# Conservative and Greedy Approaches to Classification-based Policy Iteration

**Mohammad Ghavamzadeh** and **Alessandro Lazaric**
INRIA Lille - Team SequeL, France

**Abstract**

The existing classification-based policy iteration (CBPI) algorithms can be divided into two categories: *direct policy iteration* (DPI) methods that directly assign the output of the classifier (the approximate greedy policy w.r.t. the current policy) to the next policy, and *conservative policy iteration* (CPI) methods in which the new policy is a mixture distribution of the current policy and the output of the classifier. The conservative policy update gives CPI a desirable feature, namely the guarantee that the policies generated by this algorithm improve at each iteration. We provide a detailed algorithmic and theoretical comparison of these two classes of CBPI algorithms. Our results reveal that in order to achieve the same level of accuracy, CPI requires more iterations, and thus, more samples than the DPI algorithm. Furthermore, CPI may converge to suboptimal policies whose performance is not better than DPI's.

## 1 Introduction

*Policy iteration* (Howard 1960) is one of the two main classes of dynamic programming (DP) algorithms to find an optimal policy for a Markov decision process (MDP). It is an iterative algorithm that discovers an optimal policy by generating a sequence of monotonically improving policies. At each iteration $k$ of this algorithm, given the current policy $\pi_k$, an improved policy $\pi_{k+1}$ is generated as the greedy policy w.r.t. the action-value function $Q^{\pi_k}$, i.e., $\pi_{k+1} = \mathcal{G}\pi_k$. In MDPs with large or continuous state and action spaces, policy iteration often fails to improve the policy $\pi_k$ efficiently (to find the exact $\mathcal{G}\pi_k$), mainly because it cannot compute $Q^{\pi_k}$ exactly, and as a result approximation schemes are required. There are two main approaches to approximate policy iteration (API). The most common approach is to find a good approximation of the value function of $\pi_k$ in a real-valued function space, and then compute $\pi_{k+1}$ as the greedy policy w.r.t. this approximation (e.g., Lagoudakis and Parr 2003a). The other variant of API, which is more recent and less studied, replaces the approximation of the action-value function over the entire state-action space with a learning step in a policy space (Kakade 2003; Lagoudakis and Parr 2003b; Fern, Yoon, and Givan 2006; Lazaric, Ghavamzadeh, and Munos 2010a). In this approach, at each iteration we solve a classification problem in order to find a policy in a given policy space that best predicts the greedy action at every state. Thus, this class of API algorithms is called *classification-based policy iteration* (CBPI).

The existing CBPI methods can be divided into two groups according to the way that they generate the next policy $\pi_{k+1}$ from the output of the classifier. The first group directly assigns the output of the classifier to the next policy, therefore are called *direct policy iteration* (DPI) algorithms. The CBPI algorithms by Lagoudakis and Parr (2003b) and Fern et al. (2006) and the algorithm presented and analyzed in Lazaric et al. (2010a) belong to this category. The second group of CBPI algorithms perform a more conservative policy update in which the new policy $\pi_{k+1}$ is a mixture distribution of the current policy $\pi_k$ and the output of the classifier. This group of algorithms are introduced and analyzed in (Kakade and Langford 2002) and (Kakade 2003) under the name *conservative policy iteration* (CPI).[1] By using a conservative policy update, CPI aims at avoiding the main drawback of the DPI methods, i.e., the significant policy degradation resulted from the direct use of approximate greedy policies. This gives CPI two particularly desirable features: **1)** it guarantees to improve the policy at each iteration, i.e., the value function of $\pi_{k+1}$ is larger on average than the value function of $\pi_k$, and **2)** it has a stopping condition based on the quality of the generated policy (it stops whenever it cannot guarantee that the new policy has a better performance than the previous one). These features can potentially make CPI a very appealing API algorithm, mainly because other API methods have no guarantee to generate monotonically improving policies. This includes both value function based API algorithms such as LSPI (Lagoudakis and Parr 2003a) and classification-based API methods other than CPI. Unfortunately, unlike the DPI methods, that have been fully analyzed (Lazaric,



[1] While in Kakade and Langford (2002) the algorithm is presented as a rollout value-based approach, in the more detailed description and analysis of CPI found in Kakade (2003), the algorithm is presented as a CBPI method. In Conclusions we will discuss how the comparison between DPI and CPI reported in this paper can be easily extended to any value-based approach.

Ghavamzadeh, and Munos 2010a) and successfully applied to benchmark problems (Lagoudakis and Parr 2003b; Fern, Yoon, and Givan 2006; Gabillon et al. 2011), CPI has not been empirically evaluated and its performance bounds have not been studied in details and thoroughly compared to other API algorithms.

The main objective of this paper is to better understand the behavior of the CPI algorithm and to find out whether its two main features actually lead to a better performance. To answer this question, we perform a detailed algorithmic and theoretical comparison of these two classes of CBPI algorithms. Our analysis reveals that in order to achieve the same level of accuracy, CPI requires more iterations, and thus, more samples than the DPI algorithm. This indicates that although CPI's conservative update allows it to have a monotonically improving behavior, it slows down the algorithm and increases its sample complexity. Furthermore, CPI may converge to suboptimal policies whose performance is not better than DPI.

