Generic form of the IRPO algorithm applied to several learning control tasks

This document presents generic implementation of the IRPO algorithm applied to several learning control tasks. The algorithm is introduced in [1]. Without knowing this paper, the present document is unreadable.

1 The algorithm

1. Draw the control action \( a_t \):

\[
a_t \sim \varphi(\cdot; \tilde{\theta}(s_t; w_\theta))\]

2. Perform the control action \( a_t \), and observe the next state \( s_{t+1} \) and the reinforcement \( r_{t+1} \).

3. Add the five \( \langle s_t, a_t, r_{t+1}, s_{t+1}, \varphi_t \rangle \) to the dataset \( D \) where \( \varphi_t := \varphi(a_t; \tilde{\theta}(s_t; w_\theta)) \).

4. (The internal loop) Repeat \( \min\{n_1 \# D, n_\infty \} \) times:

   (a) Calculate \( d_i = r_{t+i} - \tilde{V}(s_{t+i}; w_V) \) if \( s_{t+i} \) was not acceptable and \( d_i = r_{t+i+1} + \gamma \tilde{V}(s_{t+i+1}; w_V) - \tilde{V}(s_{t+i}; w_V) \) otherwise.

   (b) Adjust the approximation of the value function:

\[
w_V := w_V + \beta_i d_i \rho(\varphi(a_t; \tilde{\theta}(s_t; w_\theta)), \varphi_t) \times \frac{d\tilde{V}(s_t; w_V)}{dw_V}
\]

   (c) Adjust the policy:

\[
g_i = G \left( \frac{d\rho(\varphi(a_t; \tilde{\theta}(s_t; w_\theta)), \varphi_t)}{d\tilde{\theta}(s_t; w_\theta)} \right. \left. \tilde{\theta}(s_t; w_\theta) \right) \]

\[
w_\theta := w_\theta + \beta_i d_i \frac{d\tilde{\theta}(s_t; w_\theta)}{dw_\theta} g_i
\]

   (d) Set \( i := i + 1 \).

5. Set \( t := t + 1 \) and repeat from Step 1.
2 Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\varphi$</td>
<td>smoothly-exploring distribution</td>
</tr>
<tr>
<td>$M^\theta$</td>
<td>number of hidden neurons of $\theta$</td>
</tr>
<tr>
<td>$M^V$</td>
<td>number of hidden neurons of $V$</td>
</tr>
<tr>
<td>$\beta_i^\theta$</td>
<td>learning rate of $\theta$</td>
</tr>
<tr>
<td>$\beta_i^V$</td>
<td>learning rate of $V$</td>
</tr>
<tr>
<td>$b$</td>
<td>the upper bound of $\rho$</td>
</tr>
<tr>
<td>$n_\infty$</td>
<td>max. absolute intensity of the internal loop,</td>
</tr>
<tr>
<td>$n_1$</td>
<td>max. intensity of the internal loop relative to the current cardinality $#D$ of the dataset $D$.</td>
</tr>
</tbody>
</table>

The sequence $\{t_i\}_{i \in \mathbb{N}}$ is a concatenation of random permutations of time indexes available in the dataset $D$ (it is a “better” form of simple drawing a random sample from the data). The typical form of $G$ function is as follows

$$G_j(g, \theta) = \begin{cases} 
  g_j & \text{if } \theta_j \text{ is appropriate} \\
  1 & \text{if } \theta_j \text{ is too small} \\
  -1 & \text{if } \theta_j \text{ is too large}
\end{cases}$$

and $\rho$ function takes the form

$$\rho(d, d_0) = \min \left\{ \frac{d}{d_0}, b \right\}$$

where $b$ is a small number larger than 1 e.g. 5. We use the largest value of $n_\infty$ still enabling for our simulations (PC with AthlonTM 1400 MHz) to be carried in real time of the plant. While the plant is really simulated and $n_\infty$ of any size may be set, we want to check IRPO’s behavior when run on a common PC and confronted with a real plant.

References