Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence

Tree-Based On-Line Reinforcement Learning

André da Motta Salles Barreto
Laboratório Nacional de Computação Científica
Petrópolis, RJ, Brazil

Abstract

Fitted Q-iteration (FQI) stands out among reinforcement learning algorithms for its flexibility and ease of use. FQI can be combined with any regression method, and this choice determines the algorithm's statistical and computational properties. The combination of FQI with an ensemble of regression trees gives rise to an algorithm, FQIT, that is computationally efficient, scalable to high dimensional spaces, and robust to noise. Despite its nice properties and good performance in practice, FQIT also has some limitations: the fact that an ensemble of trees must be constructed (or updated) at each iteration confines the algorithm to the batch scenario. This paper aims to address this specific issue. Based on a strategy recently proposed in the literature, called the stochastic-factorization trick, we propose a modification of FQIT that makes it fully incremental, and thus suitable for on-line learning. We call the resulting method tree-based stochastic factorization (TBSF). We derive upper bounds for the difference between the value functions computed by FQIT and TBSF, and also show in which circumstances the approximations coincide. A series of computational experiments is presented to illustrate the properties of TBSF and to show its usefulness in practice, including a medical problem involving the treatment of patients infected with HIV.

1 Introduction

Batch reinforcement-learning algorithms learn a decision policy based on a fixed set of sample transitions. Among them, fitted Q-iteration (FQI) stands out for its flexibility and ease of use. Probably because of this, FQI has been adopted by researches and practitioners, and today it is possible to find several reports of successful applications of the algorithm (Ernst, Geurts, and Wehenkel 2005; Riedmiller 2005; Ernst et al. 2006; Kalyanakrishnan and Stone 2007).

FQI can be combined with any regression method, and this choice determines the algorithm's statistical and computational properties. In their original work, Ernst, Geurts, and Wehenkel (2005) suggest the use of an ensemble of regression trees. The authors point out several advantages of these models, such as their computational efficiency, their scalability to high dimensional spaces, and their robustness with respect to irrelevant variables, outliers, and noise. The combination of FQI with regression trees (FQIT) has also been extensively tested in practice, in general with very good results (Ernst, Geurts, and Wehenkel 2005; Ernst et al. 2006).

Despite its nice properties and good performance in practice, FQIT also has some limitations: the fact that an ensemble of trees must be constructed (or updated) at each iteration confines the algorithm to the batch scenario. This paper aims to address this specific issue. Based on a strategy recently proposed in the literature, called the stochastic-factorization trick, we propose a modification of FQIT that makes its computational complexity independent of the number of sample transitions. The resulting method, tree-based stochastic factorization (TBSF), is significantly faster than its precursor and can be used on-line.

We derive TBSF and analyze it theoretically and empirically. We prove that the distance between the approximations computed by FQIT and TBSF is bounded. We also show empirically that, since our algorithm can process more sample transitions in the same amount of time, it can match FQIT's performance using less computation.

2 Reinforcement Learning

We consider the basic framework of reinforcement learning: an agent interacts with an environment and selects actions in order to maximize the amount of reward received in the long run (Sutton and Barto 1998). As usual, we assume that this interaction can be modeled as a *Markov decision process* (MDP, Puterman, 1994). An MDP is a tuple $M \equiv (S, A, P^a, r^a, \gamma)$, where S is the state space and A is the (finite) action set. For each action $a \in A$, $P^a(\cdot|s)$ is the next-state distribution upon taking action a in state s. The reward received at transition $s \xrightarrow{a} s'$ is given by $R^a(s, s')$. Usually, one is interested in the expected reward resulting from the execution of a in s, that is, $r^a(s) = E_{s' \sim P^a(\cdot|s)}\{R^a(s, s')\}$. The discount factor $\gamma \in [0, 1)$ gives smaller weights to rewards received further in the future.

The algorithm presented in this paper can be applied to MDPs with continuous and discrete state spaces, and in both cases the strategy will be to derive a discrete model. When both the state and action spaces are finite, an MDP can be

represented in matrix form: each function P^a becomes a matrix $\mathbf{P}^a \in \mathbb{R}^{|S|\times|S|}$, with $p^a_{ij} = P^a(s_j|s_i)$, and each function r^a becomes a vector $\mathbf{r}^a \in \mathbb{R}^{|S|}$, with $r^a_i = r^a(s_i)$.[1]

The goal of the agent is to find an optimal *policy*, that is, a mapping $\pi^* : S \mapsto A$ that maximizes the expected sum of discounted rewards. When the MDP is finite, dynamic programming can be used to find an optimal policy $\pi^* \in A^{|S|}$ in polynomial time (Littman, Dean, and Kaelbling 1995). Starting from $Q_0 = 0$, the *value-iteration* update rule, $Q_{t+1}(s_i, a) = r^a(s_i) + \gamma \sum_{j=1}^{|S|} p^a_{ij} \max_{b\in A} Q_t(s_j, b)$, with $t > 0$, gives the optimal t-step *action-value function*, from which the t-step policy can be obtained by selecting any $\pi^{(t)}(s_i) \in \operatorname{argmax}_a Q_t(s_i, a)$. As $t \to \infty$ the function $Q_t(s_i, a)$ approaches $Q^*(s_i, a)$, the value function associated with all optimal policies π^*.

In reinforcement learning it is assumed that the MDP is unknown, and the agent must learn a policy based on transitions sampled from the environment. If the process of learning a decision policy is based on a fixed set of sample transitions, we call it *batch* reinforcement learning. In *on-line* reinforcement learning the computation of a decision policy takes place concomitantly with the collection of data.

