Importance Sampling based transfer in RL

POLITECNICO DI MILANO

Andrea Sessa

Mat. 850082

Supervisor: Prof. *Marcello Restelli*

Cosupervisor: Dott. *Matteo Pirotta*

Introduction

Motivation: solving a task in Reinforcement Learning, in general, is not easy: **a lot** of experience (interactions with the environment) is needed. In many real-world situation (e.g. robotics) this can be a problem.

Many times previous experience in tasks similar to the target one is available.

Goal: reuse the past experience to speed-up the learning performance in the target task avoiding at the same time the **negative transfer** problem.

Reinforcement Learning (RL)

A task in the context of RL is formalized as a **Markov Decision Process** (MDP).
A MDP is defined a tuple, $< \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma >$, where:

- $\mathcal{S}$ is the state space.
- $\mathcal{A}$ is the action space.
- $\mathcal{P} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathcal{S})$ is **transition** function.
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Pi(\mathbb{R})$ is the **reward** function.
- $\gamma \in [0, 1]$ is the discount factor for the MDP.

The goal is learn an optimal (greedy) *policy*:

$$\pi^*(s) = arg \max_{a \in \mathcal{A}} Q^*(s, a)$$

where $Q^*(s, a)$ (optimal Q-function) accounts for the discounted expected reward.

Batch Reinforcement Learning

In the rest of this presentation we will assume the context of **Batch RL**:

An **experience sample** is defined as a tuple (s, a, s', r).

Batch Reinforcement Learning

In the rest of this presentation we will assume the context of **Batch RL**:

An **experience sample** is defined as a tuple (s, a, s', r).

Then the learning procedure is divided into two distinct phases:

- **Sampling Phase**: Samples are collected from the MDP and stored.
- **Learning Phase**: The samples collected in the previous phase are used to learn a policy π over the MDP.

The separation between the two phases gives us some advantages in the transfer procedure.

Transfer Learning

We denote using T the target task.
We denote by $\{S_i\}_{i=1}^{N_s}$ the set of source tasks.

We focus on the transfer of experience **samples** from S_i to T.

Transfer Learning

We denote using T the target task.
We denote by $\{S_i\}_{i=1}^{N_s}$ the set of source tasks.

We focus on the transfer of experience **samples** from S_i to T.
The transfer is achieved by associating a weight $w \geq 0$ to each sample:

- $w \approx 0$ indicates a sample that should **not** be transferred to T.
- $w > 0$ indicates a sample that should be transferred to T.

Every sample in T has $w = 1$.

Importance Sampling

Weights are calculated using the idea of **Importance Sampling**:

For each sample (s, a, s', r) we consider two weights w_r and w_s associated to the reward and transition described by the sample.

Importance Sampling

Weights are calculated using the idea of **Importance Sampling**:

For each sample (s, a, s', r) we consider two weights w_r and w_s associated to the reward and transition described by the sample.

Using the theory of Importance Sampling the definition of the weights is:

$$w_s = \frac{\mathcal{P}_T(s'|s,a)}{\mathcal{P}_S(s'|s,a)} \quad w_r = \frac{\mathcal{R}_T(r|s,a)}{\mathcal{R}_S(r|s,a)}$$

And then taking $w = w_r w_s$.

The estimations is **unbiased** but the variance could be very high (even infinite in some situation).

Estimating the weights - 1

In practice the model of rewards and transition are not known.

We use a pair of **Gaussian Process regressor** (Gaussian assumption) for each task (target included).
Under these conditions the weight are distributed according to an unknown distribution.

Estimating the weights - 1

In practice the model of rewards and transition are not known.

We use a pair of **Gaussian Process regressor** (Gaussian assumption) for each task (target included).
Under these conditions the weight are distributed according to an unknown distribution.

We are able to prove the following result:

$$\mathbb{E}[\tilde{w}_x(\tilde{\mu}_T, \tilde{\mu}_S)] = \begin{cases} \frac{\sigma^2}{\sigma^2 - \sigma^2_{GP,S}} \frac{\mathcal{N}(x; \mu_{GP,T}, \sigma^2 + \sigma^2_{GP,T})}{\mathcal{N}(x; \mu_{GP,T}, \sigma^2 - \sigma^2_{GP,S})} & \sigma^2_{GP,S} < \sigma^2 \\ \infty \quad Otherwise \end{cases}$$

where x can be either r or s'.