## 2 Preliminaries

In this section, we set the notation used throughout the paper. A discounted Markov decision process (MDP) $\mathcal{M}$ is a tuple $\langle \mathcal{X}, \mathcal{A}, r, p, \gamma \rangle$, where the state space $\mathcal{X}$ is a bounded closed subset of a Euclidean space $\Re^d$, the set of actions $\mathcal{A}$ is finite ($|\mathcal{A}| < \infty$), the reward function $r : \mathcal{X} \times \mathcal{A} \to [0, 1]$, the transition model $p(\cdot|x, a)$ is a distribution over $\mathcal{X}$, and $\gamma \in (0, 1)$ is a discount factor. We denote by $\rho$ a distribution over the state space $\mathcal{X}$. We define a policy $\pi : \mathcal{X} \to \mathcal{P}(\mathcal{A})$ as a function mapping each state to a probability distribution over actions. If the policy is deterministic then $\pi$ simply maps each state to an action in $\mathcal{A}$, i.e., $\pi : \mathcal{X} \to \mathcal{A}$. We use $\mathcal{B}^\pi(\mathcal{X})$ to denote the space of deterministic policies.

For a given policy $\pi$, we define its $\gamma$-discounted value function $V^\pi : \mathcal{X} \to [0, 1]$ as the normalized expected sum of discounted rewards obtained by following $\pi$, i.e.,

$$V^\pi(x) = (1-\gamma)\mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(x^t, a^t) | x^0 = x, a^t \sim \pi(x^t)\Big],$$

where the expectation is w.r.t. to the stochastic transitions and the stochastic policy (if $\pi$ is stochastic). Given a state distribution $\rho$, with an abuse of notation, we define the average value function as $V^\pi(\rho) = \mathbb{E}_{x\sim\rho}\big[V^\pi(x)\big]$. The optimal policy $\pi^*$ is the policy which maximizes the value function (i.e., $\pi^* = \arg\max V^\pi(\rho)$) and its corresponding value is the optimal value function $V^* = V^{\pi^*}$. We also define the $\gamma$-discounted action-value function $Q^\pi : \mathcal{X} \times \mathcal{A} \to [0, 1]$ as

$$Q^\pi(x, a) = (1-\gamma)\Big(r(x, a) + \mathbb{E}\Big[\sum_{t=1}^{\infty} \gamma^t r(x^t, a^t) | a^t \sim \pi(x^t)\Big]\Big).$$

If the action $a$ is chosen according to a policy $\pi'$, then we define $Q^\pi(x, \pi') = \mathbb{E}_{a\sim\pi'(x)}\big[Q^\pi(x, a)\big]$, whose average value w.r.t. a state distribution $\rho$ is denoted by $Q^\pi(\rho, \pi') = \mathbb{E}_{x\sim\rho}\big[Q^\pi(x, \pi')\big]$. We then define the greedy policy operator $\mathcal{G} : \mathcal{B}^\pi(\mathcal{X}) \to \mathcal{B}^\pi(\mathcal{X})$ as

$$(\mathcal{G}\pi)(x) = \arg\max_{a\in\mathcal{A}} Q^\pi(x, a), \quad \forall x \in \mathcal{X}, \tag{1}$$

so that $\mathcal{G}\pi$ is the greedy policy w.r.t. $Q^\pi$. We also define the advantage function $A^\pi : \mathcal{X} \times \mathcal{A} \to [-1, 1]$, which indicates how much an action $a$ improves the performance of $\pi$ in state $x$,

$$A^\pi(x, a) = Q^\pi(x, a) - V^\pi(x).$$

We define the advantage of a policy $\pi'$ w.r.t. to $\pi$ as $A^\pi(x, \pi') = \mathbb{E}_{a\sim\pi'(x)}\big[A^\pi(x, a)\big]$, and its average advantage w.r.t. to a state distribution $\rho$ as $A^\pi(\rho, \pi') = \mathbb{E}_{x\sim\rho}\big[A^\pi(x, \pi')\big]$. Finally, we recall the definition of the future-state discounted distribution $d^\pi_\rho$ of policy $\pi$ when the initial state is drawn from a distribution $\rho$ as

$$d^\pi_\rho(x) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mathbb{P}[x^t = x | \rho, \pi].$$

For each of the above quantities, we need to define their estimated counterparts. First, a $H$-horizon rollout of a policy $\pi$ for a state-action pair $(x, a)$ is defined as

$$R^\pi(x, a) = (1-\gamma)\big(r(x, a) + \sum_{t=1}^{H-1} \gamma^t r(x^t, a^t)\big), \tag{2}$$

where $x^1 \sim p(\cdot|x, a)$, $x^t \sim p\big(\cdot\,|x^{t-1}, a^{t-1}\big)$, and $a^{t-1} \sim \pi(x^{t-1})$. If $M$ independent rollouts are available at a state-action pair $(x, a)$, then we can estimate $Q^\pi(x, a)$ as

$$\widehat{Q}^\pi(x, a) = \frac{1}{M}\sum_{j=1}^{M} R^\pi_j(x, a). \tag{3}$$

Let $\mathcal{D} = \{x_i\}_{i=1}^N$ be a set of $N$ states drawn independently from a state distribution $\mu$, then the estimated average action-value function for a policy $\pi'$ is defined as

$$\widehat{Q}^\pi(\widehat{\mu}, \pi') = \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{a\sim\pi'(x_i)}\big[\widehat{Q}^\pi(x_i, a)\big].$$

Similarly, we may define the estimated advantages $\widehat{A}^\pi(x, a)$, $\widehat{A}^\pi(x, \pi')$, and $\widehat{A}^\pi(\widehat{\rho}, \pi')$.

## 3 CPI and DPI Algorithms

In this section, we provide a detailed algorithmic comparison of *direct policy iteration* (DPI) (Lazaric, Ghavamzadeh, and Munos 2010a) and *conservative policy iteration* (CPI) (Kakade and Langford 2002; Kakade 2003) methods.