3 Fitted Q-Iteration

Fitted Q-iteration (FQI) is a batch reinforcement-learning algorithm proposed by Ernst, Geurts, and Wehenkel (2005) based on the works by Gordon (1995) and Ormoneit and Sen (2002). Let $S^a \equiv \{(s^a_i, r^a_i, \hat{s}^a_i)\}$, $i = 1, 2, ..., n_a$, be sample transitions associated with action $a \in A$, where $s^a_i, \hat{s}^a_i \in S$ and $r^a_i \in \mathbb{R}$. FQI keeps an approximation $\hat{Q} : S \times A \mapsto \mathbb{R}$ which is successively updated in the following way. Starting from $\hat{Q}_0 = 0$, at each iteration $t > 0$ a training set $H_t \equiv \{[(s^a_i, a), \mathcal{Q}_t(s^a_i, a)]\}$ composed of $n = \sum_{a\in A} n_a$ input-output pairs is constructed, with $\mathcal{Q}_t(s^a_i, a) = r^a_i + \gamma \max_b \hat{Q}_{t-1}(\hat{s}_i, b)$. Then, a regression procedure uses this training set to compute the next approximation, $\hat{Q}_t = \mathrm{fit}(H_t)$.

The combination of the function approximator $\hat{Q}(s, a)$ with the regression method $\mathrm{fit}(\cdot)$ gives rise to different instances of FQI. Here we will focus on approximators of the form $\hat{Q}(s, a) = \sum_{b\in A} \sum_{j=1}^{n_b} \kappa^a(s, s^b_j) \hat{Q}(s^b_j, a)$, where $\kappa^a(s, s^b_j) : S \times S \mapsto \mathbb{R}$ can be intuitively seen as a measure of the similarity between s and s^b_j. We will assume that $\kappa^a(s, s^b_j) = 0$ if $a \neq b$, which simplifies the expression above to

$$\hat{Q}(s, a) = \sum_{j=1}^{n_a} \kappa^a(s, s^a_j) \hat{Q}(s^a_j, a). \tag{1}$$

When $\hat{Q}_t(s, a)$ satisfies (1) it is fully defined by the values of the states s^a_i; hence, FQI's regression step reduces to updating these values:

$$\hat{Q}_{t+1}(s^a_i, a) = \mathcal{Q}_{t+1}(s^a_i, a) = r^a_i + \gamma \max_b \hat{Q}_t(\hat{s}^a_i, b). \tag{2}$$

[1]Throughout the paper vectors will be denoted by small boldface letters and matrices will be denoted by capital boldface letters.

Combining (2) and (1), we can see that

$$\hat{Q}_t(\hat{s}^a_i, b) = \sum_{j=1}^{n_b} \kappa^b(\hat{s}^a_i, s^b_j) \left[r^b_j + \gamma \max_c \hat{Q}_{t-1}(\hat{s}^b_j, c) \right]. \tag{3}$$

It is well known that if

$$\kappa^b(\hat{s}^a_i, s^b_j) \geq 0 \text{ and } \sum_{j=1}^{n_b} \kappa^b(\hat{s}^a_i, s^b_j) = 1, \tag{4}$$

then $\hat{Q}_t$ converges to a fixed point as $t \to \infty$ (Gordon 1995; Ormoneit and Sen 2002). This is easy to see if we recast FQI as the solution of a derived MDP as follows. Let $\hat{S}$ be the set composed of the n states $\hat{s}^a_i$. Define

$$\hat{R}^a(\hat{s}^b_i, \hat{s}^c_j) = r^c_j \text{ and } \hat{P}^a(\hat{s}^c_j|\hat{s}^b_i) = \kappa^a(\hat{s}^b_i, s^c_j). \tag{5}$$

Comparing (3) and (5), it is easy to see that FQI is equivalent to value iteration in the MDP $\hat{M} \equiv (\hat{S}, A, \hat{P}^a, \hat{r}^a, \gamma) = (\hat{S}, A, \hat{\mathbf{P}}^a, \hat{\mathbf{r}}^a, \gamma)$, where $\hat{r}^a(\hat{s}^b_i) = \sum_{c\in A} \sum_{j=1}^{n_c} \kappa^a(\hat{s}^b_i, s^c_j) r^c_j = \sum_{j=1}^{n_a} \kappa^a(\hat{s}^b_i, s^a_j) r^a_j$.

Antos, Munos, and Szepesvári (2007) have analyzed the finite-sample performance of FQI for the case in which the class of policies is restricted and the regression procedure $\mathrm{fit}(\cdot)$ is a weighted least-squares method. Farahmand et al. (2009) provide similar analysis for the scenario where the least-squares problem is regularized. Here we will focus on the version of FQI originally proposed by Ernst, Geurts, and Wehenkel (2005), which we discuss next.

Fitted Q-Iteration Using Trees

Ernst, Geurts, and Wehenkel (2005) suggest that the approximation $\hat{Q}(s, a)$ be represented by an ensemble of regression trees. Besides all the advantages discussed in Section 1, ensembles of trees can be defined in order to satisfy (1) and (4).

Let T^a be a set of $|T^a|$ trees T^a_i associated with action $a \in A$. Each T^a_i partitions the state space in m^a_i sets T^a_{ij}. The value associated with partition T^a_{ij} at iteration t is simply the average of the values of states s^a_l that belong to T^a_{ij}, that is:

$$\hat{v}_t(T^a_{ij}) = \frac{\sum_{l=1}^{n_a} I\{s^a_l \in T^a_{ij}\} \mathcal{Q}_t(s^a_l, a)}{\sum_{l=1}^{n_a} I\{s^a_l \in T^a_{ij}\}}, \tag{6}$$

where $I\{\text{true}\} = 1$ and $I\{\text{false}\} = 0$ is the indicator function. We define $\sum_{l=1}^{n_a} I\{s^a_l \in T^a_{ij}\} = |T^a_{ij}|$. Recall that, when using an ensemble of trees, the value of a state $\hat{s}^a_i$ is the average value of all partitions $\hat{s}^a_i$ belongs to, that is:

$$\begin{aligned} \hat{Q}_t(\hat{s}^a_i, b) &= \frac{1}{|T^b|} \sum_{l=1}^{|T^b|} \sum_{u=1}^{m^b_l} I\{\hat{s}^a_i \in T^b_{lu}\} \hat{v}(T^b_{lu}) \\ &= \frac{1}{|T^b|} \sum_{l=1}^{|T^b|} \sum_{u=1}^{m^b_l} \frac{1}{|T^b_{lu}|} I\{\hat{s}^a_i \in T^b_{lu}\} \sum_{j=1}^{n_b} I\{s^b_j \in T^b_{lu}\} \mathcal{Q}_t(s^b_j, b). \end{aligned} \tag{7}$$

