Sparse Multi-Task Reinforcement Learning

Daniele Calandriello *
INRIA Lille - Nord Europe
Team SequeL, France

Alessandro Lazaric*
INRIA Lille - Nord Europe
Team SequeL, France

Marcello Restelli†
Politecnico di Milano
Milan, Italy

1 Introduction

In multi-task reinforcement learning (MTRL), the objective is to simultaneously learn multiple tasks and exploit their similarity to improve the performance w.r.t. single-task learning. In this paper we investigate the case when all the tasks can be accurately represented in a linear approximation space using the same small subset of the original (large) set of features. This is equivalent to assuming that the weight vectors of the task value functions are *jointly sparse*, i.e., the set of their non-zero components is small and it is shared across tasks. Building on existing results in multi-task regression, we develop two multi-task extensions of the fitted Q-iteration algorithm [4]. While the first algorithm assumes that the tasks are jointly sparse in the given representation, the second one learns a transformation of the features in the attempt of finding a more sparse representation. For both algorithms we provide numerical simulations, while an extensive theoretical analysis is reported in the long version of this paper [3].

2 Preliminaries

A Markov decision process (MDP) is a tuple $\mathcal{M} = (\mathcal{X}, \mathcal{A}, R, P, \gamma)$, where the state space $\mathcal{X}$ is a bounded closed subset of the Euclidean space, the action space $\mathcal{A}$ is finite (i.e., $|\mathcal{A}| < \infty$), $R : \mathcal{X} \times \mathcal{A} \to [0, 1]$ is the reward of a state-action pair, $P : \mathcal{X} \times \mathcal{A} \to \mathcal{P}(\mathcal{X})$ is the transition distribution over the states achieved by taking an action in a given state, and $\gamma \in (0, 1)$ is a discount factor. A deterministic policy $\pi : \mathcal{X} \to \mathcal{A}$ is a mapping from states to actions. We denote by $\mathcal{B}(\mathcal{X} \times \mathcal{A}; b)$ the set of measurable state-action functions $f : \mathcal{X} \times \mathcal{A} \to [-b; b]$ absolutely bounded by b. Solving an MDP corresponds to computing the optimal action–value function $Q^* \in \mathcal{B}(\mathcal{X} \times \mathcal{A}; Q_{\max} = 1/(1-\gamma))$, defined as the largest expected sum of discounted rewards that can be collected in the MDP and fixed point of the optimal Bellman operator $\mathcal{T} : \mathcal{B}(\mathcal{X} \times \mathcal{A}; Q_{\max}) \to \mathcal{B}(\mathcal{X} \times \mathcal{A}; Q_{\max})$ defined as $\mathcal{T}Q(x, a) = R(x, a) + \gamma \sum_y P(y|x, a) \max_{a'} Q(y, a')$. The optimal policy is obtained as the greedy policy w.r.t. the optimal value function as $\pi^*(x) = \arg\max_{a \in \mathcal{A}} Q^*(x, a)$. In this paper we study the multi-task reinforcement learning (MTRL) setting where the objective is to solve T tasks, defined as $\mathcal{M}_t = (\mathcal{X}, \mathcal{A}, P_t, R_t, \gamma_t)$ with $t \in [T] = \{1, \dots, T\}$, with the same state-action space, but different dynamics P_t and goals R_t. The objective of MTRL is to exploit possible relationships between tasks to improve the performance w.r.t. running T independent instances of single-task learning. In particular, we choose linear fitted Q-iteration as the single-task baseline and we propose multi-task extensions tailored to exploit the sparsity in the structure of the tasks.

3 Multi-task Fitted Q-iteration

Whenever $\mathcal{X}$ and $\mathcal{A}$ are large or continuous, we need to resort to approximation schemes to learn a near-optimal policy. One of the most popular approximation methods is the fitted-Q iteration (FQI) algorithm [4], which extends value iteration to approximate action-value functions. While exact value iteration proceeds by iterative applications of the Bellman operator (i.e., $Q^k = \mathcal{T}Q^{k-1}$), in FQI, each iteration approximates $\mathcal{T}Q^{k-1}$ by solving a regression problem. Among possible instances, here we focus on a specific implementation of FQI in the fixed design setting with linear approximation and we assume access to a generative model of the MDP. Since the action space $\mathcal{A}$ only contains a finite number of A actions, we approximate a value function Q as a collection