### 3.1 The General Structure

Before getting into the details of each of the two algorithms, we first describe their common structure (Figure 1). A classification-based policy iteration (CBPI) algorithm takes as input a policy space $\Pi$, a state distribution $\rho$, and an accuracy parameter $\epsilon$ that defines the stopping condition of the algorithm. Given the desired accuracy $\epsilon$, the algorithm first sets the number of rollout states $N$, the number of rollouts per state-action pair $M$, and the rollout horizon $H$ (Step **(a)**). These parameters define the total number of transitions used at each iteration of the algorithm (the per-iteration budget $B$). As it will be discussed in the next section, the choice of

**Input:** policy space $\Pi$, state distribution $\rho$, accuracy $\epsilon$
**Initialize:** Select an arbitrary policy $\pi_0 \in \Pi$ and set $k = 0$
**(a)** Compute the per-iteration budget $B(\epsilon)$ and $N$, $M$, $H$
**repeat**
  **(b)** Define a suitable sampling policy $\mu_k$ from $\rho$ and $\pi_k$
  Build the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N,\ x_i \sim \mu_k$
  **for all** states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**
    **(c)** Compute a rollout estimate $\widehat{Q}^{\pi_k}(x_i, a)$ (see Eq. 3)
  **end for**
  **(d)** Compute the approximate greedy policy $\pi'$ w.r.t. $\pi_k$
  **(e)** Construct the next policy $\pi_{k+1}$ from $\pi'$ and $\pi_k$
  $k = k + 1$
**until (f)** Termination($\epsilon$) = `false`

Figure 1: The general structure of a classification-based policy iteration (CBPI) algorithm.

**Input:** policy space $\Pi$, state distribution $\rho$, accuracy $\epsilon$
**Initialize:** Select an arbitrary policy $\pi_0 \in \Pi$ and set $k = 0$
**(a)** Compute the per-iteration budget $B(\epsilon)$ and $N$, $M$, $H$
**(a')** Compute the total number of iterations $K(\epsilon)$
**repeat**
  **(b)** Set $\mu_k = \rho$
  Build the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N,\ x_i \sim \mu_k$
  **for all** states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**
    **(c)** Compute an estimate $\widehat{Q}^{\pi_k}(x_i, a)$ with $M$ rollouts
  **end for**
  **(d)** Compute the approximate greedy policy $\pi'$ w.r.t. $\pi_k$
  **(e)** $\pi_{k+1} = \pi'$
  $k = k + 1$
**until (f)** $k \leq K(\epsilon)$

Figure 2: The direct policy iteration (DPI) algorithm.

these parameters depends on the performance bound of the algorithm at each iteration. Starting with an arbitrary policy $\pi_0 \in \Pi$, the algorithm can be summarized in the following main steps. At Step **(b)**, the algorithm builds a rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N$, with states $x_i$ drawn from a sampling distribution $\mu_k$ defined according to the input distribution $\rho$ and the current policy $\pi_k$. For each of the rollout states $x \in \mathcal{D}_k$ and for each action $a \in \mathcal{A}$, a rollout estimate of the action-value function of the current policy $\widehat{Q}^{\pi_k}(x, a)$ is computed at Step **(c)** (see Eqs. 2 and 3). Step **(d)** receives the estimated action-values $\widehat{Q}^{\pi_k}(x, a), \forall x \in \mathcal{D}_k, \forall a \in \mathcal{A}$, and computes an approximation of $\mathcal{G}\pi_k$. More precisely, the approximated greedy policy $\pi'$ is computed as

$$\pi' = \arg\max_{\pi \in \Pi} \widehat{Q}^{\pi_k}(\widehat{\mu}_k, \pi). \tag{4}$$

Note that this problem can be cast as a cost-sensitive classification problem, in which the training set is of the form $\left\{(x_i, a), \widehat{Q}^{\pi_k}(x_i, a)\right\}_{i=1}^N, \forall a \in \mathcal{A}$. Finally, at Steps **(e)** and **(f)**, the algorithm first constructs the next policy $\pi_{k+1}$ using $\pi_k$ and $\pi'$, and then checks a termination condition.

### 3.2 DPI Algorithm

Figure 2 shows how the direct policy iteration (DPI) algorithm implements each of the steps of the general CBPI structure. Although we focus on the specific DPI algorithm analyzed in Lazaric et al. (2010a), the pseudo-code of Figure 2 also includes other DPI algorithms in the literature that use a $0/1$ loss function in the computation of the approximate greedy policy (Step **(d)**) instead of solving a cost-sensitive classification problem (e.g., Lagoudakis and Parr 2003b). After computing the per-iteration budget (i.e, $N$, $M$, and $H$), DPI also determines the total number of iterations $K(\epsilon)$ needed to meet the required accuracy $\epsilon$ (Step **(a')**). This reduces the termination condition at Step **(f)** to a simple condition on the number of iterations ran so far. Note that at each iteration the algorithm builds the rollout set with states drawn from $\rho$ (Step **b**), and runs $M$ rollouts for each state-action pair (Step **(c)**). Finally, the policy at the next iteration $\pi_{k+1}$ is set to $\pi'$ (Step **(e)**).

**Input:** policy space $\Pi$, state distribution $\rho$, accuracy $\epsilon$
**Initialize:** Select an arbitrary policy $\pi_0 \in \Pi$ and set $k = 0$
**(a)** Compute the per-iteration budget $B(\epsilon)$ and $N$, $M$, $H$
**repeat**
  **(b)** Set $\mu_k = d_\rho^{\pi_k}$
  Build the rollout set $\mathcal{D}_k = \{x_i\}_{i=1}^N,\ x_i \sim \mu_k$
  **for all** states $x_i \in \mathcal{D}_k$ and actions $a \in \mathcal{A}$ **do**
    **(c)** Compute an estimate $\widehat{Q}^{\pi_k}(x_i, a)$ with one rollout
  **end for**
  **(d)** Compute the approximate greedy policy $\pi'$ w.r.t. $\pi_k$
  **(e)** Compute $\alpha = \frac{\widehat{A}(\widehat{\mu}_k, \pi') - \epsilon/3}{4}(1 - \gamma)$
  **(e')** $\pi_{k+1} = (1 - \alpha)\pi_k + \alpha\pi'$
  $k = k + 1$
**until (f)** $\widehat{A}(\widehat{\mu}_k, \pi') \leq \frac{2\epsilon}{3}$

Figure 3: The conservative policy iteration (CPI) algorithm.