There are several methods available to build a regression tree T^a based on a training set H (see Ernst, Geurts, and Wehenkel's article and references therein, 2005). If we assume that the structure of the trees is fixed throughout FQIT's iterations, then we can look at them as just a specific way of

defining $\kappa^b(\hat{s}_i^a, s_j^b)$. Comparing (1), (2), and (7), we see that when using an ensemble of trees

$$\kappa^b(\hat{s}_i^a, s_j^b) = \frac{1}{|T^b|}\sum_{l=1}^{|T^b|}\sum_{u=1}^{m_l^b}\frac{1}{|T_{lu}^b|}I\{\hat{s}_i^a \in T_{lu}^b \text{ and } s_j^b \in T_{lu}^b\}. \tag{8}$$

It is clear that $\kappa^b(\hat{s}_i^a, s_j^b) \geq 0$. Also, if $\hat{s}_i^a \in T_{lu}^b$, the summation $\sum_{j=1}^{n_b} I\{\hat{s}_i^a \in T_{lu}^b \text{ and } s_j^b \in T_{lu}^b\} = |T_{lu}^b|$. Since there are $|T^b|$ such summations, we have $\sum_{j=1}^{n_b}\kappa^b(\hat{s}_i^a, s_j^b) = 1$. Therefore, $\kappa^b(\hat{s}_i^a, s_j^b)$ satisfies (4).

The fact that each FQI iteration involves solving a regression problem with n training examples makes it impractical for larger domains, depending on the choice of regression method $\mathrm{fit}(\cdot)$ and the computational resources available. Moreover, even if $\hat{Q}$ is updated through a simple rule like (2), the computational complexity of each FQI iteration will necessarily depend on n, which clearly precludes the use of the algorithm as an on-line method. In the next section we discuss a possible way to circumvent these limitations.

4 Tree-Based Stochastic Factorization

In order to derive our algorithm we will resort to the "stochastic-factorization trick," an idea introduced by Barreto and Fragoso (2011) and later explored by Barreto, Precup, and Pineau (2011; 2012) in reinforcement learning.

Given a finite MDP $M \equiv (S, A, \mathbf{P}^a, \mathbf{r}^a, \gamma)$, let $\mathbf{P}^a = \mathbf{D}^a\mathbf{K}^a$ be $|A|$ factorizations in which $\mathbf{D}^a \in \mathbb{R}^{n\times m}$ and $\mathbf{K}^a \in \mathbb{R}^{m\times n}$ are stochastic matrices. Also, let $\mathbf{D}^a\bar{\mathbf{r}}^a = \mathbf{r}^a$ for all a. Then, if we swap the factors $\mathbf{D}^a$ and $\mathbf{K}^a$, we obtain another transition matrix $\bar{\mathbf{P}}^a = \mathbf{K}^a\mathbf{D}^a$ that retains some basic properties of $\mathbf{P}^a$ (Barreto and Fragoso 2011). The insight is that, instead of solving M, one can solve $\bar{M} \equiv (\bar{X}, A, \bar{\mathbf{P}}^a, \bar{\mathbf{r}}^a, \gamma)$, where $\bar{X}$ is a set of m states that will be defined shortly. Based on the optimal value function of $\bar{M}$, $\bar{Q}^*$, an approximation of Q^* can be computed as $\tilde{Q}(s_i, a) = \sum_{j=1}^m d_{ij}\bar{Q}^*(\bar{x}_j, a)$. When $m \ll n$, replacing M with $\bar{M}$ significantly reduces memory usage and computing time.

The stochastic-factorization trick can be applied to any finite MDP. Since when (1) and (4) hold FQI is equivalent to the solution of a discrete MDP $\hat{M}$, as discussed in Section 3, in principle the technique described above can be used with any approximator $\hat{Q}$ that has these properties. Note though that, depending on the choice of $\hat{Q}$, it is not clear how to efficiently compute $\mathbf{D}^a\mathbf{K}^a = \hat{\mathbf{P}}^a$ and $\mathbf{D}^a\bar{\mathbf{r}}^a = \hat{\mathbf{r}}^a$ (see (5)). One possible strategy is to select a set of m representative states $\bar{s}_l \in S$ and use the approximation $\kappa^b(\hat{s}_i^a, s_j^b) \approx \sum_{l=1}^m \kappa(\hat{s}_i^a, \bar{s}_l)\kappa^b(\bar{s}_l s_j^b)$ (Barreto, Precup, and Pineau 2011). Although this is a valid strategy in general, it only provides an approximate factorization of the MDP. Besides, one has to find a way to appropriately define the representative states $\bar{s}_l$. We now show that, in contrast, in the particular case in which $\hat{Q}$ is represented by an ensemble of trees, it is possible to determine an *exact* factorization of the MDP $\hat{M}$ without any computation.

Stochastic Factorization

In order to derive our algorithm we will need two assumptions:

(i) FQIT builds the ensembles of trees at iteration $t = 1$ using any algorithm and then keeps the structure of the trees fixed for $t > 1$. This means that the partitions T_{ij}^a remain the same throughout the execution of the algorithm, although their values $\hat{v}(T_{ij}^a)$ may change.

(ii) The number of *partitions* in the ensembles T^a is the same for all a, that is, $\sum_{l=1}^{|T^a|} m_l^a = m$, $a \in A$.

