,

Hyp-RL

Hyperparameter optimization with reinforcement learning

Bartłomiej Sobieski

Faculty of Mathematics and Information Science
Warsaw University of Technology

May 23, 2022

Overview

1. What is reinforcement learning?
2. Q-learning
3. Reinforcement learning and hyperparameter optimization
4. Our approach

What is reinforcement learning?

How do children learn to walk?

Intuitive understanding

An **agent** lives in the **environment** that, at a given moment, can be described by its **state**. By taking an **action** in this environment , the agent receives a **reward** and changes the state of the environment.

Intuitive understanding

- Environment
- Agent
- States
- Actions
- Rewards

The agent's goal is to maximize the cumulative rewards it receives over time.

Markov decision process

A Markov decision process is characterized by:

- **S** - set of states
- **A** - set of actions
- **R** - set of rewards

For timestep $t = 0, 1, 2, \ldots$

$$S_t \in \mathsf{S} \rightarrow A_t \in \mathsf{A} \rightarrow (S_t, A_t) \xrightarrow{\text{t+1}} f(S_t, A_t) = R_{t+1},\ S_{t+1} \in \mathsf{S}$$

Return and discounted return

Return:

$$G_t = R_{t+1} + R_{t+2} + ... + R_T$$

Discounted return:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \text{ for } \gamma \in (0, 1)$$

The agent's goal is to maximize the expected discounted return of rewards.

How does an agent act?

An agent acts according to a **policy** π.

$$\pi\left(A_t | S_t\right) = \mathbb{P}\left(A_t | S_t\right)$$

Q-learning

State values

$$v_\pi(S_t = s) = E_\pi\left(G_t \mid S_t = s\right)$$

Q-value function

$$Q_\pi(S_t = s, A_t = a) = E_\pi \left(G_t \mid S_t = s, A_t = a\right) =$$

$$E_\pi \left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a \right)$$

Optimal policy

$$\pi \geq \pi' \Leftrightarrow \forall_{s \in S}\ v_\pi(s) \geq v_{\pi'}(s)$$

$$\pi \text{ is an optimal policy} \Leftrightarrow \forall_{\pi'}\ \pi \geq \pi'$$

Bellman optimality equation

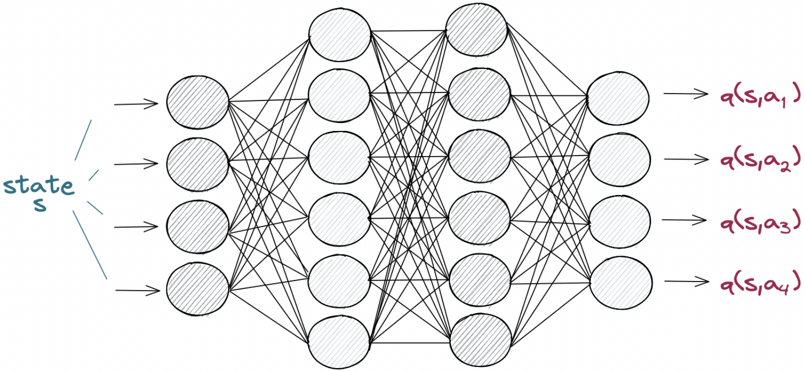
For an optimal policy π^*

$$Q_{\pi^*}(s,a) = \max_{\pi} Q_{\pi}(s,a).$$

Bellman optimality equation:

$$Q_{\pi^*}(s,a) = E\left(R_{t+1} + \gamma \max_{a'} Q_{\pi^*}(s',a')\right).$$

Deep Q-learning



Policy network

Deep Q-learning

$$Loss = Q_{\pi^*}(s, a) - Q(s, a)$$

$$\Leftrightarrow$$

$$Loss = E\left(R_{t+1} + \gamma \max_{a'} Q_{\pi^*}(s', a')\right) - E\left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right)$$

Target network

$$Loss = E\left(R_{t+1} + \gamma \max_{a'} Q_{\pi^*}(s', a')\right) - E\left(\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}\right)$$

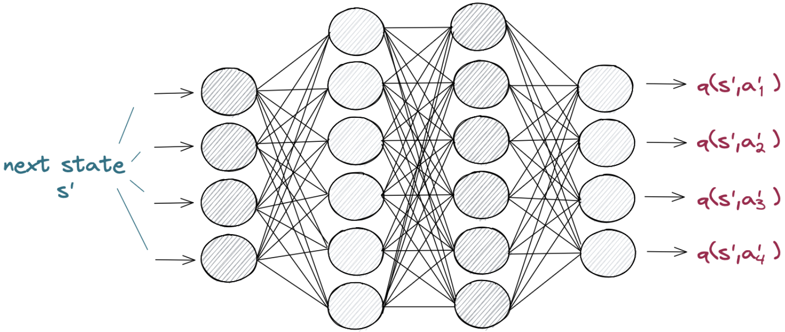
$$\downarrow$$

$$\max_{a'} Q_{\pi^*}(s', a')$$

$$\uparrow$$

Target network

Target network



Target network

Collecting experience

$$e_t = (s_t, a_t, r_{t+1}, s_{t+1})$$

$$\text{Replay buffer} = \{e_t : t \in T - \text{capacity}, ..., T\},$$

where T is the current timestep number.

