Is the Bellman residual a bad proxy?

Authored by:

Matthieu Geist
Bilal Piot
Olivier Pietquin

Abstract

This paper aims at theoretically and empirically comparing two standard optimization criteria for Reinforcement Learning: i) maximization of the mean value and ii) minimization of the Bellman residual. For that purpose, we place ourselves in the framework of policy search algorithms, that are usually designed to maximize the mean value, and derive a method that minimizes the residual $—T_* v_pi - v_pi—_{1,nu}$ over policies. A theoretical analysis shows how good this proxy is to policy optimization, and notably that it is better than its value-based counterpart. We also propose experiments on randomly generated generic Markov decision processes, specifically designed for studying the influence of the involved concentrability coefficient. They show that the Bellman residual is generally a bad proxy to policy optimization and that directly maximizing the mean value is much better, despite the current lack of deep theoretical analysis. This might seem obvious, as directly addressing the problem of interest is usually better, but given the prevalence of (projected) Bellman residual minimization in value-based reinforcement learning, we believe that this question is worth to be considered.

1 Paper Body

Reinforcement Learning (RL) aims at estimating a policy ? close to the optimal one, in the sense that its value, v? (the expected discounted return), is close to maximal, i.e kv? ? v? k is small (v? being the optimal value), for some norm. Controlling the residual kT? v? ? v? k (where T? is the optimal Bellman operator and v? a value function parameterized by ?) over a class of parameterized value functions is a classical approach in value-based RL, and especially in Approximate Dynamic Programming (ADP). Indeed, controlling this residual allows controlling the distance to the optimal value function: generally speaking, we have that C kv? ? v?v? k ? kT? v? ? v? k, (1) 1?? with the policy ?v? being greedy with respect to v? [17, 19]. Some classical ADP approaches actually minimize a projected Bellman residual, k?(T? v? ? v?)k, where ? is the operator projecting onto the hypothesis space to which

v? belongs: Approximate Value Iteration (AVI) [11, 9] tries to minimize this using a fixed-point approach, v?k+1 = ?T? v?k , and it has been shown recently [18] that Least-Squares Policy Iteration (LSPI) [13] tries to minimize it using a Newton approach1 . Notice that in this case (projected residual), there is no general performance bound2 for controlling kv? ? v?v? k. 1

(Exact) policy iteration actually minimizes kT? v ? vk using a Newton descent [10]. With a single action, this approach reduces to LSTD (Least-Squares Temporal Differences) [5], that can be arbitrarily bad in an off-policy setting [20]. 2

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Despite the fact that (unprojected) residual approaches come easily with performance guarantees, they are not extensively studied in the (value-based) literature (one can mention [3] that considers a subgradient descent or [19] that frames the norm of the residual as a delta-convex function). A reason for this is that they lead to biased estimates when the Markovian transition kernel is stochastic and unknown [1], which is a rather standard case. Projected Bellman residual approaches are more common, even if not introduced as such originally (notable exceptions are [16, 18]). An alternative approach consists in maximizing directly the mean value E? [v? (S)] for a userdefined state distribution ?, this being equivalent to directly minimizing kv? ? v? k1,? , see Sec. 2. This suggests defining a class of parameterized policies and optimizing over them, which is the predominant approach in policy search3 [7]. This paper aims at theoretically and experimentally studying these two approaches: maximizing the mean value (related algorithms operate on policies) and minimizing the residual (related algorithms operate on value functions). In that purpose, we place ourselves in the context of policy search algorithms. We adopt this position because we could derive a method that minimizes the residual kT? v? ? v? k1,? over policies and compare to other methods that usually maximize the mean value. On the other hand, adapting ADP methods so that they maximize the mean value is way harder4 . This new approach is presented in Sec. 3, and we show theoretically how good this proxy. In Sec. 4, we conduct experiments on randomly generated generic Markov decision processes to compare both approaches empirically. The experiments are specifically designed to study the influence of the involved concentrability coefficient. Despite the good theoretical properties of the Bellman residual approach, it turns out that it only works well if there is a good match between the sampling distribution and the discounted state occupancy distribution induced by the optimal policy, which is a very limiting requirement. In comparison, maximizing the mean value is rather insensitive to this issue and works well whatever the sampling distribution is, contrary to what suggests the sole related theoretical bound. This study thus suggests that maximizing the mean value, although it doesn?t provide easy theoretical analysis, is a better approach to build efficient and robust RL algorithms.