As mentioned before, Assumption (i) is needed to guarantee that the approximations $\hat{Q}(\hat{s}^a, b)$ are updated through (2). Changing the structure of the trees from one iteration to another corresponds to defining a new MDP $\hat{M}_t$ at each iteration t, and thus the guarantee of convergence to a fixed point is lost. We will discuss how to relax Assumption (i) later. Assumption (ii) is necessary to guarantee that the MDP $\bar{M}$ is well defined, as will become clear shortly.

In order to facilitate the exposition, we will refer to s_i^a as s_j, with $j = \sum_{b=1}^{a-1} n_b + i$ (the same for r_i^a and $\hat{s}_i^a$). Similarly, we will use $\bar{T}_k^a$ to refer to T_{ij}^a, with $k = \sum_{l=1}^{i-1} m_l^a + j$. Finally, we introduce the function $\mathrm{act} : \{1, 2, ..., n\} \mapsto \{1, 2, ..., |A|\}$ as $\mathrm{act}(i) = 1 + \{\max a \text{ such that } \sum_{b=1}^a n_b < i\}$—that is, $\mathrm{act}(i)$ is the action associated with s_i, r_i, and $\hat{s}_i$. Using these definitions and (5) and (8) we can write

$$\hat{p}_{ij}^a = \sum_{l=1}^m \frac{1}{|T^a|}I\{\hat{s}_i \in \bar{T}_l^a\}\frac{1}{|\bar{T}_l^a|}I\{\mathrm{act}(j) = a \text{ and } s_j \in \bar{T}_l^a\}.$$

Thus, if we define the elements of $\mathbf{D}^a$ and $\mathbf{K}^a$ as

$$\begin{aligned} d_{ij}^a &= \frac{1}{|T^a|}I\{\hat{s}_i \in \bar{T}_j^a\} \quad \text{and} \\ k_{ij}^a &= \frac{1}{|\bar{T}_i^a|}I\{\mathrm{act}(j) = a \text{ and } s_j \in \bar{T}_i^a\}, \end{aligned} \tag{9}$$

it is clear that $\mathbf{D}^a\mathbf{K}^a = \mathbf{P}^a$ for all a. In order to have a complete factorization of the MDP $\hat{M}$, it remains to find vectors $\bar{\mathbf{r}}^a$ such that $\mathbf{D}^a\bar{\mathbf{r}}^a = \hat{\mathbf{r}}^a$ for all $a \in A$. We know from (5) that $\hat{r}^a(\hat{s}_i) = \sum_{j=1}^n \kappa^a(\hat{s}_i, s_j)r_j$, which implies that $\hat{\mathbf{r}}^a = \hat{\mathbf{P}}^a\mathbf{r}$ ($\mathbf{r} \in \mathbb{R}^n$ is the vector composed of the sampled rewards). Now, if we define $\bar{\mathbf{r}}^a = \mathbf{K}^a\mathbf{r}$, we can write $\hat{\mathbf{r}}^a = \hat{\mathbf{P}}^a\mathbf{r} = \mathbf{D}^a\mathbf{K}^a\mathbf{r} = \mathbf{D}^a\bar{\mathbf{r}}^a$, and we have an exact factorization of $\hat{M}$.

Once we have determined $\mathbf{D}^a$, $\mathbf{K}^a$, and $\bar{\mathbf{r}}^a$, we can apply the stochastic-factorization trick to the MDP $\hat{M}$. Let $\bar{\mathbf{P}}^a = \mathbf{K}^a\mathbf{D}^a$ for all $a \in A$. From (9), we have

$$\bar{p}_{ij}^a = \frac{1}{|T^a||\bar{T}_i^a|}\sum_{l=1}^n I\{\mathrm{act}(l) = a \text{ and } s_l \in \bar{T}_i^a \text{ and } \hat{s}_l \in \bar{T}_j^a\}. \tag{10}$$

Thus, we simply count the number of transitions in S^a that started in $\bar{T}_i^a$ and divide it by the number of transitions that started in this partition and ended in $\bar{T}_j^a$. Since the states s_i and $\hat{s}_i$ simultaneously belong to $|T^a|$ partitions $\bar{T}_i^a$, in order to obtain $\bar{p}_{ij}^a$ this fraction must be further divided by $|T^a|$. As for the rewards, from the definition of $\bar{\mathbf{r}}^a$ we have

$$\bar{r}_i^a = \frac{1}{|\bar{T}_i^a|}\sum_{j=1}^n I\{\mathrm{act}(j) = a \text{ and } s_j \in \bar{T}_i^a\}r_j^a, \tag{11}$$

which is simply the average of the rewards r_j^a associated with states s_j that belong to $\bar{T}_i^a$. We see an interesting inversion here: while in the MDP $\hat{M}$ the reward function $\hat{R}^a(\hat{s}_i, \hat{s}_j) = r_j$ is defined by the end-state $\hat{s}_j$, as shown in (5), here $\bar{r}_i^a$ only depends on the initial states s_i.

Because of Assumption (ii), we know that $\bar{\mathbf{P}}^a \in \mathbb{R}^{m\times m}$ and $\bar{\mathbf{r}}^a \in \mathbb{R}^{m\times m}$ for all $a \in A$. Thus, we can define the MDP $\bar{M} \equiv (\bar{X}, A, \bar{\mathbf{P}}^a, \bar{\mathbf{r}}^a, \gamma)$. Note that, unlike Barreto, Precup, and Pineau (2011), we do not explicitly define $\bar{X} \subset S$ as a set of representative states. In the particular case in which the trees in all ensembles T^a have the same structure—that is, T_i^a and T_i^b partition S in the same way—we can think of the partitions $\bar{T}_i$ as the states of $\bar{M}$. In general, though, the states $\bar{x}_i \in \bar{X}$ are not subsets of S.