Exploration vs. exploitation

$$U \sim \mathcal{U}([0,1])$$

$$\epsilon = \epsilon_{end} + \left(\epsilon_{start} - \epsilon_{end}\right) \cdot e^{-t \cdot \epsilon_{decay}}$$

$$\epsilon < U \rightarrow \text{ action chosen randomly.}$$

$$\epsilon > U \rightarrow \text{ action chosen by the agent.}$$

Reinforcement learning and hyperparameter optimization

Problem overview

$$M_\lambda \left(D_{train}\right)$$

$$\lambda \in \Lambda = \Lambda_1 \times \ldots \times \Lambda_P$$

$$\lambda^* = \underset{\lambda \in \Lambda}{\arg\min}\, \mathcal{L}\left(M_\lambda \left(D_{train}\right), D_{valid}\right)$$

Find the best hyperparameter configuration for a given dataset.

Hyp-RL

$$\mathcal{D} = \{D^1, ..., D^m\}$$

$$\mathsf{A} = \Lambda$$

$$\mathsf{R} = \left\{\mathsf{f}\left(M_\lambda(D^i_{valid})\right) : \lambda \in \Lambda,\ i = 1, ..., m\ \right\}$$

$$\mathsf{S} \ni s = \left(s_{static}, s_{dynamic}\right)$$

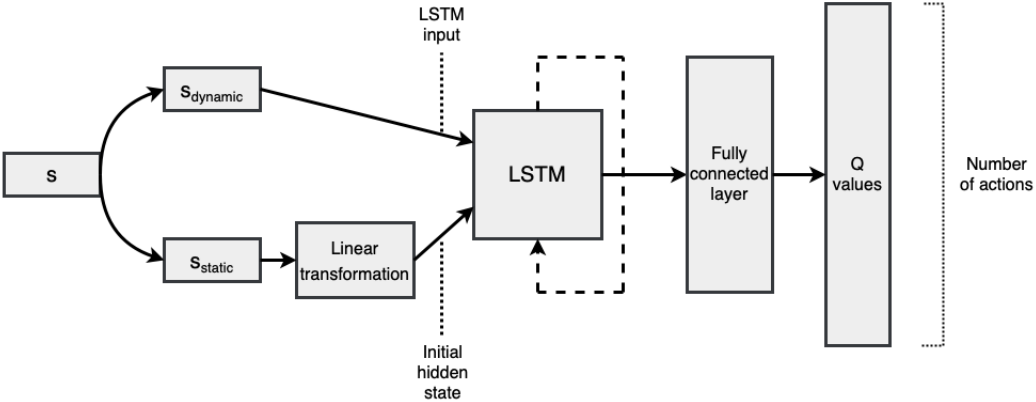
State decomposition

$$s = (s_{static}, s_{dynamic})$$

$$s_{static} \in \{\text{metadata}(D^i) : i = 1, \ldots, m\}$$

$$s_{dynamic} \in (\Lambda \times R)^t \ \text{ for t = 1,...,T}$$

Policy network architecture



Algorithm

Algorithm Hyp-RL

Input: $\mathcal{D}$ - set of datasets, Λ - hyperparameter grid, γ - discount factor, N_{target} - target update frequency, N_{replay} - replay buffer capacity, N_e - number of episodes per dataset, T - number of actions per episode.

Initialize policy network Q_{policy} parameters randomly and make Q_{target} as its clone, create replay buffer $\mathcal{B} = \emptyset$.

for $N_e \cdot |\mathcal{D}|$ **do**

  Choose dataset D^i randomly

  $s_t = \left(s_{static}(D^i), s_{dynamic} = (\{0\}^{P+1}\right)$

  for $t = 0, ..., T$ and while s_t is not terminal **do**

    Determine next action as $a_t = \begin{cases} \sim \mathcal{U}(\Lambda) & p \sim \mathcal{U}([0,1]) < \epsilon \\ arg\max\limits_{a}\ Q_{policy}(s_t, a) & \text{otherwise} \end{cases}$

    Receive reward $r_t = f(M_{\lambda=a_t}(D^i_{valid}))$

    Generate new state $s_{t+1} = s_t \cup \{(\lambda = a_t, r_t)\}$

    Store new experience: $\mathcal{B} = \mathcal{B} \cup \{(s_t, s_{t+1}, a_t, r_t)\}$, replace oldest element if $|\mathcal{B}| > N_{replay}$