2

Background

2.1

Notations

Let ?X be the set of probability distributions over a finite set X and Y X the set of applications from X to the set Y . By convention, all vectors are column vectors, except distributions (for left multiplication). A Markov Decision Process (MDP) is a tuple {S, A, P, R, ?}, where S is the finite state space5 , A is the finite action space, P ? (?S)S?A is the Markovian transition kernel (P (s0 —s, a) denotes the probability of transiting to s0 when action a is applied in state s), R ? RS?A is the bounded reward function (R(s, a) represents the local benefitPof doing action a in state s) and ? ? (0, 1) is the discount factor. For v ? RS , we write kvk1,? = s?S ?(s)—v(s)— the ?-weighted ‘1 -norm of v. Notice that when the function v ? RS is componentwise positive, that is v ? 0, the ?-weighted ‘1 -norm of v is actually its expectation with respect to ?: if v ? 0, then kvk1,? = E? [v(S)] = ?v. We will make an intensive use of this basic property in the following. A stochastic policy ? ? (?A)S associates a distribution over actions to each state. The policy-induced reward and transition kernels, R? ? RS and P? ? (?S)S , are defined as R? (s) = E?(.—s) [R(s, A)] and P? (s0 —s) = E?(.—s) [P (s0 —s, A)]. The quality of a policy is quantified by the associated value function v? ? RS : X v? (s) = E[? t R? (St)—S0 = s, St+1 ? P? (.—St)]. t?0

3

A remarkable aspect of policy search is that it does not necessarily rely on the Markovian assumption, but this is out of the scope of this paper (residual approaches rely on it, through the Bellman equation). Some recent and effective approaches build on policy search, such as deep deterministic policy gradient [15] or trust region policy optimization [23]. Here, we focus on the canonical mean value maximization approach. 4 Approximate linear programming could be considered as such but is often computationally intractable [8, 6]. 5 This choice is done for ease and clarity of exposition, the following results could be extended to continuous state and action spaces.

2

The value v? is the unique fixed point of the Bellman operator T? , defined as T? v = R? + ?P? v for any v ? RS . Let define the second Bellman operator T? as, for any v ? RS , T? v = max??(?A)S T? v. A policy ? is greedy with respect to v ? RS , denoted ? ? G(v) if T? v = T? v. There exists an optimal policy ?? that satisfies componentwise v?? ? v? , for all ? ? (?A)S . Moreover, we have that ?? ? G(v?), with v? being the unique fixed point of T? . Finally, for any distribution ? ? ?S , the ?-weighted occupancy measure induced by the policy ? when the initial state is sampled from ? is defined as X d?,? = (1 ? ?)? ? t P?t = (1 ? ?)?(I ? ?P?)?1 ? ?S . t?0

For two distributions ? and ?, we write k ?? k? the smallest constant C satisfying, for all s ? S, ?(s) ? C?(s). This quantity measures the mismatch between the two distributions. 2.2

Maximizing the mean value

Let P be a space of parameterized stochastic policies and let ? be a distribution of interest. The optimal policy has a higher value than any other policy, for any state. If the MDP is too large, satisfying this condition is not

reasonable. Therefore, a natural idea consists in searching for a policy such that the associated value function is as close as possible to the optimal one, in expectation, according to a distribution of interest ?. More formally, this means minimizing kv? ? vk1,? = E? [v? (S) ? v? (S)] ? 0. The optimal value function being unknown, one cannot address this problem directly, but it is equivalent to maximizing E? [v? (S)]. This is the basic principle of many policy search approaches: max J? (?) with J? (?) = E? [v? (S)] = ?v? . ??P

