FACULTY OF ENGINEERING
AND ARCHITECTURE

Successor Features for Transfer in Reinforcement Learning

Michaela Hanajikova, Nicola Maritan, Maria Sgaravatto

GHENT
UNIVERSITY

What is *transfer* in Reinforcement Learning?

GHENT UNIVERSITY

Transfer in Reinforcement Learning

- Process of leveraging knowledge acquired in one or more tasks to improve performance on other tasks.
- In this paper, *a task is a Markov Decision Process* (MDP).
- In particular, we are interested in the transfer between tasks:
  – Having the same components of an MDP…
  – … except for the reward function.

Example

$$M_{\text{thirsty}} = \left(\mathcal{S}, \mathcal{A}, p, r_{\text{thirsty}}, \gamma\right) \qquad M_{\text{hungry}} = \left(\mathcal{S}, \mathcal{A}, p, r_{\text{hungry}}, \gamma\right)$$

Transfer

Same MDP except for the reward function.

Successor features

GHENT UNIVERSITY

Successor features

Consider an MDP

$$M = (\mathcal{S}, \mathcal{A}, p, r, \gamma).$$

We can express the one-step reward associated with transition (s, a, s') as

$$r(s, a, s') = \boldsymbol{\phi}(s, a, s')^\top \boldsymbol{w}$$

without loss of generality.

Successor features

Let $\boldsymbol{\phi}_t \triangleq \boldsymbol{\phi}(s_t, a_t, s_{t+1})$. For a given policy π on M we have:

$$Q^\pi(s,a) \triangleq \mathbb{E}^\pi[r_{t+1} + \gamma r_{t+2} + \cdots \mid s_t = s, a_t = a]$$

$$= \mathbb{E}^\pi[\boldsymbol{\phi}_{t+1}^\top \boldsymbol{w} + \gamma \boldsymbol{\phi}_{t+2}^\top \boldsymbol{w} + \cdots \mid s_t = s, a_t = a]$$

$$= \mathbb{E}^\pi\left[\sum_{i=t}^{\infty} \gamma^{i-t} \boldsymbol{\phi}_{i+1} \mid s_t = s, a_t = a\right]^\top \boldsymbol{w}$$

$$= \boldsymbol{\psi}^\pi(s,a)^\top \boldsymbol{w}.$$

GHENT UNIVERSITY

Successor features

Consider the decomposition

$$Q^{\pi}(s,a) = \boldsymbol{\psi}^{\pi}(s,a)^{\top}\boldsymbol{w}.$$

We call $\boldsymbol{\psi}^{\pi}(s,a)$ the *successor features* (SFs) of (s,a) under policy π.

GHENT UNIVERSITY

Successor features

Consider the decomposition

$$Q^{\pi}(s,a) = \boldsymbol{\psi}^{\pi}(s,a)^{\top}\boldsymbol{w}.$$

Main idea is having:

- $\boldsymbol{\psi}^{\pi}$ to summarize M's dynamics induced by π on the environment.
- $\boldsymbol{w}$ to capture M's rewards.

Transfer via SFs

Family of MDPs

Suppose $\boldsymbol{\phi}$ is fixed. We define

$$\mathcal{M}^{\phi}(\mathcal{S}, \mathcal{A}, p, \gamma) \triangleq$$

$$\triangleq \{M(\mathcal{S}, \mathcal{A}, p, r, \gamma) \mid r(s, a, s') = \boldsymbol{\phi}(s, a, s')^{\top}\boldsymbol{w}\}.$$

The task $M_i \in \mathcal{M}^{\phi}$ is entirely defined by the corresponding $\boldsymbol{w}_i$.

Generalized Policy Improvement with SFs

Let:

- $\pi_i^\star$ be the optimal policy for M_i and $Q_i^{\pi_i^\star}$ its action-value function.
- $Q_j^{\pi_i^\star}$ the action-value function of $\pi_i^\star$ when executed in $M_j \in \mathcal{M}^\phi$.