### 3.3 CPI Algorithm

Figure 3 shows how the conservative policy iteration (CPI) algorithm (Kakade and Langford 2002; Kakade 2003) implements each of the steps of the general CBPI structure. In CPI, the sampling distribution $\mu_k$ is set to the future-state discounted distribution $d_\rho^{\pi_k}$ of the current policy $\pi_k$ (Step **(b)**). In practice, in order to generate samples $x_i$ from the distribution $d_\rho^{\pi_k}$, we first draw a state $x^0$ from $\rho$ and then we follow policy $\pi_k$. At each step $t$, the current state $x^t$ is either accepted as $x_i$ with probability $\gamma$ and the simulation stops or it is rejected with probability $1 - \gamma$ and the rollout continues. If the rollout is not terminated after $H$ steps (the rollout horizon), then it is forced to terminate and the last state $x^H$ is selected as $x_i$. Once the rollout set is built, one single rollout is run for each state-action pair (Step **(c)**). While the approximate greedy policy $\pi'$ is computed as in Eq. 4 (Step **(d)**), CPI constructs the next policy $\pi_{k+1}$ in a conservative way as a mixture distribution of the current policy $\pi_k$ and the approximate greedy policy $\pi'$ with a suitable mixing parameter $\alpha$. The algorithm stops when the estimated advantage of $\pi'$ is smaller than $2\epsilon/3$ (Step **(f)**).

### 3.4 The Algorithmic Comparison

Given the sketch of the DPI and CPI algorithms in Figures 2 and 3, we can now easily highlight their most relevant algorithmic differences. The first difference is in the construction

of the rollout set $\mathcal{D}_k$, where DPI and CPI use the sampling distributions $\rho$ and $d_\rho^{\pi_k}$, respectively. This implies that the greedy policies approximated by the two algorithms are different. In fact, if we let the size of the rollout set, the number of rollouts, and the rollout horizon go to infinity, we have

$$\pi'_{\text{CPI}} = \arg\max_{\pi\in\Pi} Q^{\pi_k}(d_\rho^{\pi_k}, \pi), \quad \pi'_{\text{DPI}} = \arg\max_{\pi\in\Pi} Q^{\pi_k}(\rho, \pi).$$

If $\Pi$ contains the actual greedy policy $\mathcal{G}(\pi_k)$, then $\pi'_{\text{CPI}} = \pi'_{\text{DPI}}$, otherwise CPI favors policies that improve $\pi_k$ in the regions of the state space reachable by $\pi_k$ itself, while DPI prefers policies with high action-values in the states covered by the state distribution $\rho$. The choice of $d_\rho^{\pi_k}$ is critical since it allows CPI to estimate the advantage $A^{\pi_k}(d_\rho^{\pi_k}, \pi'_{\text{CPI}})$ which is used to compute a suitable $\alpha$ that guarantees $\pi_{k+1}$ improves over $\pi_k$ (Step **(e)**). On the other hand, since DPI does not ensure any monotonic improvement, it does not require any specific sampling strategy. Once the rollout set is built, Steps **(c)** and **(d)** are the same for both algorithms. The other major difference is in the way the next policy $\pi_{k+1}$ is generated in Step **(e)**. While DPI simply sets the next policy $\pi_{k+1}$ to $\pi'$, CPI returns a policy $\pi_{k+1}$ which is a mixture of the current policy $\pi_k$ and the approximate greedy policy $\pi'$. In fact, CPI is based on the evidence that if $\pi'$ has a sufficiently large average advantage $A^{\pi_k}(\mu_k, \pi')$, then it is possible to compute a mixing parameter $\alpha$ such that the corresponding mixture $\pi_{k+1}$ is guaranteed to improve the current policy (see Corollary 7.2.3 in Kakade 2003 for more details), i.e., the value function of $\pi_{k+1}$ is larger than the value function of $\pi_k$ ($V^{\pi_{k+1}}(\rho) \geq V^{\pi_k}(\rho)$). Although this is a major advantage over DPI, it comes at the cost of using stochastic policies instead of deterministic ones, which may be problematic in some applications (e.g., in robotics). The difference in the computation of $\pi_{k+1}$ also explains the different termination conditions (Step **(f)**). In fact, while DPI simply limits the number of iterations, CPI stops whenever it cannot guarantee a sufficient improvement over the current policy (which depends on the advantage $A^{\pi_k}(\widehat{\mu}_k, \pi')$). As discussed in the next section, this difference has a deep impact on the actual performance of the algorithms. In fact, the monotonic improvement of CPI may result in an excessively slow convergence to a solution which may not even be better than the solution found by DPI in fewer iterations.

# 4 CPI and DPI Performance Bounds

While in the last section we compared CPI and DPI from an algorithmic perspective, we now study how these differences affect the performance of these algorithms. In particular, we elaborate on their theoretical analysis in (Kakade and Langford 2002) (for CPI) and (Lazaric, Ghavamzadeh, and Munos 2010a) (for DPI), and reformulate their performance bounds in order to capture their similarities and differences and better explain their behaviors.