Algorithm

As discussed, when Assumption (i) holds, FQIT builds the MDP $\hat{M}$ at iteration $t = 1$ and then solves it iteratively through value iteration. Based on (10) and (11) we can define an algorithm that does the same using $\bar{M}$ instead of $\hat{M}$. At any iteration t, the value function $\hat{Q}_t(\hat{s}_i, a)$ computed by FQIT, given by (7), can be approximated as $\tilde{Q}_t(\hat{s}_i, a) = \sum_{j=1}^{m} d_{ij}^a \bar{Q}_t(\bar{x}_j, a)$. More generally, from the definition of $\mathbf{D}^a$ in (9) we see that the action-values of a state s not necessarily in the sets S^a can be computed as $\tilde{Q}_t(s, a) = 1/|T^a| \sum_{j=1}^{m} I\{s \in \bar{T}_j^a\}\bar{Q}_t(\bar{x}_j, a)$. We call the proposed algorithm *tree-based stochastic factorization* (TBSF).

One immediate advantage of replacing FQIT with TBSF is the potential reduction on the computational cost. Let $\eta_{\min}$ denote the minimum number of points required to split a node during the construction of the trees (see Section 6). Each iteration of FQIT involves the construction (or update) of $|A|$ ensembles of trees, each one requiring at least $O(|T^a| n_a \log(n_a/\eta_{\min}))$ operations, and the improvement of the current decision policy, which is $O(n|A|)$. In contrast, TBSF executes $O(|T^a| n_a \log(n_a/\eta_{\min}))$ operations only once per action, to build $\bar{\mathbf{P}}^a$ and $\bar{\mathbf{r}}^a$, and then performs only $O(m^2|A|)$ operations in each iteration—assuming the worst-case scenario where the matrices $\bar{\mathbf{P}}^a$ are dense. As one can see, when $m \ll n$, replacing FQIT with TBSF can result in significant gains in terms of computation.

There is however a stronger reason to adopt TBSF instead of FQIT. Suppose that at iteration t a new batch of sample transitions $S_2^a \equiv \{(s_i^a, r_i^a, \hat{s}_i^a)\}$ becomes available, with $a \in A$. One way of incorporating the new data into FQIT's approximation is to set $\hat{Q}_t(\hat{s}_i^a, a) = 0$ for all states $\hat{s}_i^a \in S_2^a$ and then proceed normally from there. However, since FQIT's mechanics require that all transitions used in the approximation are available, as shown in (2), there is an inbuilt limit on the amount of data that this algorithm can process. This is why FQIT is inherently a batch-mode algorithm. In contrast, TBSF can extract the information contained in the sample transitions and then discard them, which means that it can also be used in the on-line regime.

Let $S_1 \equiv S^a$ and $S_2 \equiv S_2^a$ (we drop the "a" superscript in the sets S^a and S_2^a to improve readability). Let $\bar{\mathbf{P}}^{S_1}$ and $\bar{\mathbf{r}}^{S_1}$ be matrix $\bar{\mathbf{P}}^a$ and vector $\bar{\mathbf{r}}^a$ computed by TBSF through (10) and (11) using only the n_a transitions in S_1. We assume that we have already discarded the data in S_1 and want to compute $\bar{\mathbf{P}}^{S_1\cup S_2}$ and $\bar{\mathbf{r}}^{S_1\cup S_2}$ from $\bar{\mathbf{P}}^{S_1}$, $\bar{\mathbf{r}}^{S_1}$, and S_2.

To have an algorithm that is as general as possible, we consider the possibility that the ensembles change with the new data, either because new partitions are added to the trees already in the ensemble or because new trees are added altogether. We denote the two ensembles by T^{S_1} and $T^{S_1\cup S_2}$. We also define m_i' as the number of partitions associated with the i^{th} tree of $T^{S_1\cup S_2}$ and set $m' = \sum_{i=1}^{|T^{S_1\cup S_2}|} m_i'$. If $m' > m$, we set $\bar{r}_l^a = 0$ and $\bar{p}_{lj}^a = \bar{p}_{jl}^a = 0$ for $l = m+1, m+2, ..., m'$, $j = 1, 2, ..., m'$, and all $a \in A$.

We start by rewriting (10) as $\bar{p}_{ij}^{S_1} = z_{ij}^{S_1}/(|T^{S_1}| z_i^{S_1})$, where $z_i^{S_1} = \sum_{l=1}^{n_a} I\{s_l \in T_i^a\}$ and $z_{ij}^{S_1} = \sum_{l=1}^{n_a} I\{s_l \in T_i^a \text{ and } \hat{s}_l \in T_j^a\}$ (note that $\text{act}(l) = a$ for all l). Then, if we define the variables $z_i^{S_2}$ and $z_{ij}^{S_2}$ in an analogous way, we have

$$\bar{p}_{ij}^{S_1\cup S_2} = (z_{ij}^{S_1} + z_{ij}^{S_2}) / \left[|T^{S_1\cup S_2}|(z_i^{S_1} + z_i^{S_2})\right]. \quad (12)$$

In order to avoid keeping $z_{ij}^{S_1}$ for all i, j, we rewrite (12) as

$$\bar{p}_{ij}^{S_1\cup S_2} = \frac{\bar{p}_{ij}^{S_1}|T^{S_1}| z_i^{S_1} + z_{ij}^{S_2}}{|T^{S_1\cup S_2}|(z_i^{S_1} + z_i^{S_2})}. \quad (13)$$

Similarly, if we define $b_i^{S_1} = \sum_{j=1}^{n_a} I\{s_j \in T_i^{S_1}\} r_j$ and proceed in the same way as above starting from (11), we arrive at the update rule

$$\bar{r}_i^{S_1\cup S_2} = \frac{\bar{r}_i^{S_1} z_i^{S_1} + b_i^{S_2}}{z_i^{S_1} + z_i^{S_2}}. \quad (14)$$

Since $z_i^{S_2}$, $z_{ij}^{S_2}$, and $b_i^{S_2}$ are computed based on S_2, and $|T^{S_1\cup S_2}|$ is readily available, in order to have an incremental algorithm we only have to keep $|T^{S_1}|$ and $z_i^{S_1}$ for $i = 1, 2, ..., m$. Note that, when $T^{S_1} = T^{S_1\cup S_2}$, the model computed through (13) and (14) is the same that would be built if the transitions in $S_1 \cup S_2$ were used all at once.