*{daniele.calandriello,alessandro.lazaric}@inria.fr
†marcello.restelli@polimi.it

of A independent functions. In particular, we introduce a d_x-dimensional feature vector $\phi(\cdot) = [\varphi_1(\cdot), \varphi_2(\cdot), \dots, \varphi_{d_x}(\cdot)]^\mathsf{T}$, with $\sup_x \|\phi(x)\|_2 \leq L$. From ϕ we obtain a linear approximation space for action-value functions as $\mathcal{F} = \{f_w(x,a) = \phi(x)^\mathsf{T} w_a, x \in \mathcal{X}, a \in \mathcal{A}, w_a \in \mathbb{R}^{d_x}\}$. F$Q$I receives as input a fixed set of states $\mathcal{S} = \{x_i\}_{i=1}^{n_x}$ (*fixed design setting*) and the space $\mathcal{F}$. Starting from $w^0 = \mathbf{0}$ defining the function $\widehat{Q}^0$, at each iteration k, FQI first draws a (fresh) set of samples $(r_{i,a}^k, y_{i,a}^k)_{i=1}^{n_x}$ from the generative model of the MDP for each action $a \in \mathcal{A}$ on each of the states $\{x_i\}_{i=1}^{n_x}$ (i.e., $r_{i,a}^k = R(x_i,a)$ and $y_{i,a}^k \sim P(\cdot|x_i,a)$). From the samples, A independent training sets $\mathcal{D}_a^k = \{(x_i,a), z_{i,a}^k\}_{i=1}^{n_x}$ are generated, where $z_{i,a}^k = r_{i,a}^k + \gamma \max_{a'} \widehat{Q}^{k-1}(y_{i,a}^k, a')$, and $\widehat{Q}^{k-1}(y_{i,a}^k, a')$ is computed using the weight vector learned at the previous iteration as $\phi(y_{i,a}^k)^\mathsf{T} w_{a'}^{k-1}$. Then F$Q$I solves A linear regression problems, each fitting the training set $\mathcal{D}_a^k$ and returns vectors $\widehat{w}_a^k$ defining the action value function $f_{\widehat{w}^k}$ with $\widehat{w}^k = [\widehat{w}_1^k, \dots, \widehat{w}_A^k]$. The process is repeated until a fixed number of iterations K is reached or no significant change in the weight vector is observed. The performance of FQI is studied in detail in [8] and in [6, Thm. 5]. When moving to the multi-task setting, we consider different state sets $\{\mathcal{S}_t\}_{t=1}^T$ and we denote by $\widehat{W}_a^k \in \mathbb{R}^{d_x \times T}$ the matrix with vector $\widehat{w}_{a,t}^k \in \mathbb{R}^{d_x}$ as the t–th column. The structure of FQI in the multi-task setting is reported in Fig. 1.

input: Input sets $\left\{\mathcal{S}_t = \{x_i\}_{i=1}^{n_x}\right\}_{t=1}^T$, tol, K
Initialize $W^0 \leftarrow \mathbf{0}$, $k = 0$
do
$k \leftarrow k+1$
for $a \leftarrow 1, \dots, |\mathcal{A}|$ **do**
for $t \leftarrow 1, \dots, T$, $i \leftarrow 1, \dots, n_x$ **do**
Sample $r_{i,a,t}^k = R_t(x_{i,t}, a)$ and $y_{i,a,t}^k \sim P_t(\cdot|x_{i,t}, a)$
Compute $z_{i,a,t}^k = r_{i,a,t}^k + \gamma \max_{a'} \widetilde{Q}_t^k(y_{i,a,t}^k, a')$
end for
Build datasets $\mathcal{D}_{a,t}^k = \{(x_{i,t}, a), z_{i,a,t}^k\}_{i=1}^{n_x}$
Compute $\widehat{W}_a^k$ by solving Eqs. 1, 2, or 3 on $\{\mathcal{D}_{a,t}^k\}$
end for
while $\left(\max_a \left\|W_a^k - W_a^{k-1}\right\|_2 \geq tol\right)$ **and** $k < K$

Figure 1: Linear FQI with fixed design and fresh samples at each iteration in a multi-task setting.

High-dimensional regression. We consider a high-dimensional assumption, which guarantees that the target action-value functions that can be encountered over iterations all belong to the space $\mathcal{F}$. We define $w = [w_1, \dots, w_A]$ and $\psi(x,a) = [0_1, \dots, 0_{a-1}, \phi(x), 0_{a+1}, \dots, 0_A]$.