Thanks to the comparison in the previous section, we notice that, while the two algorithms are very similar in approximating the greedy policy $\pi'$ at each iteration, they mainly differ in the way they compute the next policy $\pi_{k+1}$. Thus, we split the theoretical comparison into two parts, the part that takes into account the error in approximating the greedy policy $\pi'$, and the one which studies how this error is propagated through the iterations of the algorithms.[2]

## 4.1 Error in Approximating the Greedy Policy

As described in Figure 1 Steps **(b-d)**, at each iteration $k$, the two algorithms (a CBPI algorithm in general) build a rollout set $\mathcal{D}_k$ according to $\mu_k$, compute a rollout estimate $\widehat{Q}^{\pi_k}(x_i, a), \forall x_i \in \mathcal{D}_k, \forall a \in \mathcal{A}$, and finally generate a training set for a classifier that solves a cost-sensitive classification problem and returns an estimated greedy policy $\pi'$ (w.r.t. $\pi_k$). The quality of such a policy depends on two main factors (see also Eqs. 1 and 4): the number of transitions used to compute $\widehat{Q}^{\pi_k}(x_i, a)$, i.e., the per-iteration budget $B(\epsilon)$ at Step **(a)**, and the policy space $\Pi$ (the classifier) to which $\pi'$ belongs to. Although the results in (Kakade and Langford 2002) and (Lazaric, Ghavamzadeh, and Munos 2010a) are stated differently and analyze slightly different quantities, it is easy to derive a general theorem bounding the action-value of $\pi'$ w.r.t. policy $\pi_k$ and a distribution $\mu_k$. A proof of the following theorem is reported in Appendix.

**Theorem 1** *If the policy space $\Pi$ received by the algorithm of Figure 1 has a finite VC-dimension $VC(\Pi) = h < \infty$, and the per-iteration budget is $B(\epsilon) = 2NMH$ with*

$$N = \Omega\left(\frac{h}{\epsilon^2}\log\frac{1}{\epsilon\delta}\right), \quad M = \Omega(1), \quad H = \Omega\left(\frac{\log 1/\epsilon}{1-\gamma}\right),$$

*then at each iteration $k$, the algorithm returns an approximate greedy policy $\pi'$ such that*

$$Q^{\pi_k}(\mu_k, \mathcal{G}\pi_k) - Q^{\pi_k}(\mu_k, \pi') \leq \inf_{\pi\in\Pi}\left[Q^{\pi_k}(\mu_k, \mathcal{G}\pi_k) - Q^{\pi_k}(\mu_k, \pi)\right] + \epsilon, \tag{5}$$

*with probability at least $1-\delta$ (w.r.t. to random rollouts).*

This bound actually decomposes the loss in the performance of $\pi'$ w.r.t. $\mathcal{G}\pi_k$ into a term which depends on the "richness" of the policy space $\Pi$ and is usually referred to as *approximation error*, and the term $\epsilon$ which depends on the per-iteration budget (i.e., the number of transitions needed at each iteration) and is called the *estimation error*. As we will see in the next section, this decomposition is crucial to understand how this error is propagated by the two algorithms. We also notice that the number of rollouts per state-action pair $M$ can be actually set to 1 for both algorithms.

## 4.2 Error Propagation

In Step **(e)**, CPI and DPI use the approximate greedy policy $\pi'$ in a completely different way. While DPI simply uses it as the next policy $\pi_{k+1}$ and then iterates for $K(\epsilon)$ iterations, CPI generates a more conservative policy $\pi_{k+1}$ by combining the current policy $\pi_k$ and $\pi'$ which guarantees a monotonic improvement over $\pi_k$ in terms their average value functions. This different behavior naturally leads to a difference in the final performance. In fact, DPI usually does

[2] As in (Kakade and Langford 2002) and (Lazaric, Ghavamzadeh, and Munos 2010a), from this point on, we assume that the action space contains only two actions $|\mathcal{A}| = 2$.

not converge to a fix policy and keeps oscillating between different policies (i.e., the so-called *convergence to region*). On the other hand, CPI monotonically improves the policies until it converges to a policy that cannot be improved any further. Although this nice property may suggest that CPI outperforms DPI, by a careful inspection of their theoretical properties we obtain some surprising results. Before stating the main theorems that contain the performance bounds of the algorithms, we need to introduce a few terms and assumptions that play a critical role in the comparison. Since we are interested in bounding the final performance of the algorithms in $\sigma$-norm, which might be different than the sampling distribution $\rho$, we use the following assumptions:

**Assumption 1** *For any policy $\pi$ and any non-negative integers $s$ and $t$, there exists a constant $C_{\sigma,\rho}(s,t) < \infty$ such that $\sigma(P^*)^s(P^\pi)^t \leq C_{\sigma,\rho}(s,t)\rho$. We define $C_{DPI}(\sigma,\rho) = (1-\gamma)^2 \sum_{s=0}^{\infty}\sum_{t=0}^{\infty}\gamma^{s+t}C_{\mu,\rho}(s,t)$.*

**Assumption 2** *The term $C_{CPI}(\sigma,\rho) = ||d_\sigma^{\pi^*}/\rho||_\infty$, which represents the mismatch between the $\gamma$-discounted future state distribution of the optimal policy $\pi^*$ for the starting state distribution $\sigma$ and the distribution $\rho$, is bounded.*

Similar to (Lazaric, Ghavamzadeh, and Munos 2010a), we refer to $C_{\text{DPI}}$ and $C_{\text{CPI}}$ as *concentrability* terms. We also define the approximation error of the algorithms as:

**Definition 1** *The inherent greedy error of a policy space $\Pi$ is defined for DPI and CPI as follows:*

$$d_{DPI}(\Pi,\mathcal{G}\Pi) = \sup_{\pi\in\Pi}\inf_{\pi'\in\Pi}\big[Q^\pi(\rho,\mathcal{G}\pi) - Q^\pi(\rho,\pi')\big],$$

$$d_{CPI}(\Pi,\mathcal{G}\Pi) = \sup_{\pi\in\Pi}\inf_{\pi'\in\Pi}\big[Q^\pi(d_\rho^\pi,\mathcal{G}\pi) - Q^\pi(d_\rho^\pi,\pi')\big].$$

We now state the main results, i.e., the performance bounds of the algorithms.