Notice that we used a sampling distribution ? here, possibly different from the distribution of interest ?. Related algorithms differ notably by the considered criterion (e.g., it can be the mean reward rather than the ?-discounted cumulative reward considered here) and by how the corresponding optimization problem is solved. We refer to [7] for a survey on that topic. Contrary to ADP, the theoretical efficiency of this family of approaches has not been studied a lot. Indeed, as far as we know, there is a sole performance bound for maximizing the mean value. Theorem 1 (Scherrer and Geist [22]). Assume that the policy space P is stable by stochastic mixture, that is ??, ? 0 ? P, ?? ? (0, 1), (1 ? ?)? + ?? 0 ? P. Define the ?-greedy-complexity of the policy space P as E? (P) = max min d?,? (T? v? ? T?0 v?). 0 ??P ? ?P

Then, any policy ? that is an -local optimum of J? , in the sense that ?v(1??)?+??0 ? ?v? ? , ? enjoys the following global performance guarantee:

d?,?? 1

(E? (P) +) . ?(v? ? v?) ? (1 ? ?)2 ? ? ?? 0 ? ?,

lim

??0

This bound (as all bounds of this kind) has three terms: an horizon term, a concentrability term and 1 an error term. The term 1?? is the average optimization horizon. This concentrability coefficient (kd?,?? /?k?) measures the mismatch between the used distribution ? and the ?-weighted occupancy measure induced by the optimal policy ?? when the initial state is sampled from the distribution of interest ?. This tells that if ? is the distribution of interest, one should optimize Jd?,?? , which is not feasible, ?? being unknown (in this case, the coefficient is equal to 1, its lower bound). This coefficient can be arbitrarily large: consider the case where ? concentrates on a single starting state (that is ?(s0) = 1 for a given state s0) and such that the optimal policy leads to other states (that is, d?,?? (s0) ¡ 1), the coefficient is then infinite. However, it is also the best concentrability coefficient according to [21], that provides a theoretical and empirical comparison of Approximate Policy Iteration (API) schemes. The error term is E? (P) + , where E? (P) measures the capacity of 3

the policy space to represent the policies being greedy with respect to the value of any policy in P and tells how the computed policy ? is close to a local optimum of J? . There exist other policy search approches, based on ADP rather than on maximizing the mean value, such as Conservative Policy Iteration (CPI) [12] or Direct Policy Iteration (DPI) [14]. The bound of Thm. 1 matches the bounds of DPI or CPI. Actually, CPI can be shown to be a boosting approach maximizing the mean value. See the discussion in [22] for

more details. However, this bound is also based on a very strong assumption (stability by stochastic mixture of the policy space) which is not satisfied by all commonly used policy parameterizations.

3

Minimizing the Bellman residual

Direct maximization of the mean value operates on policies, while residual approaches operate on value functions. To study these two optimization criteria together, we introduce a policy search method that minimizes a residual. As noted before, we do so because it is much simpler than introducing a value-based approach that maximizes the mean value. We also show how good this proxy is to policy optimization. Although this algorithm is new, it is not claimed to be a core contribution of the paper. Yet it is clearly a mandatory step to support the comparison between optimization criteria. 3.1

Optimization problem

We propose to search a policy in P that minimizes the following Bellman residual: min J? (?) with J? (?) = kT? v? ? v? k1,? . ??P

Notice that, as for the maximization of the mean value, we used a sampling distribution ?, possibly different from the distribution of interest ?. From the basic properties of the Bellman operator, for any policy ? we have that T? v? ? v? . Consequently, the ?-weighted ‘1 -norm of the residual is indeed the expected Bellman residual: J? (?) = E? [[T? v?](S) ? v? (S)] = ?(T? v? ? v?). Therefore, there is naturally no bias problem for minimizing a residual here, contrary to other residual approaches [1]. This is an interesting result on its own, as removing the bias in value-based residual approaches is far from being straightforward. This results from the optimization being done over policies and not over values, and thus from v? being an actual value (the one of the current policy) obeying to the Bellman equation6 . Any optimization method can be envisioned to minimize J? . Here, we simply propose to apply a subgradient descent (despite the lack of convexity). Theorem 2 (Subgradient of J?). Recall that given the considered notations, the distribution ?PG(v?) is the state distribution obtained by sampling the initial state according to ?, applying the action being greedy with respect to v? and following the dynamics to the next state. This being said, the subgradient of J? is given by