Algorithm 1 shows a generic step by step description of TBSF. As one can see, several variations of the algorithm are possible. If the variable $\text{it}_{\max}$ defined in line 3 is set to 1, TBSF reduces to a batch method; otherwise it is incremental. If $\text{it}_{\max} > 1$ and the distribution μ is updated based on $\tilde{Q}$—line 8—TBSF is an on-line method. If in addition $n = 1$, we have an algorithm that does not store sample transitions (line 4). Finally, if in line 5 new trees or new partitions are created with $\text{it} > 1$, we have an adaptive method (this is akin to relaxing Assumption (i) for FQIT). It is also trivial to modify the algorithm for the scenario where the sample transitions are given.

5 Theoretical Analysis

In this section we analyze some properties of TBSF. We will focus on the batch version of the algorithm, that is, the case in which $\text{it}_{\max} = 1$ in Algorithm 1. We will assume that Assumptions (i) and (ii) hold, and compare the approximation $\hat{Q}_t$ computed by FQTI after t iterations with the approximation $\tilde{Q}_t$ computed by TBSF after t applications of the value-iteration update rule in $\bar{M}$ (line 7 of Algorithm 1). Our first result depends on the following assumption:

Algorithm 1 TBSF

Input: fit ▷ Algorithm to construct the trees
μ ▷ Distribution over $S \times A$
Output: Approximate value function $\tilde{Q}$
for it $\leftarrow 1, 2, ..., \text{it}_{\max}$ **do**
  Collect n transitions based on μ and store in S^a
  $T^a \leftarrow \text{fit}(S^a)$ for all $a \in A$ ▷ Construct or grow
  Update $\bar{M}$ using (13) and (14)
  Apply t iterations of value iteration to $\bar{Q}$
  Modify μ based on $\tilde{Q}(s,a)$ ▷ Optional
  $S^a \leftarrow \emptyset$ for all $a \in A$ ▷ Discard transitions

(iii) The number of trees in the ensembles is the same, that is, $|T^a| = |T|$ for all $a \in A$.

Note that the structure of the trees can differ. Let $\bar{\mathbf{r}}^+$ be the vector whose elements are $\bar{r}_i^+ = \max_{a \in A} \bar{r}_i^a$. Analogously, let $\bar{\mathbf{r}}^-$ be the vector with entries $\bar{r}_i^- = \min_{a \in A} \bar{r}_i^a$. Let $\bar{R}^+_{\max}$ be the sum of the $|T|$ largest elements of $\bar{\mathbf{r}}^+$ and let $\bar{R}^-_{\min}$ be the sum of the $|T|$ smallest elements of $\bar{\mathbf{r}}^-$. Then, we can show the following:

Proposition 1 *For* $t > 0$, $|\hat{Q}_t(\hat{s}_i, a) - \tilde{Q}_t(\hat{s}_i, a)| \leq \frac{1}{|T|} \frac{\gamma - \gamma^t}{1 - \gamma} \left(\bar{R}^+_{\max} - \bar{R}^-_{\min}\right)$ *for any* $\hat{s}_i$ *and any* a.

The proof of Proposition 1 is in the supplementary material (Barreto 2014). Note that the derived bound corresponds to the maximum difference between the values of two states belonging to an MDP whose rewards are restricted to $[\bar{R}^-_{\min}/|T|, \bar{R}^+_{\max}/|T|]$. This reflects the special structure of the MDPs $\hat{M}$ and $\bar{M}$ imposed by the trees; in the general case the interval above would be replaced by $[\min_{a,i} \bar{r}_i^a, \max_{a,i} \bar{r}_i^a]$, which is a superset of its counterpart.

For the next results we will need an extra assumption:

(iv) The $|T|$ trees in the ensembles have the same structure, that is, T_i^a and T_i^b define the same partitioning of the state space, for any i, a, and b.

From the definition of $\mathbf{D}^a$ in (9) we know that Assumption (iv) implies that $\mathbf{D}^a = \mathbf{D}$ for all $a \in A$. In this case, the transformation computed by TBSF reduces to Sorg and Singh's (2009) concept of *soft homomorphism*, and all results by these authors apply. In particular, we know that if, for any i, $I\{\hat{s}_i \in \bar{T}_j \text{ and } \hat{s}_i \in \bar{T}_l\}$ implies that $\bar{\pi}^*(\bar{x}_j) = \bar{\pi}^*(\bar{x}_l)$, where $\bar{\pi}^*$ is the optimal policy of $\bar{M}$, then the value functions computed by FQIT and TBSF coincide (see Sorg and Singh's Corollary 2, 2009). As a consequence, if a single tree is used in the ensembles the difference $|\tilde{Q}_t(\hat{s}_i, a) - \hat{Q}_t(\hat{s}_i, a)|$ is zero.

Still assuming that (iv) holds, we can show the following result. Let $\bar{\mathbf{r}}^{\text{dif}}$ be the vector whose elements are $\bar{r}_i^{\text{dif}} = \max_a \bar{r}_i^a - \min_a \bar{r}_i^a$. Let $\bar{R}_{\text{dif}}$ be the sum of the $|T|$ largest elements of $\bar{\mathbf{r}}^{\text{dif}}$ and let $\bar{R}^+_{\min}$ be the sum of the $|T|$ smallest elements of $\bar{\mathbf{r}}^+$. Then,

Proposition 2 *For* $t > 0$, $0 \leq \tilde{Q}_t(\hat{s}_i, a) - \hat{Q}_t(\hat{s}_i, a) \leq \frac{1}{|T|} \left[\gamma \bar{R}_{\text{dif}} + \frac{(\gamma^2 - \gamma^t)}{(1-\gamma)} \left(\bar{R}^+_{\max} - \bar{R}^+_{\min}\right)\right]$ *for any* $\hat{s}_i$ *and any* a.