Assumption 1. *For any function $f_w \in \mathcal{F}$, the Bellman operator $\mathcal{T}$ can be expressed using matrix $P_\psi^{\pi_w}$ as $\mathcal{T} f_w(x,a) = R(x,a) + \gamma\, \mathbb{E}_{x' \sim P(\cdot|x,a)} \left[f_w(x', \pi_w(x'))\right] = \psi(x,a)^\mathsf{T} w^R + \gamma \psi(x,a)^\mathsf{T} P_\psi^{\pi_w} w$*

This assumption implies that $\mathcal{F}$ is closed w.r.t. the Bellman operator, since for any f_w, its image $\mathcal{T} f_w$ can be computed as the linear combination of $\psi(\cdot,\cdot)$ and $w^R + P_\psi^{\pi_w} w$. As a result, the optimal value function Q^* itself belongs to $\mathcal{F}$ and it can be computed as $\psi(x,a)^\mathsf{T} w^*$. This assumption encodes the intuition that in the high–dimensional feature space $\mathcal{F}$ induced by ψ, the transition kernel P, and therefore the system dynamics, can be expressed as a linear combination of the features using the matrix $P_\psi^{\pi_w}$, which depends on both function f_w and features ψ. This condition is usually satisfied whenever the space $\mathcal{F}$ is spanned by a very large set of features that allows it to approximate a wide range of different functions, including the reward and transition kernel. Under this assumption, at each iteration k of FQI, there exists a weight vector w^k such that $\mathcal{T}\widehat{Q}^{k-1} = f_{w^k}$.

LASSO-FQI. While the regression problem at each iteration could be solved using ordinary least-squares (OLS) regression, in the high-dimensional case the number of features exceeds the number of samples ($d > n$) and OLS would have a poor performance caused by overfitting. In this case, we propose three different methods constructed under different assumptions of the sparsity of the problem. First we consider the case when all the tasks are (independently) sparse and we use the ℓ_1 regularization of LASSO [5] to recover the sparsity as:

$$\text{LASSO:} \quad \widehat{w}_{a,t}^k = \arg\min_{w \in \mathbb{R}^{d_x}} \frac{1}{n_x} \sum_{i=1}^{n_x} \left(\phi(x_i)^\mathsf{T} w - z_{i,a}^k\right)^2 + \lambda ||w||_1. \tag{1}$$

When FQI is paired with LASSO, it can be shown that the sample complexity of the algorithm (i.e., the difference in performance between the optimal policy and the policy returned by FQI) is greatly improved w.r.t. OLS. In fact, whenever the target function $\mathcal{T}_t \widehat{Q}^{k-1}$ of task t is sparse with $s_t \ll d$ non-zero features, it can be shown that while FQI with OLS would suffer from an average loss over tasks of order $\tilde{O}(d/n)$, LASSO-FQI would dramatically reduces it to $\tilde{O}(\bar{s} \log d/n)$ for LASSO

where $\bar{s} = 1/T \sum_t s_t$ is the average sparsity, thus moving from a linear dependency on the number of features to a linear dependency only on the average number of features that are actually useful in approximating the target functions. The necessary assumptions and basic results are reported in [2] and they are extended to the iterative process of FQI in the longer version of the paper.

GL-FQI. Although effective in reducing the sample complexity in learning each task, LASSO-FQI does not exploit potential similarities between tasks. For this reason, we introduce the Group LASSO (GL) algorithm [5, 7], which is based on the assumption that the *relevant* features (i.e., with non-zero weights) are similar across different tasks (*shared-sparsity*) and thus the weight matrix $W \in \mathbb{R}^{d \times T}$ could have a small $\ell_{2,1}$-norm, where the ℓ_2-norm measures the "relevance" of feature i across tasks, while the ℓ_1-norm "counts" the total number of relevant features. The corresponding optimization problem solved at each iteration for each action $a \in \mathcal{A}$ is

$$\text{Group-LASSO:} \quad \widehat{W}_a^k = \arg \min_{W \in \mathbb{R}^{d_x \times T}} \sum_{t=1}^{T} \sum_{i=1}^{n_x} \left(\phi(x_{i,t})^\mathsf{T} w_t - z_{i,a,t}^k \right)^2 + \lambda \left\| W \right\|_{2,1}. \qquad (2)$$

In this case, following from [7], the performance loss is of order $\tilde{O}\big(\tilde{s}/n(1 + \log(d)/\sqrt{T})\big)$, where $\tilde{s}$ is the number of features which are active for *at least* one of the tasks. If the shared-sparsity assumption holds (i.e., the features with non-zero weights are the same across the tasks), we expect $\tilde{s}$ to be much smaller than d and the performance of the algorithm to improve w.r.t. running LASSO on each task separately. In fact, although $\bar{s} \leq \tilde{s}$, we notice that the dependency on d is further reduced and it eventually vanishes when the number of tasks increases. This shows that GL–FQI effectively leverages over the similarity across tasks and benefits from using all the samples from all the tasks, to better identify the *useful* features.