    Sample a batch B of experiences from the replay buffer $\mathcal{B}$ and relabel it as

      $B = \{(s, a, Q(s, s', a, r)) \mid (s, s', a, r) \sim \mathcal{U}(\mathcal{B})\}$, where

      $Q(s, s', s, r) = \begin{cases} r & s' \text{ is terminal} \\ r + \gamma \max\limits_{a'} Q_{target}(s', a') & \text{otherwise} \end{cases}$

    Update Q_{policy} by minimizing

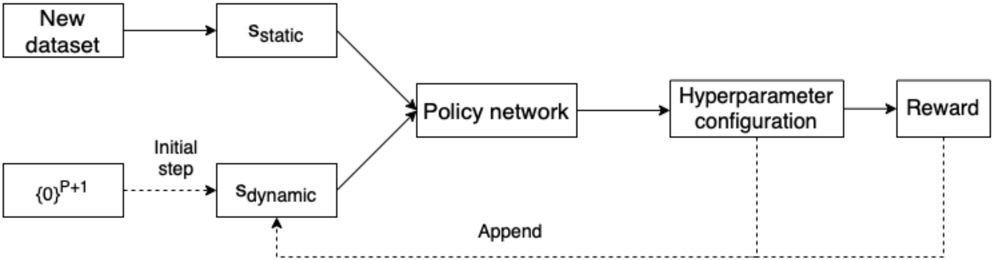
$$\sum_{(s,a,Q) \in B} (Q - Q_{policy}(s, a))^2$$

    Replace Q_{target} parameters with Q_{policy} parameters every N_{target} steps

  end for

end for

How to use it?



New dataset
s_{static}
Initial step
$\{0\}^{P+1}$
$s_{dynamic}$
Policy network
Hyperparameter configuration
Reward
Append

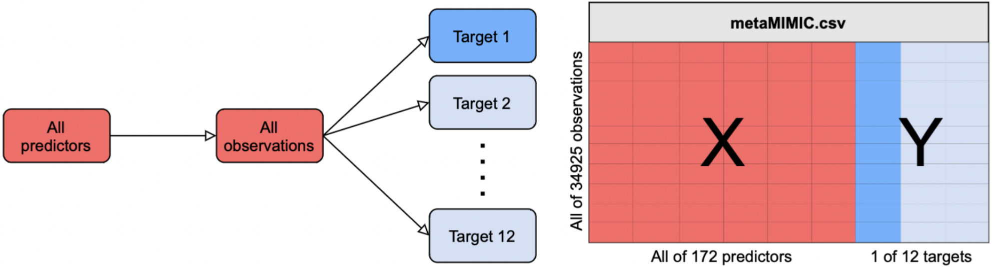
Our approach

Hyperparameter grid

Hyperparameter	Type	Lower	Upper	Distribution
n_estimators	integer	1	1000	U
learning_rate	float	0.031	1	2^U
booster	discrete	-	-	U
subsample	float	0.5	1	U
max_depth	integer	6	15	U
min_child_weight	float	1	8	2^U
colsample_bytree	float	0.2	1	U
colsample_bylevel	float	0.2	1	U

1000 independant random configurations

metaMIMIC data



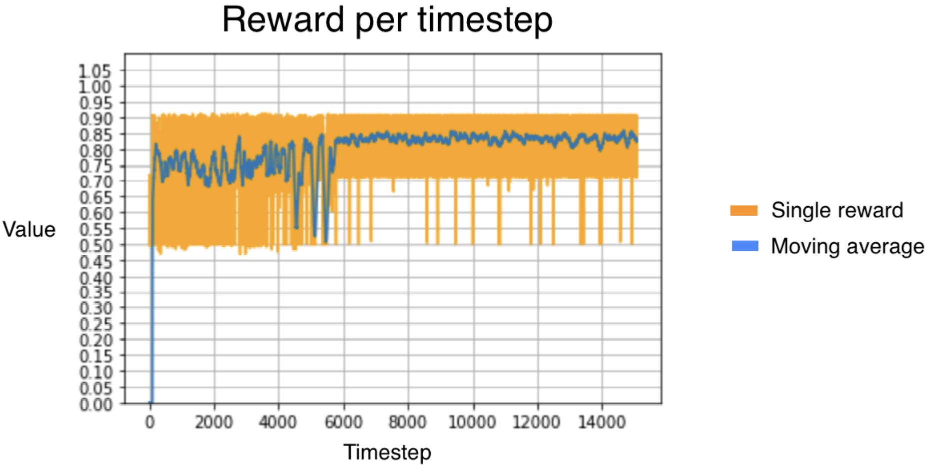
Value of ROC AUC for each hyperparameter configuration and target

Details

Reward $\rightarrow$ ROC AUC

$s_{\text{static}} \rightarrow$ Target ID

How should it look like?



Thank you for your attention