The proof of Proposition 2 is also in the supplementary material (Barreto 2014). Note that increasing the number $|T|$ of trees in the ensembles has two effects on our bounds. If on one hand it decreases the scalar $1/|T|$ multiplying the right-hand side of the bounds, on the other hand it may also increase $\bar{R}^+_{\max}$- $\bar{R}^-_{\min}$, $\bar{R}_{\text{dif}}$, and $\bar{R}^+_{\max}$- $\bar{R}^+_{\min}$, since the number of partitions $\bar{T}_i$ grows. The overall effect of increasing $|T|$ will depend on the structure of the trees.

6 Experiments

We compare the performance of our algorithm with that of FQIT combined with Geurts, Ernst, and Wehenkel's extra-trees algorithm (2006). This combination generated the best results in the extensive empirical evaluation performed by Ernst, Geurts, and Wehenkel (2005). When combined with the extra-trees algorithm FQIT has four parameters; in our experiments we used 200 iterations, $\dim(S)$ candidate cut points to build the trees, and varied $|T|$ and $\eta_{\min}$ (during the construction of the trees, a node is split only if it contains at least $\eta_{\min}$ points—hence, this parameter can be seen as an indirect way of setting the number of partitions).

We also combined TBSF with the extra-trees algorithm, and the same parameters were used to build the trees. Since the extra-trees algorithm does not allow a precise control of the size of the trees, we used trees with the same structure for all actions. As in this case it does not make sense to build the trees based on $\mathcal{Q}_1$, the criterion used to select the cut-points was simply the difference on the cardinality $|\bar{T}_i|$ of the resulting partitions. To serve as a reference, we evaluated a version of FQIT that used a fixed ensemble of trees constructed in the exact same way, referred to as "FQIT-F".

We first use the mountain car task as a proof of concept (Singh and Sutton 1996). In this case, an exploration policy that selects actions uniformly at random was used to collect a batch of n sample transitions. The algorithms's results, shown in Figure 1, correspond to the performance of the greedy policy computed using this data.[2]

As shown in the figure, the algorithms's performance improves with $|T|$ and n, as expected. In general, TBSF's results are intermediate between FQIT's and FQIT-F's. The good performance of FQIT suggests that adapting the structure of the trees at each iteration may play an important role. This ability comes at a price, though: as shown in Figure 1, FQIT's run time is orders of magnitude greater than TBSF's. This opens the possibility of using the latter to process larger amounts of data, as we explore next.

We revisit one of the most successful applications of FQIT: an important medical problem which we will refer to as the HIV domain (Adams et al. 2004; Ernst et al. 2006). Despite the effectiveness of drug cocktails in maintaining low HIV viral loads, there are several complications associated with their long-term use. This has attracted the interest of the scientific community to the problem of optimizing drug-scheduling strategies. The problem can be formulated as a reinforcement-learning task in which the actions correspond to the types of cocktail that should be administered.

[2] See the supplementary material for similar experiments with the puddle-world task (Barreto 2014).

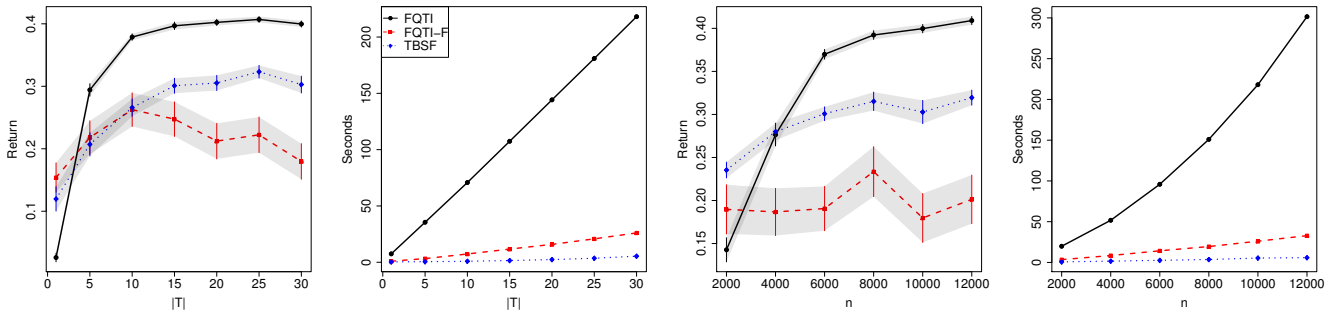

Figure 1: Results on the mountain-car task. All the algorithms used $\eta_{\min} = 30$ to build the trees. In the first two plots $n = 10000$ and in the last two $|T| = 30$. Policies were evaluated on a set of 25 states evenly distributed over $[-1.00, -0.07] \times [0.15, 0.02]$. Shadows represent one standard error over 50 runs.

The problem has a 6-dimensional state space and 4 actions.

Following Ernst et al. (2006), we performed our experiments using a realistic model that describes the interaction of the immune system with HIV (Adams et al. 2004). We also adopted the protocol proposed by these authors to collect data: starting from a batch of 6000 transitions generated by a random policy, corresponding to the treatment of 30 patients, the algorithms computed an initial approximation of the problem's value function. Based on this approximation, a 0.15-greedy policy was used to collect a second batch of 6000 transitions, which was merged with the first. This process was repeated for 10 rounds.

As one can see, this strategy for collecting transitions fits well with the on-line version of TBSF. Therefore, instead of merging the successive batches, we used Algorithm 1 and simply discarded the data after each round. Since in this case the difference between FQIT's and TBSF's computational cost is even bigger, we repeated the experiment with the latter assuming that we had more data available—first from 300 and then from 600 patients. In all other aspects FQIT and TBSF were configured as before.