**Theorem 2** *If the policy space $\Pi$ received by CPI has a finite VC-dimension $VC(\Pi) = h < \infty$, and the per-iteration budget is $B(\epsilon)$ as in Theorem 1, then CPI stops after at most*

$$K(\epsilon) = O\left(\frac{1}{\epsilon^2}\right)$$

*iterations and returns a policy $\pi_{CPI}$ such that*

$$V^*(\sigma) - V^{\pi_{CPI}}(\sigma) \leq \frac{C_{CPI}(\sigma,\rho)}{(1-\gamma)^2}\big(d_{CPI}(\Pi,\mathcal{G}\Pi) + \epsilon\big),$$

*with probability $1 - K\delta$.*

**Theorem 3** *If the policy space $\Pi$ received by DPI has a finite VC-dimension $VC(\Pi) = h < \infty$, and the per-iteration budget is $B(\epsilon)$ as in Theorem 1, then DPI runs for*

$$K(\epsilon) = O\left(\frac{\log 1/\epsilon}{(1-\gamma)}\right)$$

*iterations and returns a policy $\pi_{DPI}$ such that*

$$V^*(\sigma) - V^{\pi_{DPI}}(\sigma) \leq \frac{C_{DPI}(\sigma,\rho)}{(1-\gamma)^2}\big(d_{DPI}(\Pi,\mathcal{G}\Pi) + \epsilon\big),$$

*with probability $1 - K\delta$.*

While the proof of Theorem 2 is reported in the Appendix, Theorem 3 is a rewriting of Theorem 6 in Lazaric et al. (2010b).

*Remark 1 (approximation error).* The first interesting result is the definition of approximation error $d_{\text{CPI}}(\Pi,\mathcal{G}\Pi)$ in CPI. In order to understand it better, let us consider the asymptotic case where $\epsilon = 0$ (using infinite number of rollouts and iterations). Since CPI improves the quality of the policy at each iteration, one might expect it to return the policy $\tilde{\pi} \in \Pi$ that best approximates the performance of $\pi^*$, i.e., $\tilde{\pi} = \arg\max_{\pi\in\Pi} V^\pi(d_\rho^{\pi^*})$. However, unlike exact policy iteration where monotonic improvement is enough to guarantee convergence to the optimal policy, CPI may converge to a suboptimal policy. The approximation error $d_{\text{CPI}}(\Pi,\mathcal{G}\Pi)$ exactly bounds the error of the worst suboptimal policy that CPI can converge to (see also Appendix to find out how the approximation error appears in the final bound of CPI). Another interesting aspect of the result is the striking similarity between the approximation error terms of the algorithms. Apart from the use of two different norms $\rho$ and $d_\rho^\pi$, CPI may converge to a policy with the same quality as the one returned by DPI. This is surprising since at each iteration DPI does not have any guarantee to improve over the current policy and it may oscillate between multiple policies. Nonetheless, the performance of such policies is not worse than the suboptimal policy CPI can converge to. Finally, we note that

$$d_{\text{CPI}}(\Pi,\mathcal{G}\Pi) \geq (1-\gamma)d_{\text{DPI}}(\Pi,\mathcal{G}\Pi).$$

Although this inequality does not imply any strict ordering between the two error terms, it suggests that for $\gamma$ close to 1, CPI may have a worse approximation error than DPI.

*Remark 2 (number of iterations).* One of the major differences in the bounds is the number of iterations $K(\epsilon)$ that CPI needs to converge and DPI needs to run in order to guarantee an $\epsilon$ accuracy (estimation error). This indicates that CPI requires exponentially more iterations than DPI to obtain the same accuracy. This is because it converges after $1/\epsilon^2$ iterations, while DPI only needs $\log(1/\epsilon)/(1-\gamma)$ steps. This result is not completely surprising since CPI performs conservative updates and modifies the current policy with a step-size $\alpha$, which can be very small, while DPI always performs a full update and set the new policy to the (approximate) greedy policy. Based on the number of iterations and the approximation error, we may conclude that CPI is over-conservative, and although DPI does not necessarily improve its performance monotonically, it could converge to policies which are as good as CPI with much less iterations, and thus, less number of samples. However, we should note that $K(\epsilon)$ in DPI has a factor $1/(1-\gamma)$

which can be large when $\gamma$ is close to 1, thus forcing DPI to run for many iterations before having $\epsilon$ accuracy. Moreover, we remark that the $K(\epsilon)$ in Theorem 2 may largely over-estimate the actual number of iterations the algorithm needs before stopping. In fact, the upper-bound is tight only when at each iteration the advantage of $\pi'$ is very close to $\epsilon$ (see Step **(f)** in Figure 3), the starting policy has an average value of 0, and the returned policy has an average value of 1, conditions which are not likely to happen in general.