GHENT UNIVERSITY

Generalized Policy Improvement with SFs

Suppose you know

$$\left\{Q_i^{\pi_j^\star}\right\}_{j=1,\ldots,n}$$

for $M_1, \ldots, M_n \in \mathcal{M}^\phi$ and $M_i \in \mathcal{M}^\phi$.

Greedly choose

$$\pi(s) \in \operatorname{argmax}_a \max_j Q_i^{\pi_j^\star}(s, a).$$

GHENT UNIVERSITY

Generalized Policy Improvement with SFs

$$\pi(s) \in \operatorname{argmax}_a \max_j Q_i^{\pi_j^\star}(s,a)$$

Then we are guaranteed that

$$Q_i^{\pi_i^\star}(s,a) - Q_i^{\pi}(s,a) \leq \frac{2}{1-\gamma} \boldsymbol{\phi}_{\max} \min_j \|\boldsymbol{w}_i - \boldsymbol{w}_j\|$$

where $\boldsymbol{\phi}_{\max} = \max_{s,a} \|\boldsymbol{\phi}(s,a)\|$.

GHENT UNIVERSITY

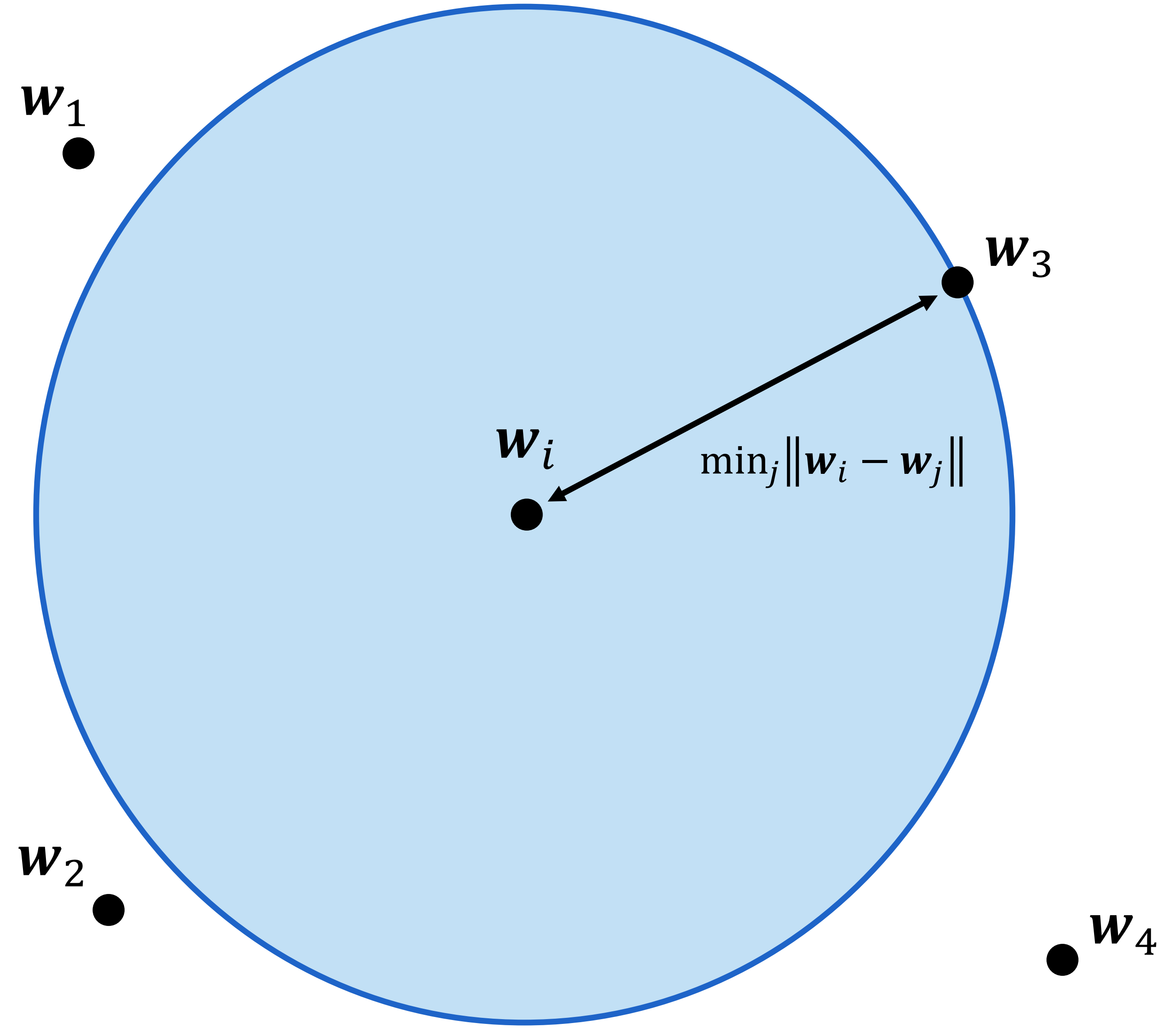
What does this mean?

