Policy Transfer using Value Function as Prior Information

Samy Aittahar[1], Aivar Sootla[1], and Damien Ernst[1]

Department of Electrical Engineering and Computer Science,
University of Liège, Belgium
{saittahar, asootla, dernst}@ulg.ac.be

Abstract. This work proposes an approach based on reward shaping techniques in a reinforcement learning setting to approximate the optimal *decision-making process* (also called the optimal *policy*) in a desired task with a limited amount of data. We extract prior information from an existing family of policies have been used as a heuristic to help the construction of the new one under this challenging condition. We use this approach to study the relationship between the similarity of two tasks and the minimal amount of data needed to compute a near-optimal policy for the second one using the prior information of the existing policy. Preliminary results show that for the least similar existing task considered compared to the desired one, only 10% of the dataset was needed to compute the corresponding near-optimal policy.

Keywords: Reinforcement learning · Transfer learning · Reward shaping · Biological systems

1 Introduction

A decision-making process can be defined as a procedure which chooses from a set of alternatives. Optimization problems related to the decision-making processes are very interesting since they lead to various concrete applications (e.g., autonomous car driving, spam classification). But often, designing analytically such an optimal decision making process can be difficult when very complex tasks are considered. A popular way to overcome this difficulty is to use existing observations to approximate an optimal decision-making process. In others words, a decision-making agent *learns* to perform these tasks using the information in the observations while trying to optimize a particular family of criteria. We call the function mapping observations to decisions *a policy*. Machine learning techniques are particularly efficient to design automatically these policies, given a set of observations [13].

However, data acquisition can also be very difficult and/or expensive. This is particularly true for biological systems, where data collection is time consuming, requires laboratory experiments using expensive equipment and materials. To overcome this issue, one possibility is to use a family of policies that have already been computed for similar tasks. Then these policies can be used to build a

new one for the desired task. In this work, we propose an approach to use an existing family of policy to build a new one with limited set of data coming from the desired task. We use a batch-mode reinforcement learning framework, in particular we employ the *Fitted-Q-Iteration* (FQI) algorithm [2]. Then we encode prior information coming from such a family of policy using the so-called *potential functions*. This technique leads to a modification of the FQI algorithm and is known as *reward shaping* [7]. This approach allows us to study the relationship between the similarity of two tasks and the minimal amount of data needed to build the second policy using the prior information of the policy built for the first task.

The case study that we propose to investigate is a biological system called *genetic toggle switch system* [5]. The goal of the task that we set up in this system is to regulate the concentration of two proteins in a cell. This example comes from a real life application that has been studied in the past [12]. A model is known, however the exact values of parameters are hard to identify due to variability in the cell phenotype. The difficulty of the problem is amplified due to small amount of available data.

This paper is organized as follows. Section 3 describes the reinforcement learning framework. Section 4 proposes an adaptation of the *Fitted-Q-Iteration* algorithm for reward shaping. Section 5 defines our case study discussed above. Section 6 shows the result of the simulation based on the whole framework. Section 7 concludes this paper.

2 Related Work

Transfer between decision-making processes using decision-tree based boosting techniques have been discussed in a supervised learning setting for classification problems [3] [14]. Mapping between heterogeneous tasks have been adressed with Restricted Boltzmann Machine (RBM) [1], a particular class of neural networks [10].

3 Reinforcement Learning Setting

Let us consider discrete-time systems, which is a specific case of a *Markov Decision Process* [8] in the following form:

$$\begin{aligned} s_{t+1} &= f(s_t, u_t) \ , \\ c_t &= c(s_t, u_t, f(s_t, u_t)) \ , \end{aligned} \tag{1}$$

where $s_t \in S$ is the state of the system at time t, $u_t \in U$ is an action applied to the system at state s_t and $c_t \in C$ is the cost of the applied action u_t in the system state s_t.