FL-FQI. Unlike other properties such as smoothness, the sparsity of a function is intrinsically related to the specific *representation* used to approximate it (i.e., the function space $\mathcal{F}$). As a result, the shared-sparsity assumption of GL may not necessarily hold for a given space $\mathcal{F}$. In this case, the shared sparsity $\tilde{s}$ may be of order of d and GL may perform significantly worse than LASSO-FQI itself. Thus, we finally introduce another algorithm that seeks to change the representation by applying a linear transformation of the features to achieve a high level of joint sparsity. This objective is pursued by the multi-task feature learning (MTFL) algorithm in [1], where it is shown that learning a suitable transformation is equivalent to solving the trace-norm regularized problem:

$$\text{FL:} \quad \widehat{W}_a^k = \arg \min_{W \in \mathbb{R}^{d_x \times T}} \sum_{t=1}^{T} \sum_{i=1}^{n_x} \left(\phi(x_{i,t})^\mathsf{T} w_t - z_{i,a,t}^k \right)^2 + \lambda \left\| W \right\|_{*}. \qquad (3)$$

While in GL–FQI we only need to learn the weight vectors, in [1] it is shown that the previous optimization problem is equivalent to learning the weights as well as a novel representation that maximizes the level of shared sparsity. While this extra complexity translates in a larger dependency on the number of features, it allows to exploit the shared sparsity of the best possible representation. Building on results from [9], we can prove that the performance loss of FL-FQI is of order $\tilde{O}\big(s^*/n(1 + d/T)\big)$, where s^* is the joint sparsity in the best representation. Whenever $s^* \ll d$, we can expect a significant improvement w.r.t. GL–FQI and LASSO-FQI as well. Finally, we notice that s^* can also be interpreted as the rank of the optimal weight matrix W_a^k and it corresponds to the number of *basis* tasks from which we can construct the weights of all the others.

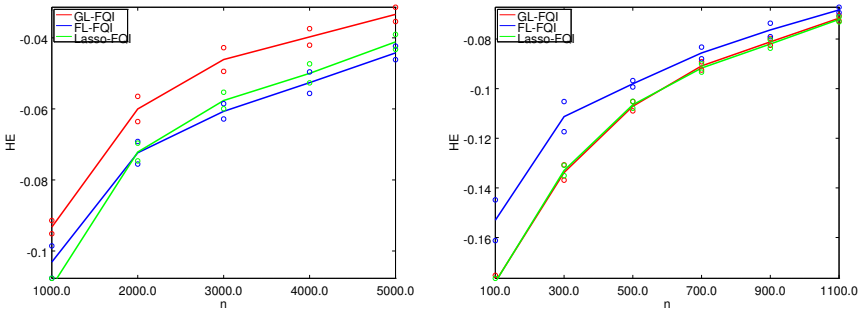
In summary, let K be the last iteration and π_t^K the greedy policy w.r.t. $\widehat{Q}_t^K$ for each task t, then the bounds on the performance loss $1/T \sum_t ||Q_t^* - Q^{\pi_t^K}||_{2,\rho}$ w.r.t. any arbitrary target distribution ρ for the three previous algorithms are

$$\text{LASSO-F}Q\text{I:} \ \tilde{O}\left(\frac{\bar{s}\log(d)}{n}\right); \quad \text{GL–F}Q\text{I:} \ \tilde{O}\left(\frac{\tilde{s}}{n}\left(1 + \frac{\log(d)}{\sqrt{T}}\right)\right); \quad \text{FL-F}Q\text{I:} \ \tilde{O}\left(\frac{s^*}{n}\left(1 + \frac{d}{T}\right)\right),$$

where $\bar{s} = 1/T \sum_t s_t$ is the average level of sparsity, $\tilde{s}$ is the joint sparsity, and s^* is the smallest number of relevant features that can be obtained by linear transformations of the features ϕ. The theoretical analysis of the three previous algorithms, reported in the long version of the paper, stems from the single-task setting of [8], and extends it to linear approximation and multi-task regression.

4 Experiments

In this section we validate the ideas of the previous algorithms and we verify the expected theoretical performance sketched in the previous section. We consider two variants of the blackjack domain.



(a) Full variant of blackjack. (b) Reduced variant of blackjack.

Figure 2: Comparison of FL-FQI, GL–FQI and LASSO–FQI. On the y axis we report the average house edge (HE) across tasks.