$$Q_i^{\pi_i^\star}(s,a) - Q_i^{\pi}(s,a) \leq \frac{2}{1-\gamma}\, \boldsymbol{\phi}_{\max} \min_j \lVert \boldsymbol{w}_i - \boldsymbol{w}_j \rVert$$

In this example:

$j \in \{1,2,3,4\}$

$\boldsymbol{w}_i, \boldsymbol{w}_j \in \mathbb{R}^2$



π performance on task M_i will **not** be close to $\pi_i^\star$

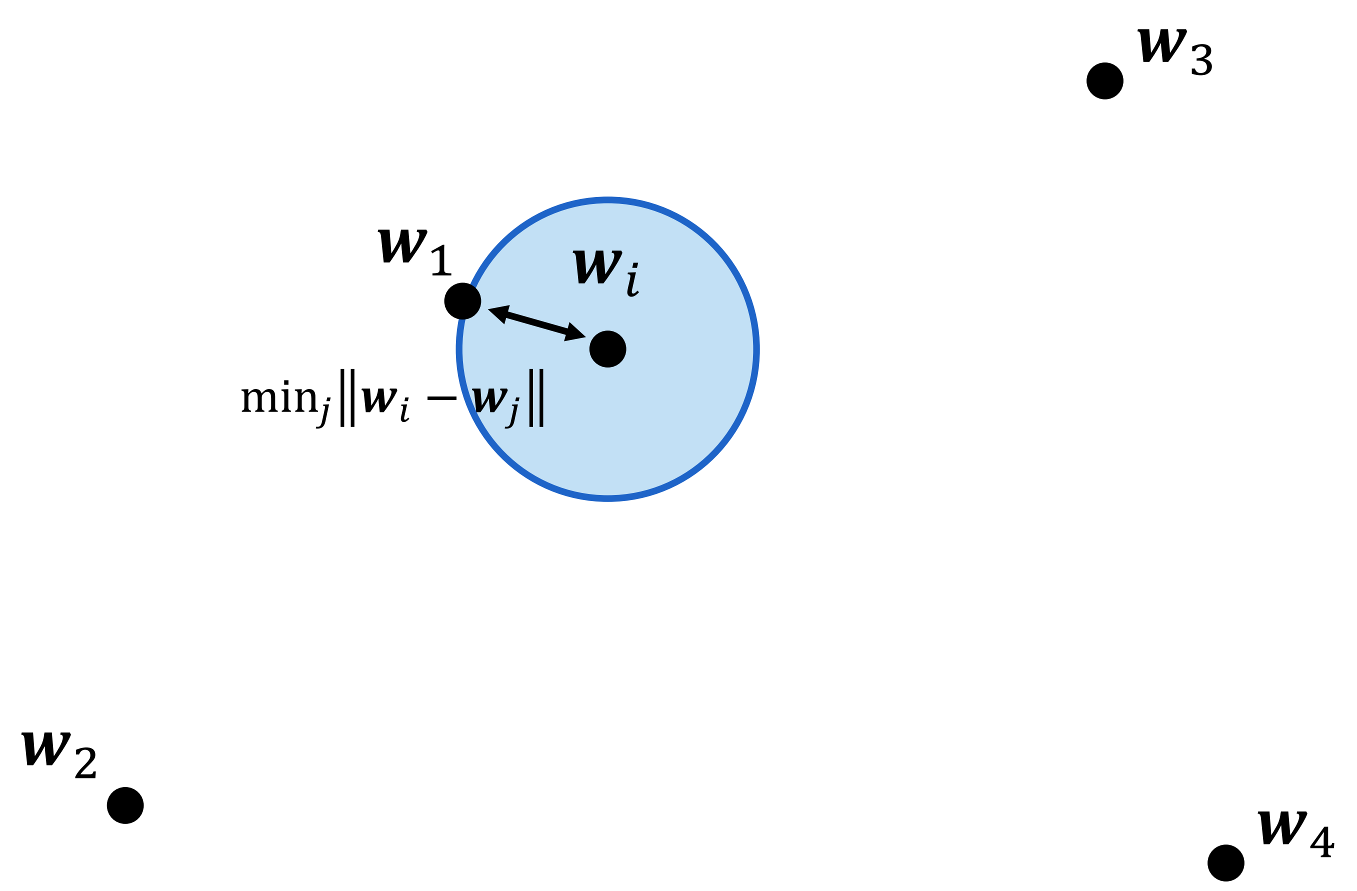
What does this mean?