We aim to find a sequence of actions $u_0 \cdots u_\infty$ in order to perform a task while minimizing the sum of all the cost from $c_0 \cdots c_\infty$. More specifically, we aim to compute a state-action mapping able to perform the task while minimizing

this sum of cost. To do so, we use the following infinite-dimensional optimization problem:

$$V(s_t) = \min_{\mu^*(\cdot)} \sum_{i=t}^{+\infty} \gamma^{i-t} c(s_i, \mu^*(s_i), f(s_i, \mu^*(s_i))) \ , \tag{2}$$

where $V(s_t)$ is called the *value function*, γ is the discount factor decreasing over time, $\mu^*(\cdot) : S \to U$ is the optimal state-action mapping (also called an *optimal policy*).

The solution of this optimization problem satisfies the Bellman equation [13]:

$$V(s) = \min_{u \in U} \left(c(s, u, f(s, u)) + \gamma V(f(s, u)) \right) . \tag{3}$$

In practice, it is more convenient to consider a mapping between a state, an action and an estimate of the long-term cost. *Q-functions* express such a mapping:

$$Q(s, u) = c(s, u, f(s, u)) + \gamma V(f(s, u)) \ . \tag{4}$$

The main approach used to compute this function is to use the following recursive procedure:

$$Q_k(s, u) = c(s, u, f(s, u)) + \gamma \min_{v \in U} Q_{k-1}(f(s, v), v), \forall k > 0 \ , \tag{5}$$

where Q_k approximates a long-term discounted cost at iteration k and $Q_0(s_t, u_t) = c(s_t, u_t)$. Intuitively, this is often an estimation of the sum of discounted cost in a time horizon of k. When the desired (i.e. *near-optimal*) Q-function Q^* is computed, the value function is given by the following equation:

$$V(s) = \min_{v \in U} Q^*(s, v) \ . \tag{6}$$

Even with this approach, it is often difficult to compute the value function analytically, because (i) the state-transition function f may not be known and (ii) even if f is known, it is often intractable to find directly the *value function*.

A popular way to approximate the value function is to observe how the system reacts given a set of actions. It means also that we can collect samples in terms of *one-step transitions*, denoted by the quadruplet $\{s_t, u_t, c_t, s_{t+1}\}$. Therefore, we assume that we have a dataset of one-step transitions, the main input to approximate the optimal *Q-function* and therefore the *value function*.

This setting is known as *Batch-mode reinforcement learning* as discussed in [4]. Many algorithms can be used to address this optimization problem (e.g., SARSA [11]). We choose the *Fitted-Q-Iteration* algorithm as it is directly designed for this reinforcement learning setting and for its convergence speed properties, discussed in [2].

Algorithm 1 Reward-shaping based FQI

Inputs: Set of quadruplets $\mathcal{F} = \{s_t, u_t, c_t, s_{t+1}\}$, stopping criterion, potential function $\Phi(\cdot)$
Outputs: Policy $\hat{\mu}^*(s)$
Set $k = 0$
Set $\hat{Q}_0(s_t, u_t) = c_t + \gamma\Phi(s_{t+1}) - \Phi(s_t), \forall\{s_t, u_t, c_t, s_{t+1}\} \in \mathcal{F}$.
repeat
    Set $k = k + 1$
    Compute, for all $\{s_t, u_t, s_{t+1}, c_t\} \in \mathcal{F}$:

$$\hat{Q}_k(s_t, u_t) = c_t + \gamma\Phi(s_{t+1}) - \Phi(s_t) + \gamma \min_{u \in U} \hat{Q}_{k-1}(s_{t+1}, u)$$

    Approximate directly the $\hat{Q}_k(s_t, u_t)$ function using a regression algorithm.
until the stopping criterion is met
Returns $\hat{\mu}^*(s) = \arg\min_{u \in U} \hat{Q}_k(s, u)$

4 Reward Shaping Based Fitted-Q-Iteration

Let us consider the following modification of the *Q-function* defined in Equation (5):

$$Q_k(s, u) = c(s, u, f(s, u)) + \gamma\Phi(f(s, u)) - \Phi(s) + \gamma \min_{v \in U} Q_{k-1}(f(s, v), v) \ \forall k > 0 \ , \tag{7}$$

where $\Phi(s_t)$ is called a potential function. Reward shaping properties have essentially been discussed in [9]. The authors have essentially stated that (i) initializing the *Q-function* with either the instantaneous cost or the potential function leads to the same *near-optimal* policy and (ii) the potential function can be used as an heuristic to speed up the learning process, as empirically shown in [9].