Estimating the weights - 2

The previous equation could be used but may lead to very high weights when σ^2 approaches $\sigma^2_{GP,S}$ and the algorithm may need to discard the sample when $\sigma^2 > \sigma^2_{GP,S}$.

Estimating the weights - 2

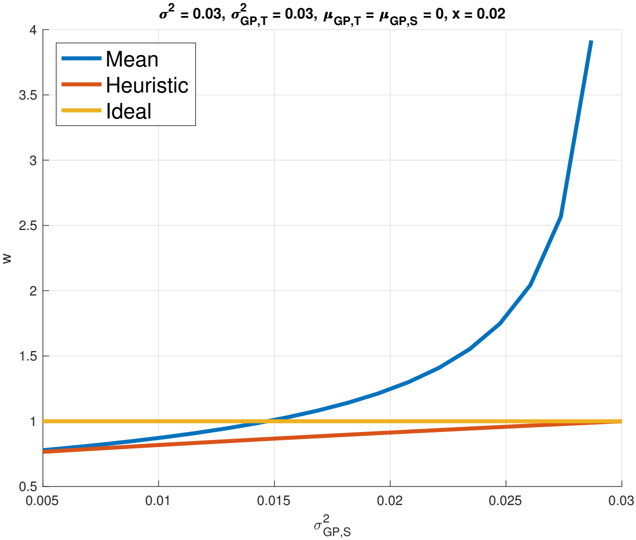
The previous equation could be used but may lead to very high weights when σ^2 approaches $\sigma^2_{GP,S}$ and the algorithm may need to discard the sample when $\sigma^2 > \sigma^2_{GP,S}$.

A possible **heuristics** may be proposed as:

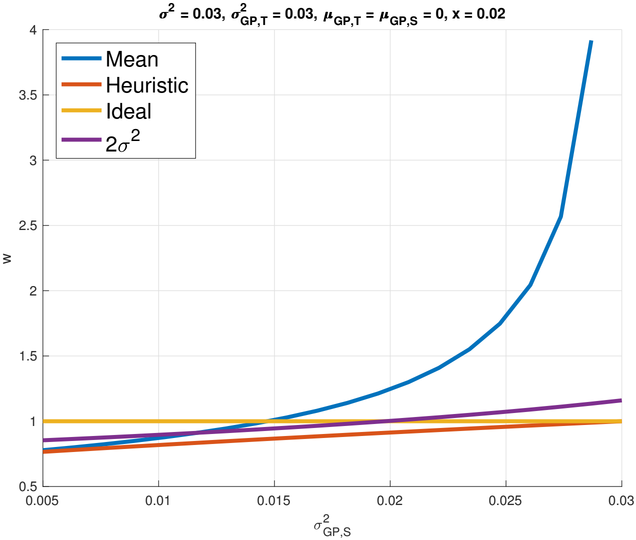
$$\tilde{w}(x) = \frac{\mathcal{N}(x; \mu_{GP,T}, \sigma^2 + \sigma^2_{GP,T})}{\mathcal{N}(x; \mu_{GP,S}, \sigma^2 + \sigma^2_{GP,S})}$$

which still converges to the ideal weights when the GP are perfectly accurate.

Comparing different estimations



Comparing different estimations



Using the weights - (W)FQI

We apply our weight estimation procedure to a well known Batch RL algorithm: **Fitted Q-Iteration (FQI)**

Using the weights - (W)FQI

We apply our weight estimation procedure to a well known Batch RL algorithm: **Fitted Q-Iteration (FQI)**
The main idea of FQI is to use the samples collected in conjunction with a **regression** algorithm to obtain at each iteration an increasingly better estimation of the Q-function.

$$\hat{Q}^{k+1} = arg \min_{f \in \mathcal{F}} \frac{1}{N_t} \sum_{i=1}^{N_t} ||f(s_i, a_i) - \mathcal{T}\hat{Q}^k||^2$$

Using the weights - (W)FQI

We apply our weight estimation procedure to a well known Batch RL algorithm: **Fitted Q-Iteration (FQI)**
The main idea of FQI is to use the samples collected in conjunction with a **regression** algorithm to obtain at each iteration an increasingly better estimation of the Q-function.

$$\hat{Q}^{k+1} = arg \min_{f \in \mathcal{F}} \frac{1}{N_t} \sum_{i=1}^{N_t} ||f(s_i, a_i) - \mathcal{T}\hat{Q}^k||^2$$

Using a **weighted** regression algorithm permits to exploit the capability of FQI adding the possibility to transfer samples across multiple tasks.