1 X ??J? (?) = d?,? (s) ? ?d?PG(v?) ,? (s) ?(a—s)? ln ?(a—s)q? (s, a), 1 ? ? s,a P with q? (s, a) = R(s, a) + ? s0 ?S P (s0 —s, a)v? (s0) the state-action value function. Proof. The proof relies on basic (sub)gradient calculus, it is given in the appendix. There are two terms in the negative subgradient ??J? : the first one corresponds to the gradient of J? , the second one (up to the multiplication by ??) is the gradient of J?PG(v?) and acts as a kind of correction. This subgradient can be estimated using Monte Carlo rollouts, but doing so is harder than for classic policy search (as it requires additionally sampling from ?PG(v?) , which requires estimating 6

The property T? v ? v does not hold if v is not the value function of a given policy, as in value-based approaches.

4

the state-action value function). Also, this gradient involves computing the

maximum over actions (as it requires sampling from ?PG(v?) , that comes from explicitly considering the Bellman optimality operator), which prevents from extending easily this approach to continuous actions, contrary to classic policy search. Thus, from an algorithmic point of view, this approach has drawbacks. Yet, we do not discuss further how to efficiently estimate this subgradient since we introduced this approach for the sake of comparison to standard policy search methods only. For this reason, we will consider an ideal algorithm in the experimental section where an analytical computation of the subgradient is possible, see Sec. 4. This will place us in an unrealistically good setting, which will help focusing on the main conclusions. Before this, we study how good this proxy is to policy optimization. 3.2

Analysis

Theorem 3 (Proxy bound for residual policy search). We have that

1

d?,?? J? (?) = 1 d?,?? kT? v? ? v? k1,? . kv? ? v? k1,? ?

1?? ? 1?? ? ? ? Proof. The proof can be easily derived from the analyses of [12], [17] or [22]. We detail it for completeness in the appendix. This bound shows how controlling the residual helps in controlling the error. It has a linear dependency on the horizon and the concentrability coefficient is the best one can expect (according to [21]). It has the same form has the bounds for value-based residual minimization [17, 19] (see also Eq. (1)). It is even better due to the involved concentrability coefficient (the ones for value-based bounds are worst, see [21] for a comparison). Unfortunately, this bound is hardly comparable to the one of Th. 1, due to the error terms. In Th. 3, the error term (the residual) is a global error (how good is the residual as a proxy), whereas in Th. 1 the error term is mainly a local error (how small is the gradient after minimizing the mean value). Notice also that Th. 3 is roughly an intermediate step for proving Th. 1, and that it applies to any policy (suggesting that searching for a policy that minimizes the residual makes sense). One could argue that a similar bound for mean value maximization would be something like: if J? (?) ? ?, then kv? ? v? k1,? ? ?v? ? ?. However, this is an oracle bound, as it depends on the unknown solution v? . It is thus hardly exploitable. The aim of this paper is to compare these two optimization approaches to RL. At a first sight, maximizing directly the mean value should be better (as a more direct approach). If the bounds of Th. 1 and 3 are hardly comparable, we can still discuss the involved terms. The horizon term is better (linear instead of quadratic) for the residual approach. Yet, an horizon term can possibly be hidden in the residual itself. Both bounds imply the same concentrability coefficient, the best one can expect. This is a very important term in RL bounds, often underestimated: as these coefficients can easily explode, minimizing an error makes sense only if it?s not multiplied by infinity. This coefficient suggests that one should use d?,?? as the sampling distribution. This is rarely reasonable, while using instead directly the distribution of interest is more natural. Therefore, the experiments we propose on the next section focus on the influence of this concentrability coefficient.