$$Q_i^{\pi_i^\star}(s,a) - Q_i^{\pi}(s,a) \leq \frac{2}{1-\gamma} \boldsymbol{\phi}_{\max} \min_j \left\| \boldsymbol{w}_i - \boldsymbol{w}_j \right\|$$

In this example:

$j \in \{1,2,3,4\}$

$\boldsymbol{w}_i, \boldsymbol{w}_j \in \mathbb{R}^2$



π performance on task M_i will be close to $\pi_i^\star$

Transfer via SFs – In practice

GHENT UNIVERSITY

The two versions of SFQL

1. **SFQL-$\boldsymbol{\phi}$**
   - assumption:
     - $\boldsymbol{\phi}$ is known

2. **SFQL-$\boldsymbol{h}$**
   - assumption:
     - $\boldsymbol{\phi}$ is *not* known
     - $\boldsymbol{\phi} \in \mathbb{R}^h$

GHENT UNIVERSITY

SFQL-ϕ

Given initial tasks $M_1, \ldots, M_n \in \mathcal{M}^{\boldsymbol{\phi}}$, learn and store

$$\left\{\boldsymbol{\psi}^{\pi_j^\star}, \boldsymbol{w}_j\right\}_{j=1,\ldots,n}$$

Given a new task $M_i \in \mathcal{M}^{\boldsymbol{\phi}}$:

1. Learn $\boldsymbol{w}_i$.
2. Compute π using the GPI with SFs Theorem.
   – This becomes *trivial* using SFs

$$Q_i^{\pi_j^\star}(s,a) = \boldsymbol{\psi}^{\pi_j^\star}(s,a)^\top \boldsymbol{w}_i$$

GHENT UNIVERSITY

SFQL-ϕ

Given initial tasks $M_1, \ldots, M_n \in \mathcal{M}^{\phi}$, learn and store

$$\left\{\boldsymbol{\psi}^{\pi_j^\star}, \boldsymbol{w}_j\right\}_{j=1,\ldots,n}$$

Given a new task $M_i \in \mathcal{M}^{\phi}$:

1. Learn $\boldsymbol{w}_i$.
2. Compute π using the GPI with SFs Theorem.
3. Learn $\boldsymbol{\psi}^{\pi_i^\star}$ starting from π.
4. Store $\left\{\boldsymbol{\psi}^{\pi_i^\star}, \boldsymbol{w}_i\right\}$.