*Remark 3 (concentrability terms).* Another difference in the bounds is in the concentrability terms $C_{\text{DPI}}(\sigma, \rho)$ and $C_{\text{CPI}}(\sigma, \rho)$ defined in Assumptions 1 and 2. These terms indicate that selecting the input distribution $\rho$ (in relation to the evaluation distribution $\sigma$ and the MDP) may have a significant effect on the final performance of the algorithms. Unfortunately, selecting the right $\rho$ requires complete knowledge of the MDP and some related quantities that are not usually available in advance, such as the optimal policy $\pi^*$ itself. This is why we usually set $\rho$ to a safe distribution such as uniform in order to make sure that the concentrability terms remain bounded. One potential advantage of CPI over DPI is that if some knowledge about the stationary distribution $d_\sigma^{\pi^*}$ of the optimal policy $\pi^*$ is available, it would be possible to select $\rho$ so as to minimize $C_{\text{CPI}}(\sigma, \rho)$, while knowing $\pi^*$ is not enough to optimize the concentrability terms $C_{\text{DPI}}(\sigma, \rho)$ in the DPI algorithm.

*Remark 4 (convergence of DPI).* In case the DPI algorithm converges to a policy $\tilde{\pi}$, using the results of Sec. 4.2 of Lazaric et al. (2010a), we can show that

$$V^* - V^{\tilde{\pi}} \leq (I - \gamma P^*)^{-1} \big( Q^{\tilde{\pi}}(\mathcal{G}\tilde{\pi}) - Q^{\tilde{\pi}}(\tilde{\pi}) \big).$$

This point-wise inequality implies the following final performance bound for the DPI algorithm:

$$V^*(\sigma) - V^{\tilde{\pi}}(\sigma) \leq \frac{C_{\text{CPI}}(\sigma, \rho)}{1 - \gamma} \big( d_{\text{DPI}}(\Pi, \mathcal{G}\Pi) + \epsilon \big). \quad (6)$$

Comparing Eq. 6 with the bound of Theorem 3, we note two major improvements in the performance bound of DPI: **1)** the bound has been improved by a factor $1/(1-\gamma)$, which is significant especially when $\gamma$ is close to 1, and **2)** the CPI concentrability term, which can be better optimized than its DPI counterpart (see Remark 3), appears in the DPI bound. This indicates that in case DPI converges, it achieves a performance superior to the one by CPI with a factor $1/(1-\gamma)$.

## 5 Conclusions

In this paper, we provided a detailed algorithmic and theoretical comparison of the two classes of classification-based policy iteration algorithms: *direct policy iteration* (DPI) and *conservative policy iteration* (CPI). The main objective was to fully understand the potential advantages of the CPI algorithm, particularly the fact that it guarantees to generate policies that improve at each iteration before the algorithm terminates. This desirable feature suggests that CPI could outperform the standard approximate policy iteration methods whose full greedy updates do not necessarily lead to monotonically improving policies. In this paper, we first identified similarities and differences between CPI and DPI, and highlighted the parts of the algorithms that have major impact in their performance. The theoretical analysis then led to two interesting findings: **1)** in order to achieve the same level of accuracy, CPI requires more iterations, and thus, more samples than DPI (see Remark 2), and **2)** CPI may converge to suboptimal policies whose performance is no better than the one achieved by DPI (see Remark 1). This indicates that CPI's conservative update comes at the cost of increasing the sample complexity of the algorithm. Nonetheless, as reported in Remarks 2 and 3, CPI may still be a valid choice when $\gamma$ is close to 1 and some knowledge of the stationary distribution of the optimal policy is available.

Finally, we remark that the results reported in the paper in terms of the comparison between the conservative and direct policy updates are not necessarily limited to the classification-based approach to policy iteration. In fact, the results and analysis can be easily extended to any value function based policy iteration methods (e.g., LSPI) by simply replacing Theorem 1 with a suitable bound on the policy evaluation error (Theorems 2 and 3 remain unchanged).

## Acknowledgments

This work was supported by French National Research Agency (ANR) through the project LAMPADA $n^\circ$ ANR-09-EMER-007, by Ministry of Higher Education and Research, Nord-Pas de Calais Regional Council and FEDER through the "contrat de projets état region (CPER) 2007–2013", European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement $n^\circ$ 231495, and by PASCAL2 European Network of Excellence.

## Appendix

In this section, we sketch the proof of Theorems 1 and 2.

*Proof of Theorem 1:* For DPI, the bound of Eq. 5 is obtained by noting that maximizing the action value (Eq. 4) is equivalent to minimize the loss defined in (Lazaric, Ghavamzadeh, and Munos 2010a) and by inverting the bound in Theorem 1 of (Lazaric, Ghavamzadeh, and Munos 2010a) (or more accurately in Theorem 5 of (Lazaric, Ghavamzadeh, and Munos 2010b)).

For CPI, we first notice that the maximization of the action-value function corresponds to maximize the advantage and the bound of Eq. 5 is an immediate consequence of Lemma 7.3.4 in (Kakade 2003). We can write the following sequence of inequalities

$$\begin{aligned} Q^{\pi_k}(\mu_k, \tilde{\pi}) - Q^{\pi_k}(\mu_k, \pi') &= Q^{\pi_k}(\mu_k, \tilde{\pi}) - \widehat{Q}^{\pi_k}(\widehat{\mu}_k, \tilde{\pi}) \\ &\quad + \widehat{Q}^{\pi_k}(\widehat{\mu}_k, \tilde{\pi}) - \widehat{Q}^{\pi_k}(\widehat{\mu}_k, \pi') \\ &\quad + \widehat{Q}^{\pi_k}(\widehat{\mu}_k, \pi') - Q^{\pi_k}(\mu_k, \pi') \leq \frac{\epsilon}{2} + 0 + \frac{\epsilon}{2}, \end{aligned}$$

where $\tilde{\pi} = \arg\max_{\pi \in \Pi} Q^{\pi_k}(\mu_k, \pi)$, and the first and the last two terms are bounded by Lemma 7.3.4 in (Kakade

2003), if $N$ and $H$ are chosen as in Theorem 1, and the second two terms are smaller than zero by the definition of $\pi'$ as the best policy in $\Pi$ w.r.t. the estimated action-value function $\widehat{Q}^{\pi_k}(\widehat{\mu}_k, \cdot)$. The statement follows by adding and subtracting $Q^{\pi_k}(\mu_k, \mathcal{G}\pi_k)$ and reordering the terms.