4

Experiments

We consider Garnet problems [2, 4]. They are a class of randomly built MDPs meant to be totally abstract while remaining representative of the problems that might be encountered in practice. Here, a Garnet G(—S—, —A—, b) is specified by the number of states, the number of actions and the branching factor. For each (s, a) couple, b different next states are chosen randomly and the associated probabilities are set by randomly partitioning the unit interval. The reward is null, except for 10% of states where it is set to a random value, uniform in (1, 2). We set ? = 0.99. ¿

For the policy space, we consider a Gibbs parameterization: P = {?w : ?w (a—s) ? ew ?(s,a) }. The features are also randomly generated, F (d, l). First, we generate binary state-features ?(s) of dimension d, such that l components are set to 1 (the others are thus 0). The positions of the 1?s are 5

selected randomly such that no two states have the same feature. Then, the state-action features, of ¿ dimension d—A—, are classically defined as ?(s, a) = (0 . . . 0 ?(s) 0 . . . 0) , the position of the zeros depending on the action. Notice that in general this policy space is not stable by stochastic mixture, so the bound for policy search does not formally apply. We compare classic policy search (denoted as PS(?)), that maximizes the mean value, and residual policy search (denoted as RPS(?)), that minimizes the mean residual. We optimize the relative objective functions with a normalized gradient ascent (resp. normalized subgradient descent) with a constant learning rate ? = 0.1. The gradients are computed analytically (as we have access to the model), so the following results represent an ideal case, when one can do an infinite number of rollouts. Unless said otherwise, the distribution ? ? ?S of interest is the uniform distribution.
4.1

Using the distribution of interest
0.6
0.4
0.4
0.2 0.0 0
0.2
200 400 600 800 1000 number of iterations
a. Error for PS(?).
0.0 0
200 400 600 800 1000 number of iterations
b. Error for RPS(?).
1.4 1.2 1.0 0.8 0.6 0.4 0.2 0.0 0
kT?v? ? v? k1,?
0.8
0.6
1.4 1.2 1.0 0.8 0.6 0.4 0.2 0.0 0
kT?v? ? v? k1,?
0.8
kv? ? v? k1,? ? kv?k?1 1,?
kv? ? v? k1,? ? kv?k?1 1,?

First, we consider ? = ?. We generate randomly 100 Garnets G(30, 4, 2) and 100 features F (8, 3). For each Garnet-feature couple, we run both algorithms for T = 1000 iterations. For each algorithm, kv? ?v? k1,? we measure two quantities: the (normalized) error kv (notice that as rewards are positive, we ? k1,? have kv? k1,? = ?v?) and the Bellman residual kT? v? ? v? k1,? , where ? depends on the algorithm and on the iteration. We show the results (mean?standard deviation) on Fig. 1.

200 400 600 800 1000 number of iterations

200 400 600 800 1000 number of iterations

c. Residual for PS(?). d. Residual for RPS(?).

Figure 1: Results on the Garnet problems, when ? = ?. Fig. 1.a shows that PS(?) succeeds in decreasing the error. This was to be expected, as it is the criterion it optimizes. Fig. 1.c shows how the residual of the policies computed by PS(?) evolves. By comparing this to Fig. 1.a, it can be observed that the residual and the error are not necessarily correlated: the error can decrease while the residual increases, and a low error does not necessarily involves a low residual. Fig. 1.d shows that RPS(?) succeeds in decreasing the residual. Again, this is not surprising, as it is the optimized criterion. Fig. 1.b shows how the error of the policies computed by RPS(?) evolves. Comparing this to Fig. 1.d, it can be observed that decreasing the residual lowers the error: this is consistent with the bound of Thm. 3. Comparing Figs. 1.a and 1.b, it appears clearly that RPS(?) is less efficient than PS(?) for decreasing the error. This might seem obvious, as PS(?) directly optimizes the criterion of interest. However, when comparing the errors and the residuals for each method, it can be observed that they are not necessarily correlated. Decreasing the residual lowers the error, but one can have a low error with a high residual and vice versa. As explained in Sec. 1, (projected) residual-based methods are prevalent for many reinforcement learning approaches. We consider a policy-based residual rather than a value-based one to ease the comparison, but it is worth studying the reason for such a different behavior. 4.2