For example, let us consider a sparse cost function, for which the cost is strictly negative only for a restricted set. Without any prior information and starting in a state outside this set, the estimated long-term cost of each available action for this state does not provide enough information to be able to choose it to reach such a set of states. Designing a proper potential function with prior information about the system can provide a more informative long-term cost and therefore helps to pick the desired action. Algorithm 1 describes a reward shaping version of the FQI algorithm.

5 Regulation of a Toggle Switch System

First, we briefly describe the benchmark problem of regulation of a genetic toggle switch. We consider two genes ($lacI$ and $tetR$) with their respective proteins

concentration, referred respectively as gene 1 and gene 2. We consider also a binary action space that represents the light pulse activating a photo-sensitive promoter controlling the expression of the *lacI* gene. When the photo-sensitive promoter is activated, its concentration is increased instantaneously by a small amount. This section also describes the considered configuration for the learning framework.

5.1 Dynamics

The state-transition dynamics describing the two proteins concentration are defined by the following equations:

$$\begin{aligned} \dot{s}^1 &= \beta_1 + \frac{c_1}{1 + (s^2/r_1)^{\alpha_1}} - c_2 s^1 + bu \ , \\ \dot{s}^2 &= \beta_2 + \frac{c_3}{1 + (s^1/r_2)^{\alpha_2}} - c_4 s^2 \ , \end{aligned} \tag{8}$$

where s^i is the concentration of the protein i at time t, c_1 and c_3 are the effective rate of synthesis of the proteins, α_i is the cooperativity coefficient of the protein i, c_2 and c_4 are the degradation rates of proteins, β_i models basal transcription level during the gene expression of the gene i, r_i is the gene repression rate of gene i, and b is the increase in protein concentration produced per unit of time as a result of one light pulse when activated. The sampling time used to compute the state transition is 1 seconds. The instantaneous cost in this system is defined by the following equation:

$$c(\langle s^1, s^2 \rangle, u, \langle s^1_+, s^2_+ \rangle) = -\frac{s^1_+}{n_1} + \frac{s^2_+}{n_2} + pu \ , \tag{9}$$

where p is the penalty coefficient applied to the action, $\langle s^1_+, s^2_+ \rangle = f(\langle s^1, s^2 \rangle, u)$ and n_1, n_2 are used to restrict interval of the instantanous cost in order to make the parameter tuning more convenient. Table 1 enumerates the different dynamics with their respective parameters.

Table 1. Parameters of the generation model described in Equations 8 and 9

MDP	β_1	β_2	c_1	c_2	c_3	c_4	α_1	α_2	b	r_1	r_2	n_1	n_2	p
T	0	0	7000	1	10000	1	2	2	400	10000	10000	10000	100	40
S_1	0	0	7000	1	10000	1	2	2	1000	5000	5000	10000	100	40
S_2	0	0	7000	1	10000	1	2	2	4000	100	100	10000	100	40
S_3	0	0	10000	1	7000	1	2	2	400	100	100	10000	100	40

5.2 Dataset

5000 time series of 20 steps have been generated by a random decision process (uniform distribution of the actions) during the simulation for each generation model.

5.3 Learning Configuration

Fitted-Q-Iteration 100 iterations have been performed with $\gamma = 0.75$. The approximator used is the well-known Extremely Randomized Trees algorithm [6] with a forest of 100 trees. We also take $\Phi(s)$ to be an existing value function V_S built from another task S.