GHENT UNIVERSITY

SFQL-ϕ – Learning $\boldsymbol{\psi}^{\pi}$

It is easy to see that

$$\boldsymbol{\psi}^{\pi}(s,a) = \boldsymbol{\phi}_{t+1} + \gamma \mathbb{E}^{\pi}\left[\boldsymbol{\psi}^{\pi}\left(s_{t+1}, \pi(s_{t+1})\right) \mid s_t = s, a_t = a\right]$$

- SFs satisfy a Bellman equation.
- Any RL method can be used.
- The paper uses Q-Learning (QL), hence the name SFQL.

GHENT UNIVERSITY

SFQL-ϕ – Learning $\boldsymbol{w}$

Since we assume $\boldsymbol{\phi}$ is known and

$$r(s, a, s') = \boldsymbol{\phi}(s, a, s')^\top \boldsymbol{w}$$

This is an easy supervised learning problem.

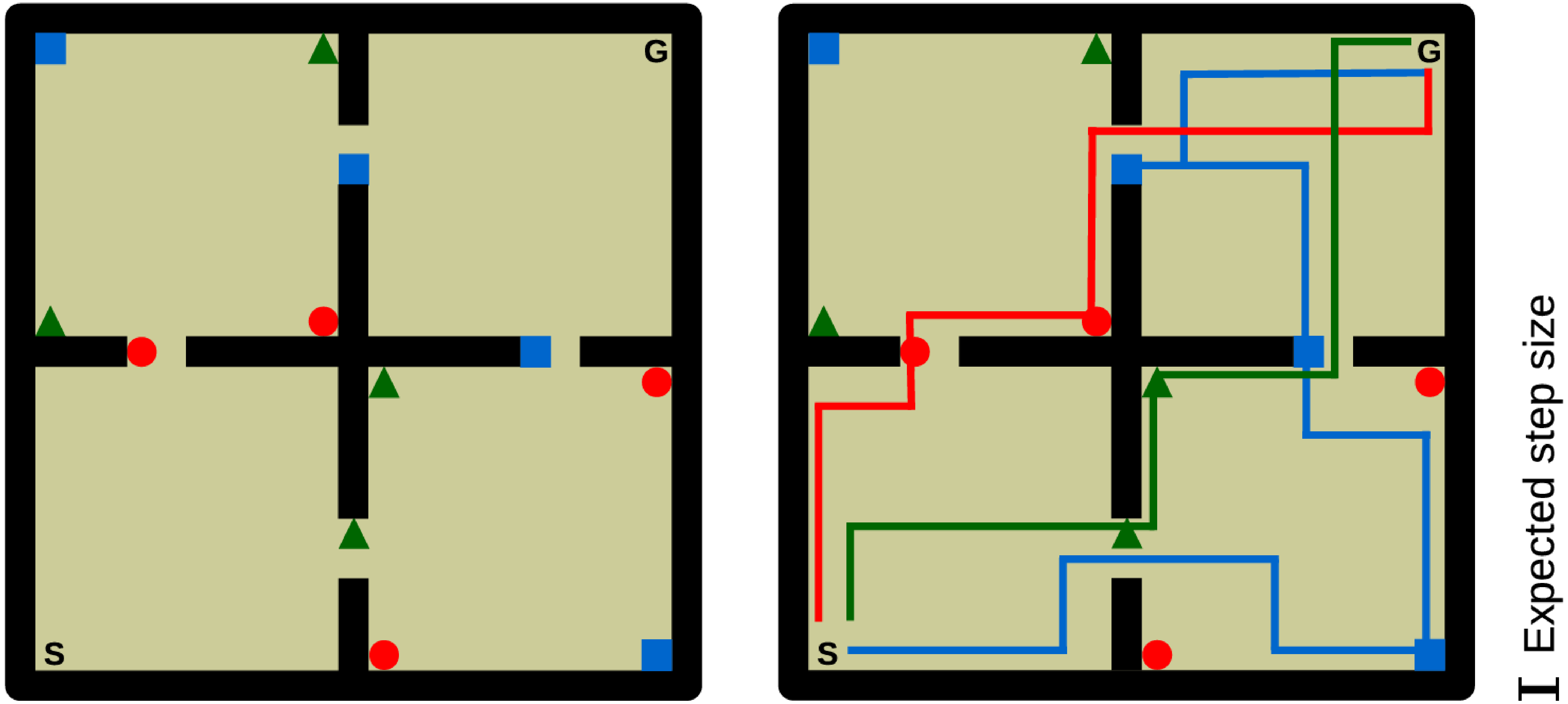
SFQL-h

Then by using the samples collected by QL in the first 20 tasks, we simultaneously learn $\boldsymbol{w}_1, \dots, \boldsymbol{w}_{20}$ and $\boldsymbol{\phi}$.