The results of the experiment are shown in Figure 2. As one can see, when FQIT and TBSF use the same amount of data the former outperforms the latter by a large margin. This is expected, since TBSF was used with a fixed ensemble of trees built based solely on the first batch of transitions. However, when the number of transitions used by TBSF increases, so does the quality of the resulting policies, which eventually start to be competitive with FQIT's. When using $\eta_{\min} = 10$ with the augmented datasets, TBSF is still faster than FQIT and produces policies of about the same quality.

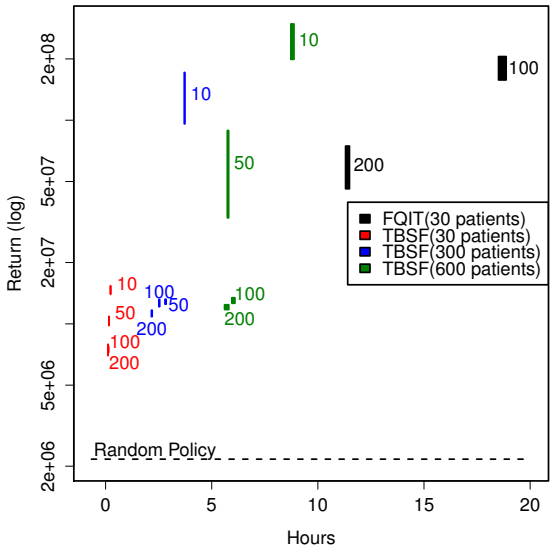

Figure 2: Results on the HIV domain. The algorithms were run with $|T| = 30$; the numbers beside each rectangle is $\eta_{\min}$. Policies were evaluated for 5000 days starting from a state representing an unhealthy condition of the patients (Ernst et al. 2006). The length of the rectangles's edges represents one standard error over 50 runs.

7 Conclusion

In this paper we applied the stochastic-factorization trick to FQIT. The resulting algorithm, TBSF, inherits many desirable properties of its precursor: it is simple to implement, flexible, scalable to high-dimensional spaces, and robust to noise. The main difference between FQIT and TBSF is in the way they construct their models: while the former builds an MDP whose states are the n end-states in the sample transitions, the latter constructs an MDP whose size is the number m of partitions in the ensembles of trees. This has two consequences. First, when $m \ll n$ TBSF is orders of magnitude faster than FQIT. Second, TBSF can be used on-line.

We showed that the distance between the value functions computed by FQIT and TBSF is bounded. Empirically, we verified that FQIT tends to outperform the non-adaptive version of TBSF, especially if the latter is used on-line. However, since TBSF's computational cost is significantly lower than FQIT's, it can process more data in the same amount of time, which can result in better performance. Besides, in problems that are inherently on-line TBSF is a natural candidate to replace FQIT. An interesting direction for future research would be to study the adaptive version of TBSF.

Acknowledgments

The author would like to thank the anonymous reviewers for their comments and suggestions.

References

Adams, B.; Banks, H.; Kwon, H.; and Tran, H. 2004. Dynamic multidrug therapies for HIV: optimal and STI control approaches. *Mathematical Biosciences and Engineering* 1(2):223–41.

Antos, A.; Munos, R.; and Szepesvári, C. 2007. Fitted Q-iteration in continuous action-space MDPs. In *Advances in Neural Information Processing Systems (NIPS)*, 9–16.

Barreto, A. M. S., and Fragoso, M. D. 2011. Computing the stationary distribution of a finite Markov chain through stochastic factorization. *SIAM Journal on Matrix Analysis and Applications* 32:1513–1523.

Barreto, A. M. S.; Precup, D.; and Pineau, J. 2011. Reinforcement learning using kernel-based stochastic factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 720–728.

Barreto, A. M. S.; Precup, D.; and Pineau, J. 2012. Online reinforcement learning using incremental kernel-based stochastic factorization. In *Advances in Neural Information Processing Systems (NIPS)*, 1484–1492.

Barreto, A. M. S. 2014. Tree-based on-line reinforcement learning: Supplementary material. Available on-line.

Ernst, D.; Stan, G.; Gonçalves, J.; and Wehenkel, L. 2006. Clinical data based optimal STI strategies for HIV: a reinforcement learning approach. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, 124–131.

Ernst, D.; Geurts, P.; and Wehenkel, L. 2005. Tree-based batch mode reinforcement learning. *Journal of Machine Learning Research* 6:503–556.

Farahmand, A. M.; Ghavamzadeh, M.; Szepesvári, C.; and Mannor, S. 2009. Regularized fitted Q-iteration for planning in continuous-space Markovian decision problems. In *Proceedings of the American Control Conference*, 725–730.

Geurts, P.; Ernst, D.; and Wehenkel, L. 2006. Extremely randomized trees. *Machine Learning* 36(1):3–42.

Gordon, G. J. 1995. Stable function approximation in dynamic programming. In *Proceedings of the International Conference on Machine Learning (ICML)*, 261–268.

Kalyanakrishnan, S., and Stone, P. 2007. Batch reinforcement learning in a complex domain. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 650–657.

Littman, M. L.; Dean, T. L.; and Kaelbling, L. P. 1995. On the complexity of solving Markov decision problems. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 394–402.

Ormoneit, D., and Sen, S. 2002. Kernel-based reinforcement learning. *Machine Learning* 49 (2–3):161–178.

Puterman, M. L. 1994. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc.

Riedmiller, M. 2005. Neural fitted Q-iteration—first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning (ECML)*, 317–328. Springer.

Singh, S. P., and Sutton, R. S. 1996. Reinforcement learning with replacing eligibility traces. *Machine Learning* 22(1–3):123–158.

Sorg, J., and Singh, S. 2009. Transfer via soft homomorphisms. In *Autonomous Agents & Multiagent Systems/Agent Theories, Architectures, and Languages*, 741–748.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. MIT Press.