5.4 Metrics

Let us consider the following score function related to a performance of a policy μ in any given one-step transition function $f(x, u)$:

$$s(\mu, f) = \sum_{t=0}^{T} \gamma^t c(s_t, \mu(s_t), f(s_t, \mu(s_t))) \ . \tag{10}$$

We also consider to compute the *regret* score as a proxy for the task similarity purpose. We defined the *regret* score as a distance between the best policy μ^* fitted for a given one-step transition function f and another policy μ' fitted for another one-step transition function by the following equation:

$$r(\mu^*, \mu', f) = s(\mu', f) - s(\mu^*, f) \ . \tag{11}$$

Let us define μ^*_K to be the policy fitted with a one-step transition database, computed with f, of size $K \in \mathbb{N}$. We say that K is the *minimal amount* of data needed to reach the performance of μ^* fitted if the following inequation is satisfied:

$$\frac{s(\mu^*, f)}{s(\mu^*_K, f)} \leqslant \epsilon \ , \tag{12}$$

with $\epsilon = 0.01$ in our settings.

6 Results and Discussion

First, we show that using directly an existing policy that has not been built for the desired does not allow a proper regulation of the proteins concentration. Figure 1 shows an example of such a statement. Figure 6 shows the performance achieved by using prior information from existing policy built generation model S_3 to build the desired policy for the generation model T, compared to classical reinforcement learning, with a limited amount of data.

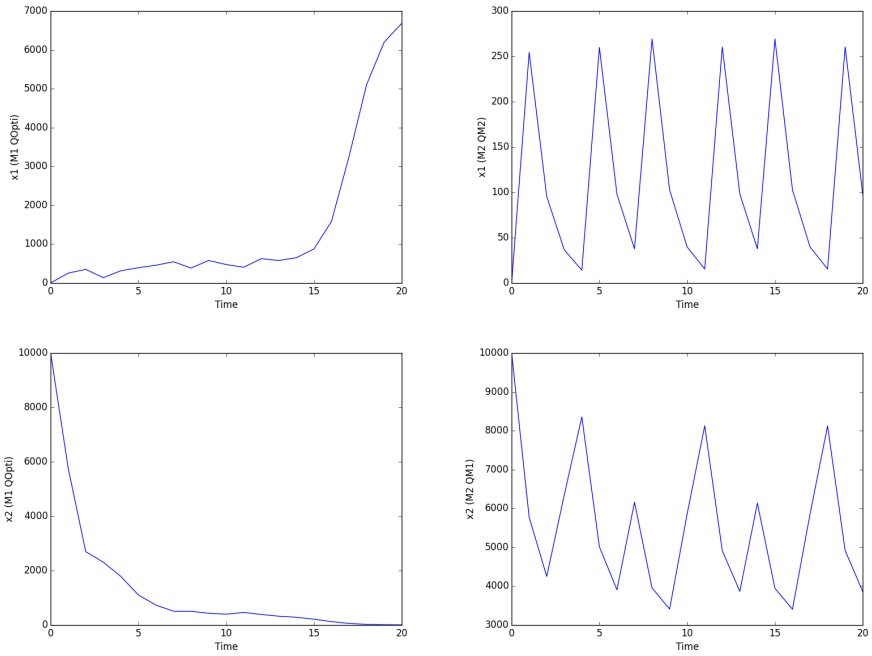

Fig. 1. Optimal policy convergence for the target model T (left) compared to the non-convergence of optimal policy fitted for model S_3 (right)

Since the dataset used to fit the existing policy contains 100000 samples, and the prior information from policy built for S_3 allows to have a dataset containing 10000 samples, it can be stated that only 10% of the dataset was needed to reach the desired performance. Table 2 shows the regret score computed with the target model and all considered source generation models.