Experiment

Environment layout

Layout and optimal trajectories associated with specific tasks (i.e. colors).



Results

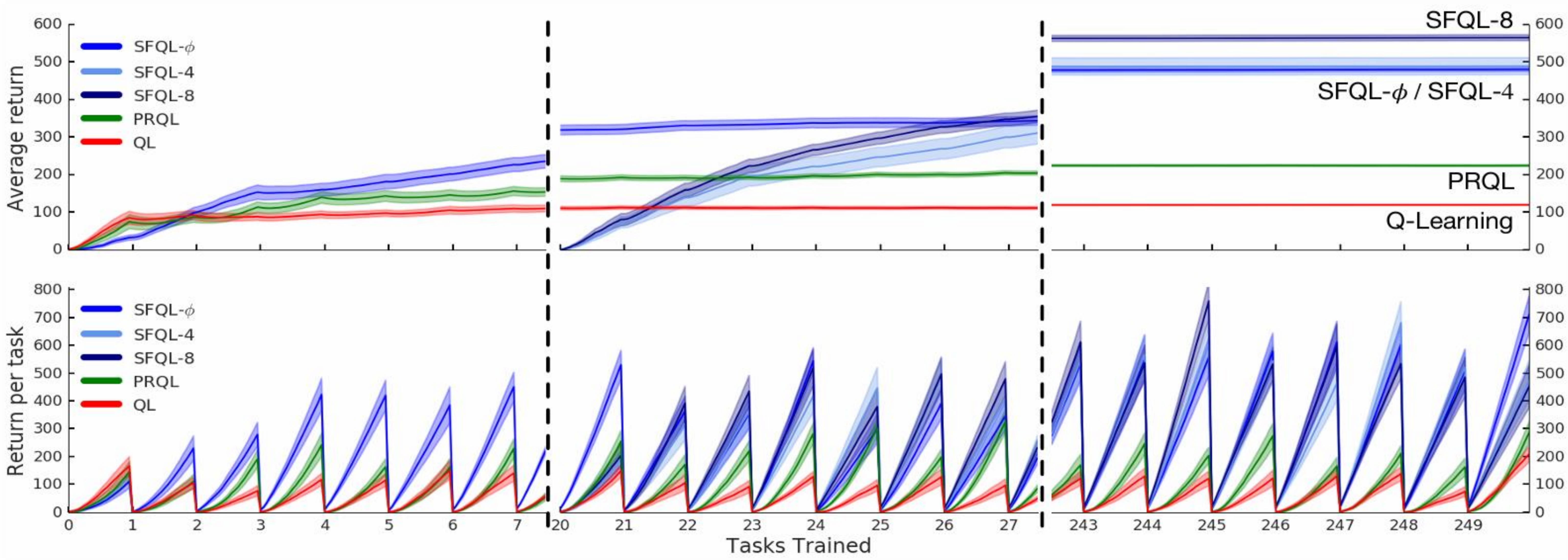


Figure 2: Average and cumulative return per task in the four-room domain. SFQL-h receives no reward during the first 20 tasks while learning $\tilde{\phi}$. Error-bands show one standard error over 30 runs.

GHENT UNIVERSITY

Results

- All versions of SFQL significantly outperform the other two methods.
- SFQL-h seems to achieve good overall performance faster than SFQL-ϕ.
  - One possible explanation could be that the learnt $\boldsymbol{\phi}$ may be less sparse than the real one, thus facilitating learning.

Thank you for your attention

GHENT UNIVERSITY