We now turn to the proof of Theorem 2. In the original analysis of CPI reported in (Kakade and Langford 2002) no approximation error appears explicitly in the final bound. The reason is that by Definition 4.3 in (Kakade and Langford 2002), it is assumed that the greedy policy $\pi'$ returned at Step **(d)** is always $\epsilon$ close to the actual greedy policy $\mathcal{G}\pi_k$. In general, it is not possible to guarantee such a performance, since the quality of the policy space $\Pi$ in approximating the greedy policy is directly related to the classifier. In fact, it is clear from Theorem 1 that even if an infinite number of rollouts is used to compute $\pi'$, its performance is still constrained by the quality of the policy space $\Pi$. In the following, we report the steps needed to introduce the approximation error in the analysis of (Kakade and Langford 2002).

*Proof of Theorem 2:* Similar to (Kakade and Langford 2002), we define $\text{OPT}(\mu, \pi) = \max_{\pi'} A^{\pi}(\mu, \pi')$ as the maximum amount a policy $\pi$ can be improved w.r.t. a distribution $\mu$. Note that $\pi'$ is any stationary policy defined over the MDP at hand and not the best policy in the policy space $\Pi$. We may rewrite $\text{OPT}(\mu, \pi)$ as

$$\begin{aligned}
\text{OPT}(\mu, \pi) &= \max_{\pi'} \left[ Q^{\pi}(\mu, \pi') - V^{\pi}(\mu) \right] \\
&= Q^{\pi}(\mu, \mathcal{G}\pi) - V^{\pi}(\mu) \\
&= Q^{\pi}(\mu, \mathcal{G}\pi) - Q^{\pi}(\mu, \tilde{\pi}) + Q^{\pi}(\mu, \tilde{\pi}) - V^{\pi}(\mu) \\
&= \inf_{\pi' \in \Pi} \left[ Q^{\pi}(\mu, \mathcal{G}\pi) - Q^{\pi}(\mu, \pi') \right] + A^{\pi}(\mu, \tilde{\pi}),
\end{aligned}$$

where $\tilde{\pi} = \arg\max_{\pi' \in \Pi} Q^{\pi}(\mu, \pi')$. From Theorem 7.3.3 in (Kakade 2003), if the per-iteration budget is the same as in Theorem 1 and CPI is run for $O(1/\epsilon^2)$ iterations, then it returns a policy $\pi_{\text{CPI}}$ such that for any policy $\bar{\pi} \in \Pi$, $A(d_{\rho}^{\pi_{\text{CPI}}}, \bar{\pi}) \leq \epsilon$. This means that the returned policy cannot be improved more than $\epsilon$ w.r.t. $d_{\rho}^{\pi_{\text{CPI}}}$ by the policies in the policy space $\Pi$. As a result, for any $\bar{\pi} \in \Pi$, and thus, also for $\tilde{\pi}$, and $\mu = d_{\rho}^{\pi_{\text{CPI}}}$, we have

$$\text{OPT}(\mu, \pi_{\text{CPI}}) \leq \inf_{\pi' \in \Pi} \left[ Q^{\pi_{\text{CPI}}}(\mu, \mathcal{G}\pi_{\text{CPI}}) - Q^{\pi_{\text{CPI}}}(\mu, \pi') \right] + \epsilon.$$

Since $\pi_{\text{CPI}}$ is a random variable, the first term above is also a random variable, and thus, needs to be further upper bounded as

$$\begin{aligned}
\text{OPT}(\mu, \pi_{\text{CPI}}) &\leq \sup_{\pi \in \Pi} \inf_{\pi' \in \Pi} \left[ Q^{\pi}(\mu, \mathcal{G}\pi) - Q^{\pi}(\mu, \pi') \right] + \epsilon \\
&= d_{\text{CPI}}(\Pi, \mathcal{G}\Pi) + \epsilon.
\end{aligned}$$

We obtain the statement of Theorem 2 by plugging this result into Theorem 6.2 in (Kakade and Langford 2002).

## References


Fern, A.; Yoon, S.; and Givan, R. 2006. Approximate policy iteration with a policy language bias: Solving relational Markov decision processes. *Journal of Artificial Intelligence Research* 25:85–118.

Gabillon, V.; Lazaric, A.; Ghavamzadeh, M.; and Scherrer, B. 2011. Classification-based policy iteration with a critic. In *Proceedings of the Twenty-Eighth International Conference on Machine Learning*, 1049–1056.

Howard, R. 1960. *Dynamic Programming and Markov Processes*. MIT Press.

Kakade, S., and Langford, J. 2002. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 267–274.

Kakade, S. 2003. *On the Sample Complexity of Reinforcement Learning*. Ph.D. Dissertation, Gatsby Computational Neuroscience Unit., University College London.

Lagoudakis, M., and Parr, R. 2003a. Least-squares policy iteration. *Journal of Machine Learning Research* 4:1107–1149.

Lagoudakis, M., and Parr, R. 2003b. Reinforcement learning as classification: Leveraging modern classifiers. In *Proceedings of the Twentieth International Conference on Machine Learning*, 424–431.

Lazaric, A.; Ghavamzadeh, M.; and Munos, R. 2010a. Analysis of a classification-based policy iteration algorithm. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning*, 607–614.

Lazaric, A.; Ghavamzadeh, M.; and Munos, R. 2010b. Analysis of a classification-based policy iteration algorithm. Technical Report 00482065, INRIA.