Using the ideal distribution d

? The lower the concentrability coefficient k ?,? ? k? is, the better the bounds in Thm. 1 and 3 are. This coefficient is minimized for ? = d?,?? . This is an unrealistic case (?? is unknown), but since we work with known MDPs we can compute this quantity (the model being known), for the sake of a complete empirical analysis. Therefore, PS(d?,??) and RPS(d?,??) are compared in Fig. 2. We highlight the fact that the errors and the residuals shown in this figure are measured respectively to the distribution of interest ?, and not the distribution d?,?? used for the optimization.

6

0.4

0.2

0.2

200 400 600 800 1000 number of iterations

0.0 0

200 400 600 800 1000 number of iterations

200 400 600 800 1000 number of iterations
c. Residual for PS(d?,??).
a. Error for PS(d?,??). b. Error for RPS(d?,??).
1.4 1.2 1.0 0.8 0.6 0.4 0.2 0.0 0
kT?v? ? v? k1,?
0.6
0.4
1.4 1.2 1.0 0.8 0.6 0.4 0.2 0.0 0
kT?v? ? v? k1,?
0.8
0.6
0.0 0
kv? ? v? k1,? ? kv?k?1 1,?
kv? ? v? k1,? ? kv?k?1 1,?
0.8
200 400 600 800 1000 number of iterations
d. Residual for RPS(d?,??).

Figure 2: Results on the Garnet problems, when ? = d?,?? . Fig. 2.a shows that PS(d?,??) succeeds in decreasing the error kv? ? v? k1,? . However, comparing Fig. 2.a to Fig. 1.a, there is no significant gain in using ? = d?,?? instead of ? = ?. This suggests that the dependency of the bound in Thm. 1 on the concentrability coefficient is not tight. Fig. 2.c shows how the corresponding residual evolves. Again, there is no strong correlation between the residual and the error. Fig. 2.d shows how the residual kT? v? ? v? k1,? evolves for RPS(d?,??). It is not decreasing, but it is not what is optimized (the residual kT? v? ? v? k1,d?,?? , not shown, decreases indeed, in a similar fashion than Fig. 1.d). Fig. 2.b shows how the related error evolves. Compared to Fig. 2.a, there is no significant difference. The behavior of the residual is similar for both methods (Figs. 2.c and 2.d). Overall, this suggests that controlling the residual (RPS) allows controlling the error, but that this requires a wise choice for the distribution ?. On the other hand, controlling directly the error (PS) is much less sensitive to this. In other words, this suggests a stronger dependency of the residual approach to the mismatch between the sampling distribution and the discounted state occupancy measure induced by the optimal policy. 4.3

Varying the sampling distribution

This experiment is designed to study the effect of the mismatch between the distributions. We sample 100 Garnets G(30, 4, 2), as well as associated feature sets F (8, 3). The distribution of interest is no longer the uniform distribution, but a measure that concentrates on a single starting state of interest d ? s0 : ?(s0) = 1. This is an adverserial case, as it implies that k ?,? ? k? = ?: the branching factor being equal to 2, the optimal policy ?? cannot concentrate on s0 . The sampling distribution is defined as being a mixture between the distribution of interest and the ideal distribution. For ? ? [0, 1], ?? is defined as ?? = (1 ? ?)? + ?d?,?? . It is straightforward to show that in this case the concentrability coefficient is indeed ?1 (with the convention that 01 = ?):

d?,?? d?,?? (s0) 1 1

= max ; = .

?? (1 ? ?) + ?d (s) ? ? ?,? 0 ? ? For each MDP, the learning (for PS(??) and RPS(??)) is repeated, from the same initial policy, by setting ? = k1 , for k ? [1; 25]. Let ?t,x be the policy learnt by algorithm x (PS or RPS) at iteration t, the integrated error (resp. integrated residual) is defined as T 1 X kv? ? v?t,x k1,? T t=1 kv? k1,?