The player can choose to *hit* to obtain a new card or *stay* to end the episode. The two settings differ in the possibility of performing a *double* (doubling the bet) on the first turn. We refer to the variant with the *double* option as the *full variant*, while the other is the *reduced variant*. After the player concludes the episode, the dealer hits until a fixed threshold is reached or exceeded. Different tasks can be defined changing several parameters of the game, such as the number of decks, the value at which the dealer stays and whether she hits when the such value is research exactly with a *soft* hand.

Full variant experiment. The tasks are generated by selecting $2, 4, 6, 8$ decks, by setting the stay threshold at $\{16, 17\}$ and whether the dealer hits on soft, for a total of 16 tasks. We define a very rich description of the state space with the objective of satisfying the high-dimensional Assumption 1. At the same time, this is likely to come with a large number of useless features, which makes it suitable for sparsification. In particular, we include the player hand value, indicator functions for each possible player hand value and dealer hand value, and a large description of the cards not dealt yet (corresponding to the history of the game) under the form of indicator functions for various ranges. In total, the representation contains $d = 212$ features. We notice that although none of the features is completely useless (as requested in the definition of sparsity), the features related with the history of the game are unlikely to be very useful for most of the tasks defined in this experiment. We collect samples from up to 5000 episodes, although they may not be representative enough given the large state space of all possible histories that the player can encounter and the high stochasticity of the game. The evaluation is performed by simulating the learned policy for 2,000,000 episodes and computing the average House Edge (HE) across tasks. For each algorithm we report the performance for the best regularization parameter λ in the range $\{2, 5, 10, 20, 50\}$. Results are reported in Fig. 2a. Although the set of features is quite large, we notice that all the algorithms succeed in learning a good policy even with relatively few samples, showing that all of them can take advantage of the sparsity of the representation. In particular, GL–FQI exploits the fact that all 16 tasks share the same useless features (although the set of useful features may not overlap entirely) and its performance is the best. On the other hand, FL–FQI suffers from the increased complexity of representation learning, which in this case does not lead to any benefit since the initial representation is already sparse ($\bar{s} \approx \tilde{s} \approx s^*$). Nonetheless, it is interesting to note that the performance of FL–FQI is comparable to single-task LASSO–FQI.

Reduced variant experiment. In the second experiment we construct a representation for which we expect the weight matrix to be dense. In particular, we only consider the value of the player's hand and of the dealer's hand and we generate features as the Cartesian product of these two discrete variables plus a feature indicating whether the hand is soft, for a total of 280 features. Similar to the previous setting, the tasks are generated with $2, 4, 6, 8$ decks, whether the dealer hits on soft, and a larger number of stay thresholds in $\{15, 16, 17, 18\}$, for a total of 32 tasks. We used regularizers in the range $\{0.1, 1, 2, 5, 10\}$. Since the history is not included, the different number of decks influences only the probability distribution of the totals. Moreover, limiting the actions to either *hit* or *stay* further increases the similarity among tasks. Therefore, we expect to be able to find a dense, low-rank solution ($s^* \ll \tilde{s}$). The results in Fig. 2b confirms this guess, with FL–FQI performing significantly better than the other methods. In addition, GL–FQI and LASSO–FQI perform similarly, since the dense representation penalizes both single-task and shared sparsity. This was also observed by the fact that both methods favor low values of λ, indicating that the sparse-inducing penalties are not effective.

References

[1] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.

[2] Peter J Bickel, Ya'acov Ritov, and Alexandre B Tsybakov. Simultaneous analysis of lasso and dantzig selector. *The Annals of Statistics*, pages 1705–1732, 2009.

[3] Daniele Calandriello, Alessandro Lazaric, and Marcello Restelli. Sparse Multi-task Reinforcement Learning. In *https://hal.inria.fr/hal-01073513*, 2014.

[4] Damien Ernst, Pierre Geurts, Louis Wehenkel, and Michael L Littman. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research*, 6(4), 2005.

[5] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning*. Springer, 2009.

[6] Alessandro Lazaric and Marcello Restelli. Transfer from multiple MDPs. In *Proceedings of the Twenty-Fifth Annual Conference on Neural Information Processing Systems (NIPS'11)*, 2011.

[7] Karim Lounici, Massimiliano Pontil, Sara Van De Geer, Alexandre B Tsybakov, et al. Oracle inequalities and optimal inference under group sparsity. *The Annals of Statistics*, 39(4):2164–2204, 2011.

[8] Rémi Munos and Csaba Szepesvári. Finite-time bounds for fitted value iteration. *The Journal of Machine Learning Research*, 9:815–857, 2008.

[9] Sahand Negahban, Martin J Wainwright, et al. Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *The Annals of Statistics*, 39(2):1069–1097, 2011.