Using the weights - (W)FQI

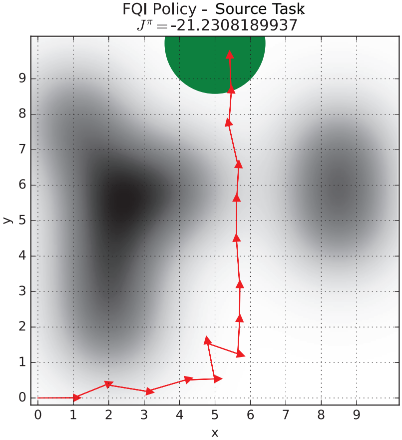
We apply our weight estimation procedure to a well known Batch RL algorithm: **Fitted Q-Iteration (FQI)**
The main idea of FQI is to use the samples collected in conjunction with a **regression** algorithm to obtain at each iteration an increasingly better estimation of the Q-function.

$$\hat{Q}^{k+1} = arg\min_{f \in \mathcal{F}} \frac{1}{N_t} \sum_{i=1}^{N_t} ||f(s_i, a_i) - \mathcal{T}\hat{Q}^k||^2$$

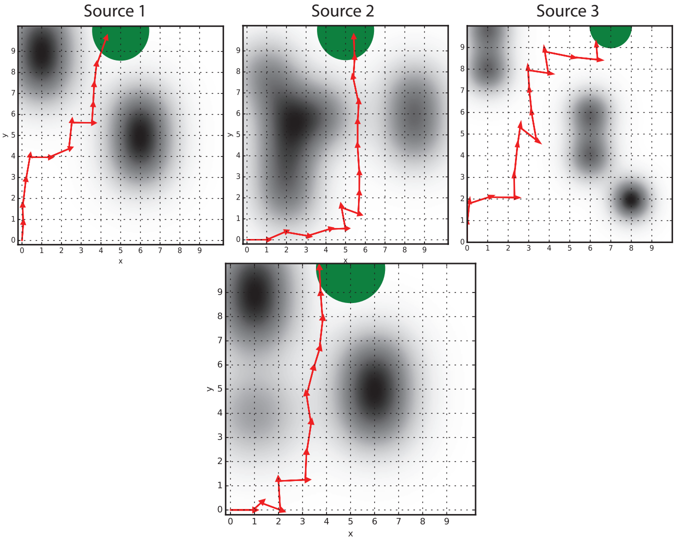
Using a **weighted** regression algorithm permits to exploit the capability of FQI adding the possibility to transfer samples across multiple tasks.
Given a sample (s, a, s', r):

$$\hat{Q}^{k+1} = arg\min_{f \in \mathcal{F}} \frac{1}{N_t + N_s} \sum_{i=1}^{N_t + N_s} w_i ||f(s_i, a_i) - \mathcal{T}\hat{Q}^k||^2$$