Table 2. Regret score when using directly *Q-function* from a source model in the target one

Dist	S_1	S_2	S_3
T	517	806	1018

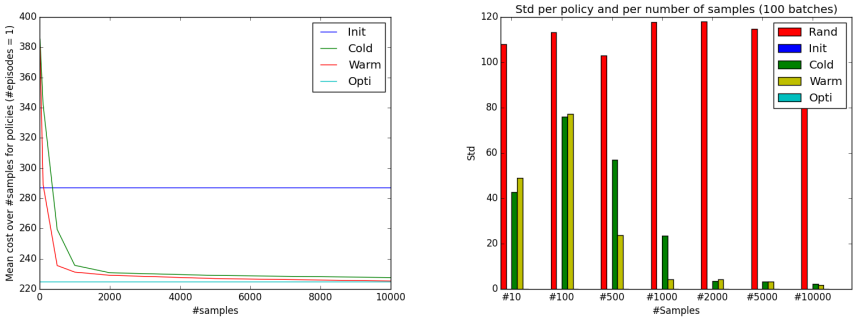

Fig. 2. Simulation results with and without reward shaping (S_3 to T, discounted cost + variance with 100 batches)

7 Conclusion

In this work, we have proposed an approach to build efficiently, with a limited amount of data, a policy for a desired task using prior information of an existing family of policies. This approach allowed us to study the relationship between the similarity of two tasks and the minimal amount of data needed to compute a near-optimal policy for the second task using prior information from the policy built for the first one. Moreover, it appeared that for the least closed task that we considered, only a database of 10000 samples (i.e. 500 time series) have been required to reach such a performance, considering that 100000 samples, (i.e., 5000 time series) are needed when no existing policy has been used.

Since we have considered a deterministic system in this paper, it would also be interesting to extend the study to a stochastic version of this system to verify the robustness of our approach in a future work.

Acknowledgments. Aivar Sootla is a Postdoctoral Fellow of the F.R.S-FNRS. The authors also thank the financial support of the Belgian Network DYSCO, an Interuniversity Attraction Poles Programme initiated by the Belgian State, Science Policy Office.

References

1. Ammar, H.B., Mocanu, D.C., Taylor, M.E., Driessens, K., Tuyls, K., Weiss, G.: Automatically mapped transfer between reinforcement learning tasks via three-way restricted boltzmann machines. In: Machine Learning and Knowledge Discovery in Databases, pp. 449–464. Springer (2013)
2. Antos, A., Szepesvári, C., Munos, R.: Fitted q-iteration in continuous action-space mdps. In: Advances in neural information processing systems. pp. 9–16 (2008)
3. Dai, W., Yang, Q., Xue, G.R., Yu, Y.: Boosting for transfer learning. In: Proceedings of the 24th international conference on Machine learning. pp. 193–200. ACM (2007)
4. Ernst, D., Geurts, P., Wehenkel, L.: Tree-based batch mode reinforcement learning. J. Mach. Learn. Res. 6, 503–556 (Dec 2005), http://dl.acm.org/citation.cfm?id=1046920.1088690
5. Gardner, T.S., Cantor, C.R., Collins, J.J.: Construction of a genetic toggle switch in escherichia coli. Nature 403(6767), 339–342 (2000)
6. Geurts, P., Ernst, D., Wehenkel, L.: Extremely randomized trees. Machine Learning 63(1), 3–42 (Apr 2006)
7. Grześ, M., Kudenko, D.: Theoretical and empirical analysis of reward shaping in reinforcement learning. In: Machine Learning and Applications, 2009. ICMLA'09. International Conference on. pp. 337–344. IEEE (2009)
8. Guo, X., Hernández-Lerma, O.: Continuous-time Markov decision processes. Springer (2009)
9. Ng, A.Y., Harada, D., Russell, S.: Policy invariance under reward transformations: Theory and application to reward shaping. In: ICML. vol. 99, pp. 278–287 (1999)
10. Riedmiller, M.: Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In: Machine Learning: ECML 2005, pp. 317–328. Springer (2005)
11. Rummery, G.A., Niranjan, M.: On-line q-learning using connectionist systems (1994)
12. Sootla, A., Oyarzún, D., Angeli, D., Stan, G.B.: Shaping pulses to control bistable systems: Analysis, computation and counterexamples. Automatica 63, 254–264 (2016)
13. Sutton, R.S., Barto, A.G.: Introduction to reinforcement learning, vol. 135. MIT Press Cambridge (1998)
14. Yao, Y., Doretto, G.: Boosting for transfer learning with multiple sources. In: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on. pp. 1855–1862. IEEE (2010)