(resp.

T 1X kT? v?t,x ? v?t,x k1,?). T t=1

Notice that here again, the integrated error and residual are defined with respect to ?, the distribution of interest, and not ?? , the sampling distribution used for optimization. We get an integrated error d ? (resp. residual) for each value of ? = k1 , and represent it as a function of k = k ?,? ?? k? , the concentrability coefficient. Results are presented in Fig. 3, that shows these functions averaged across the 100 randomly generated MDPs (mean?standard deviation as before, minimum and maximum values are shown in dashed line). Fig. 3.a shows the integrated error for PS(??). It can be observed that the mismatch between measures has no influence on the efficiency of the algorithm. Fig. 3.b shows the same thing for RPS(??). The integrated error increases greatly as the mismatch between the sampling measure and the ideal one 7

0.6 0.4 0.2 0.0 0

5 10 15 20 concentrability coefficient

25

a. Integrated error for PS(??).

0.8 0.6 0.4 0.2 0.0 0

5 10 15 20 concentrability coefficient

25

b. Integrated error for RPS(??).

integrated residual

0.8

integrated residual

1.0 integrated error

integrated error

1.0

1.5 1.0 0.5 0.0 0

5 10 15 20 concentrability coefficient

25

c. Integrated residual for PS(??).

1.5 1.0 0.5 0.0 0

5 10 15 20 concentrability coefficient

25

d. Integrated residual for RPS(??).

Figure 3: Results for the sampling distribution ?? . increases (the value to which the error saturates correspond to no improvement over the initial policy). Comparing both figures, it can be observed that RPS performs as well as PS only when the ideal distribution is used (this corresponds to a concentrability coefficient of 1). Fig. 3.c and 3.d show the integrated residual for each algorithm.

It can be observed that RPS consistently achieves a lower residual than PS. Overall, this suggests that using the Bellman residual as a proxy is efficient only if the sampling distribution is close to the ideal one, which is difficult to achieve in general (the ideal distribution d?,?? being unknown). On the other hand, the more direct approach consisting in maximizing the mean value is much more robust to this issue (and can, as a consequence, be considered directly with the distribution of interest). One could argue that the way we optimize the considered objective function is rather naive (for example, considering a constant learning rate). But this does not change the conclusions of this experimental study, that deals with how the error and the Bellman residual are related and with how the concentrability influences each optimization approach. This point is developed in the appendix.

5

Conclusion

The aim of this article was to compare two optimization approaches to reinforcement learning: minimizing a Bellman residual and maximizing the mean value. As said in Sec. 1, Bellman residuals are prevalent in ADP. Notably, value iteration minimizes such a residual using a fixed-point approach and policy iteration minimizes it with a Newton descent. On another hand, maximizing the mean value (Sec. 2) is prevalent in policy search approaches. As Bellman residual minimization methods are naturally value-based and mean value maximization approaches policy-based, we introduced a policy-based residual minimization algorithm in order to study both optimization problems together. For the introduced residual method, we proved a proxy bound, better than value-based residual minimization. The different nature of the bounds of Th. 1 and 3 made the comparison difficult, but both involve the same concentrability coefficient, a term often underestimated in RL bounds. Therefore, we compared both approaches empirically on a set of randomly generated Garnets, the study being designed to quantify the influence of this concentrability coefficient. From these experiments, it appears that the Bellman residual is a good proxy for the error (the distance to the optimal value function) only if, luckily, the concentrability coefficient is small for the considered MDP and the distribution of interest, or one can afford a change of measure for the optimization problem, such that the sampling distribution is close to the ideal one. Regarding this second point, one can change to a measure different from the ideal one, d?,?? (for example, using for ? a uniform distribution when the distribution of interest concentrates on a single state would help), but this is difficult in general (one should know roughly where the optimal policy will lead to). Conversely, maximizing the mean value appears to be insensitive to this problem. This suggests that the Bellman residual is generally a bad proxy to policy optimization, and that maximizing the mean value is more likely to result in efficient and robust reinforcement learning algorithms, despite the current lack of deep theoretical analysis. This conclusion might seems obvious, as maximizing the mean value is a more direct approach, but this discussion has never been addressed in the literature, as far as we know, and we think it to be important, given the prevalence of (projected) residual minimization in value-based RL. 8