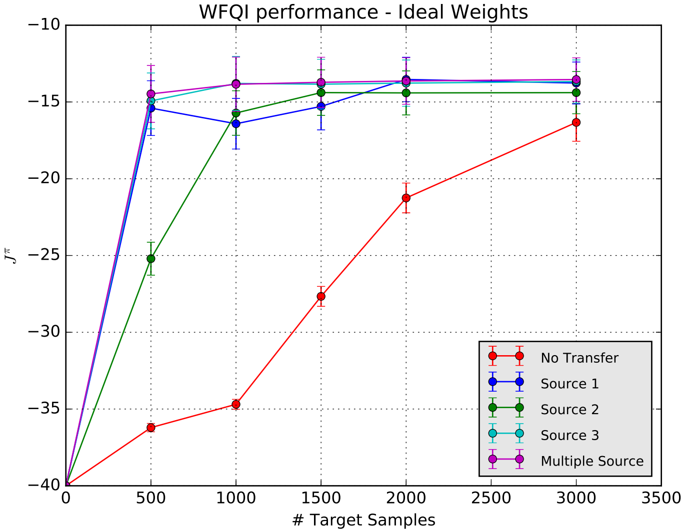
Experiments - Puddle World



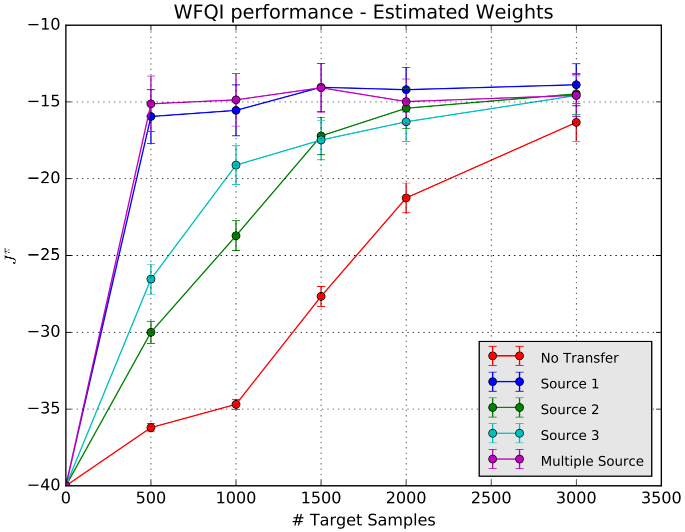
- **Continuous** state space
- **Discrete** action space
- Gaussian reward/transition model


Source 1
Source 2
Source 3
x
y
0
1
2
3
4
5
6
7
8
9

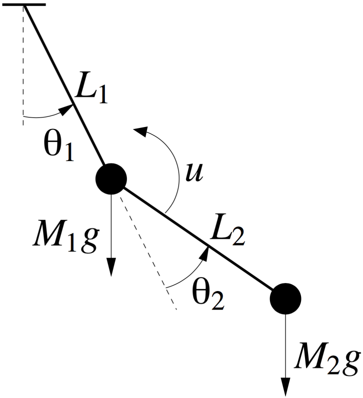
Experiments - Puddle World - Ideal Weights



Experiments - Puddle World - Estimated Weights

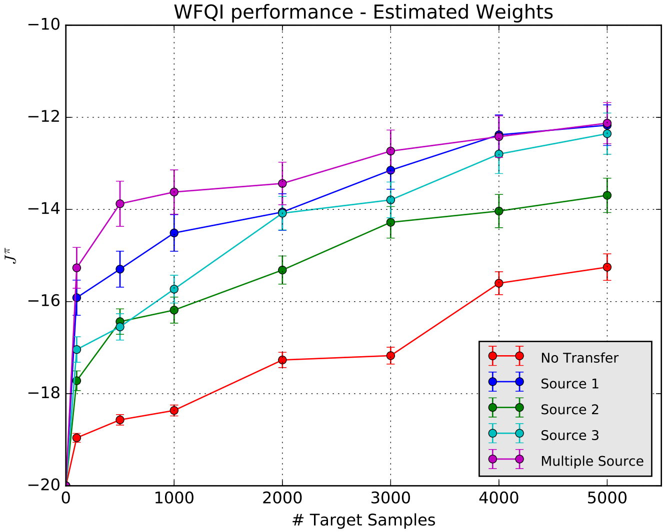


Experiments - Acrobot



- **Continuous** state space
- **Discrete** action space
- **Non**-gaussian reward/transition model

Experiments - Acrobot - Estimated Weights



Conclusions - Future Developments

We have developed a transfer learning approach with a strong **theoretical** background (not shown).

Empirical result over different environments have shown the effectiveness of such approach.

Possible future developments:

- More effective weights selection procedure
- Application to more challenging environments

Additional - The Algorithm

Additional - The Algorithm

Algorithm 2 Weighted Fitted Q-Iteration algorithm

procedure WFQI($\mathcal{D} = (s, a, s', r, w_s, w_r)_{k=1}^{N}, myWeightedRegressionAlg$)
    $k \leftarrow 0$
    $\hat{Q}^0(s, a) \leftarrow 0\ \forall (s, a) \in S \times A$
    $D' \leftarrow (s_k, a_k, r_k)_{k=1}^{N}$
    $\hat{Q}^1 \leftarrow myWeightedRegressionAlg(\mathcal{D}', w_r)$
    $\mathcal{D}' = \emptyset$
    while $checkStoppingCriteria()$ **do**
        $k \leftarrow k + 1$
        for $l = 1$ to N **do**
            $i_l = (s_l, a_l)$
            $o_l = Q^1 + \gamma \max_{a' \in A} \hat{Q}_{k-1}(s', a')$
            $\mathcal{D}'$**+=** (i_l, o_l)
        $\hat{Q}^k \leftarrow myWeightedRegressionAlg(\mathcal{D}', w_s)$