \sum_{i=1}^{N} \sum_{t=0}^{\tau^{(i)}-1} \frac{\tilde{\pi}(a_t^{(i)} | s_t^{(i)})}{\pi(a_t^{(i)} | s_t^{(i)})} \gamma^t \hat{d}_t^{(i)}$$

- Let $\hat{E}$ denote the empirical distribution, then the problem may be restated as

$$\max_{\tilde{\pi}} \hat{E}\left[\frac{\tilde{\pi}(a_t | s_t)}{\pi(a_t | s_t)} \gamma^t \hat{d}_t\right]$$

- Note: We have ignored the factor $\sum_{i=1}^{N} \tau^{(i)}/N$, as it does not affect solution.
- Note: Going forward we will ignore the factor γ^t as well. Might argue that we care equally about $E_{\tilde{\pi}}[G_t]$ for any t rather than just $E_{\tilde{\pi}}[G_0]$. γ still influences solution through the return.

Conservative policy updates

- Don't change policy too much as we are only approximating.
- Sample new "dataset" regularly
  - Policy iteration algorithm

PPO - objective

$\pi_\theta, \ \theta \in \Theta$. Let θ_{old} be the parameters of the policy we have sampled from.
Define

$$u_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

PPO - objective

$\pi_\theta, \ \theta \in \Theta$. Let θ_{old} be the parameters of the policy we have sampled from.
Define

$$u_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$$

Let $\text{clip}(x, \text{lower}, \text{upper}) := \min(\max(x, \text{lower}), \text{upper})$, then define the surrogate objective as

$$L^{PPO}(\theta) = \hat{E}[\min(u_t(\theta)\hat{d}_t, \text{clip}(u_t(\theta), 1-\epsilon, 1+\epsilon)\hat{d}_t)]$$

where ϵ is a hyperparameter, e.g. $\epsilon = 0.2$.

PPO - intuition

$$L^{PPO}(\theta) = \hat{E}[\min(u_t(\theta)\hat{d}_t, \text{clip}(u_t(\theta), 1-\epsilon, 1+\epsilon)\hat{d}_t)]$$

where ϵ is a hyperparameter, e.g. $\epsilon = 0.2$.

PPO - intuition

$$L^{PPO}(\theta) = \hat{E}[\min(u_t(\theta)\hat{d}_t, \text{clip}(u_t(\theta), 1-\epsilon, 1+\epsilon)\hat{d}_t)]$$

where ϵ is a hyperparameter, e.g. $\epsilon = 0.2$.

- The first term is the same as our surrogate objective from above

PPO - intuition

$$L^{PPO}(\theta) = \hat{E}[\min(u_t(\theta)\hat{d}_t, \text{clip}(u_t(\theta), 1-\epsilon, 1+\epsilon)\hat{d}_t)]$$

where ϵ is a hyperparameter, e.g. $\epsilon = 0.2$.

- The first term is the same as our surrogate objective from above
- The second term removes incentive to move too far away from $\pi_{\theta_{\text{old}}}$.

PPO - intuition

$$L^{PPO}(\theta) = \hat{E}[\min(u_t(\theta)\hat{d}_t, \text{clip}(u_t(\theta), 1-\epsilon, 1+\epsilon)\hat{d}_t)]$$

where ϵ is a hyperparameter, e.g. $\epsilon = 0.2$.

- The first term is the same as our surrogate objective from above
- The second term removes incentive to move too far away from $\pi_{\theta_{\text{old}}}$.
- Take minumum to get *pessimistic* bound

Policy evaluation

- So far looked at policy *improvement* step. Need policy *evaluation* as well.

Policy evaluation

- So far looked at policy *improvement* step. Need policy *evaluation* as well.
- May use any of the techniques we have learned for estimation of value functions.

Policy evaluation

- So far looked at policy *improvement* step. Need policy *evaluation* as well.
- May use any of the techniques we have learned for estimation of value functions.

As an example may fit value function v_η by e.g. minimizing loss

$$l(\eta) = \frac{1}{2}(g_t - v_\eta(s_t))^2$$

Simultaneous policy evaluation and improvement

To be able to share parameters between value function and policy function, we may combine policy evaluation and policy improvement steps, at each step optimizing

$$L = \hat{E}[L_t^{PPO}(\theta) - c\,(g_t - v_\eta(s_t))^2]$$

- L_t^{PPO} is an element in L^{PPO}.
- $c > 0$ is a hyperparameter.

Simultaneous policy evaluation and improvement

To be able to share parameters between value function and policy function, we may combine policy evaluation and policy improvement steps, at each step optimizing

$$L = \hat{E}[L_t^{PPO}(\theta) - c\,(g_t - v_\eta(s_t))^2]$$

- L_t^{PPO} is an element in L^{PPO}.
- $c > 0$ is a hyperparameter.
- Differentiate L both with respect to η and θ.
- η and θ may now actually overlap.

Pseudocode

Algorithm 2 PPO, Actor-Critic Style

Initialize value network v_η with random weights.
Initialize policy network π_θ with random weights.
Initialize $\theta_{\text{old}} = \theta$.
for iteration $= 1, 2, \dots$ **do**
  for i $= 1, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment (possibly limit time steps)
    Compute advantage estimates $\hat{d}_1, \dots, \hat{d}_{\tau^{(i)}}$
  end for
  Set surrogate objective L based on the sampled data.
  Optimize surrogate L wrt. η and θ, for K epochs and minibatch size $M \leq \sum_{i=1}^{N} \tau^{(i)}$.
  $\theta_{\text{old}} \leftarrow \theta$.
end for

Section 4

Bibliography

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller.
Playing atari with deep reinforcement learning.
arXiv preprint arXiv:1312.5602, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al.
Human-level control through deep reinforcement learning.
Nature, 518(7540):529, 2015.

Hado Van Hasselt, Arthur Guez, and David Silver.
Deep reinforcement learning with double q-learning.
In *Thirtieth AAAI conference on artificial intelligence*, 2016.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz.
Trust region policy optimization.

In *International conference on machine learning*, pages 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov.
Proximal policy optimization algorithms.
arXiv preprint arXiv:1707.06347, 2017.