2 References

[1] Andr?s Antos, Csaba Szepesv?ri, and R?mi Munos. Learning near-optimal policies with Bellman-residual minimization based fitted policy iteration and a single sample path. Machine Learning, 71(1):89?129, 2008. [2] TW Archibald, KIM McKinnon, and LC Thomas. On the generation of Markov decision processes. Journal of the Operational Research Society, pages 354?361, 1995. [3] Leemon C. Baird. Residual Algorithms: Reinforcement Learning with Function Approximation. In International Conference on Machine Learning (ICML), pages 30?37, 1995. [4] Shalabh Bhatnagar, Richard S Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor-critic algorithms. Automatica, 45(11):2471?2482, 2009. [5] Steven J. Bradtke and Andrew G. Barto. Linear Least-Squares algorithms for temporal difference learning. Machine Learning, 22(1-3):33?57, 1996. [6] Daniela Pucci de Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. Operations research, 51(6):850?865, 2003. [7] Marc Peter Deisenroth, Gerhard Neumann, Jan Peters, et al. A Survey on Policy Search for Robotics. Foundations and Trends in Robotics, 2(1-2):1?142, 2013. [8] Vijay V. Desai, Vivek F. Farias, and Ciamac C. Moallemi. Approximate dynamic programming via a smoothed linear program. Oper. Res., 60(3):655?674, May 2012. [9] Damien Ernst, Pierre Geurts, and Louis Wehenkel. Tree-Based Batch Mode Reinforcement Learning. Journal of Machine Learning Research, 6:503?556, 2005. [10] Jerzy A Filar and Boleslaw Tolwinski. On the Algorithm of Pollatschek and Avi-ltzhak. Stochastic Games And Related Topics, pages 59?70, 1991. [11] Geoffrey Gordon. Stable Function Approximation in Dynamic Programming. In International Conference on Machine Learning (ICML), 1995. [12] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In International Conference on Machine Learning (ICML), 2002. [13] Michail G. Lagoudakis and Ronald Parr. Least-squares policy iteration. Journal of Machine Learning Research, 4:1107?1149, 2003. [14] Alessandro Lazaric, Mohammad Ghavamzadeh, and R?mi Munos. Analysis of a classificationbased policy iteration algorithm. In International Conference on Machine Learning (ICML), 2010. [15] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In International Conference on Learning Representations (ICLR), 2016. [16] Hamid R Maei, Csaba Szepesv?ri, Shalabh Bhatnagar, and Richard S Sutton. Toward off-policy learning control with function approximation. In International Conference on Machine Learning (ICML), 2010. [17] R?mi Munos. Performance bounds in ‘p -norm for approximate value iteration. SIAM journal on control and optimization, 46(2):541?561, 2007. [18] Julien P?rolat, Bilal Piot, Matthieu Geist, Bruno Scherrer, and Olivier Pietquin. Softened Approximate Policy Iteration for Markov Games. In International Conference on Machine Learning (ICML), 2016. [19] Bilal Piot, Matthieu Geist, and Olivier Pietquin. Difference of Convex Functions Programming for Reinforcement Learning. In Advances in Neural Information Processing Systems (NIPS), 2014. 9

[20] Bruno Scherrer. Should one compute the Temporal Difference fix point or minimize the Bellman Residual? The unified oblique projection view. In International Conference on Machine Learning (ICML), 2010. [21] Bruno Scherrer. Approximate Policy Iteration Schemes: A Comparison. In International Conference on Machine Learning (ICML), pages 1314?1322, 2014. [22] Bruno Scherrer and Matthieu Geist. Local Policy Search in a Convex Space and Conservative Policy Iteration as Boosted Policy Search. In European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD), 2014. [23] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In International Conference on Machine Learning (ICML), 2015.

10