Anderson acceleration for reinforcement learning

Matthieu Geist, Bruno Scherrer

European Workshop on Reinforcement Learning 14 (2018) October 2018, Lille, France.

Anderson Acceleration for Reinforcement Learning

Matthieu Geist MATTHIEU.GEIST@UNIV-LORRAINE.FR
Université de Lorraine, CNRS, LIEC, F-57000 Metz, France
(Now at Google Brain)

Bruno Scherrer BRUNO.SCHERRER@INRIA.FR
Université de Lorraine, CNRS, Inria, IECL, F-54000 Nancy, France

Abstract

Anderson (1965) acceleration is an old and simple method for accelerating the computation of a fixed point. However, as far as we know and quite surprisingly, it has never been applied to dynamic programming or reinforcement learning. In this paper, we explain briefly what Anderson acceleration is and how it can be applied to value iteration, this being supported by preliminary experiments showing a significant speed up of convergence, that we critically discuss. We also discuss how this idea could be applied more generally to (deep) reinforcement learning.

Keywords: Reinforcement learning; accelerated fixed point.

1. Introduction

Reinforcement learning (RL) (Sutton and Barto, 1998) is intrinsically linked to fixed-point computation: the optimal value function is the fixed point of the (nonlinear) Bellman optimality operator, and the value function of a given policy is the fixed point of the related (linear) Bellman evaluation operator. Most of the time, these fixed points are computed recursively, by applying repeatedly the operator of interest. Notable exceptions are the evaluation step of policy iteration and the least-squares temporal differences (LSTD) algorithm[1] (Bradtke and Barto, 1996).

Anderson (1965) acceleration (also known as Anderson mixing, Pulay mixing, direct inversion on the iterative subspace or DIIS, among others[2]) is a method that allows speeding up the computation of such fixed points. The classic fixed-point iteration applies repeatdly the operator to the last estimate. Anderson acceleration considers the m previous estimates. Then, it searches for the point that has minimal residual within the subspace spanned by these estimates, and applies the operator to it. This approach has been successfully applied to fields such as electronic structure computation or computational chemistry, but it has never been applied to dynamic programming or reinforcement learning, as far as we know. For more about Anderson acceleration, refer to Walker and Ni (2011), for example.

1. In the realm of deep RL, as far as we know, all fixed points are computed iteratively, there is no LSTD.
2. Anderson acceleration and variations have been rediscovered a number of times in various communities in the last 50 years. Walker and Ni (2011) provide a brief overview of these methods, and a close approach has been recently proposed in the machine learning community (Scieur et al., 2016).

2. Anderson Acceleration for Value Iteration

In this section, we briefly review Markov decision processes, value iteration, and show how Anderson acceleration can be used to speed up convergence.

2.1 Value Iteration

Let Δ_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. A Markov Decision Process (MDP) is a tuple $\{\mathcal{S}, \mathcal{A}, P, \mathcal{R}, \gamma\}$, where $\mathcal{S}$ is the finite state space, $\mathcal{A}$ is the finite action space, $P \in (\Delta_\mathcal{S})^{\mathcal{S}\times\mathcal{A}}$ is the Markovian transition kernel ($P(s'|s,a)$ denotes the probability of transiting to s' when action a is applied in state s), $\mathcal{R} \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is the bounded reward function ($\mathcal{R}(s,a)$ represents the local benefit of doing action a in state s) and $\gamma \in (0,1)$ is the discount factor.

A stochastic policy $\pi \in (\Delta_\mathcal{A})^\mathcal{S}$ associates a distribution over actions to each state (deterministic policies being a special case of this). The policy-induced reward and transition kernels, $\mathcal{R}_\pi \in \mathbb{R}^\mathcal{S}$ and $P_\pi \in (\Delta_\mathcal{S})^\mathcal{S}$, are defined as

$$\mathcal{R}_\pi(s) = \mathbb{E}_{\pi(.|s)}[\mathcal{R}(s,A)] \text{ and } P_\pi(s'|s) = \mathbb{E}_{\pi(.|s)}[P(s'|s,A)].$$

The quality of a policy is quantified by the associated value function $v_\pi \in \mathbb{R}^\mathcal{S}$:

$$v_\pi(s) = \mathbb{E}[\sum_{t\geq 0} \gamma^t \mathcal{R}_\pi(S_t)|S_0 = s, S_{t+1} \sim P_\pi(.|S_t)].$$

The value v_π is the unique fixed point of the Bellman operator T_π, defined as $T_\pi v = \mathcal{R}_\pi + \gamma P_\pi v$ for any $v \in \mathbb{R}^\mathcal{S}$.

Let define the second Bellman operator T as, for any $v \in \mathbb{R}^\mathcal{S}$, $Tv = \max_{\pi\in(\Delta_\mathcal{A})^\mathcal{S}} T_\pi v$. This operator is a γ-contraction (in supremum norm), so the iteration

$$v_{k+1} = Tv_k$$

converges to its unique fixed-point $v_* = \max_\pi v_\pi$, for any $v_0 \in \mathbb{R}^\mathcal{S}$. This is the value iteration algorithm.

2.2 Accelerated Value Iteration

Anderson acceleration is a method that aims at accelerating the computation of the fixed point of any operator. Here, we describe it considering the Bellman operator T, which provides an accelerated value iteration algorithm.

Assume that estimates have been computed up to iteration k, and that in addition to v_k the m previous estimates $v_{k-1}, \ldots, v_{k-m}$ are known. The coefficient vector $\alpha^{k+1} \in \mathbb{R}^{m+1}$ is defined as follows:

$$\alpha^{k+1} = \underset{\alpha\in\mathbb{R}^{m+1}}{\operatorname{argmin}} \left\| \sum_{i=0}^{m} \alpha_i (Tv_{k-m+i} - v_{k-m+i}) \right\| \text{ s.t. } \sum_{i=0}^{m} \alpha_i = 1.$$

Notice that we don't impose a positivity condition on the coefficients. We will consider practically the ℓ_2-norm for this problem, but it could be a different norm (for example ℓ_1 or

ℓ_∞, in which case the optimization problem is a linear program). Then, the new estimate is given by:

$$v_{k+1} = \sum_{i=0}^{m} \alpha_i^{k+1} T v_{k-m+i}.$$

The resulting Anderson accelerated value iteration is summarized in Alg. 1. Notice that the solution to the optimization problem can be obtained analytically for the ℓ_2-norm, using the Karush-Kuhn-Tucker conditions. With the notations of Alg. 1 and writting $\mathbf{1} \in \mathbb{R}^{m_k+1}$ the vector with all components equal to one, it is

$$\alpha^{k+1} = \frac{(\Delta_k^\top \Delta_k)^{-1}\mathbf{1}}{\mathbf{1}^\top (\Delta_k^\top \Delta_k)^{-1}\mathbf{1}}. \tag{1}$$

This can be regularized to avoid ill-conditioning.

Algorithm 1: Anderson Accelerated Value Iteration

given: v_0 and $m \geq 1$
Compute $v_1 = Tv_0$;
for $k = 1, 2, 3 \ldots$ **do**
  Set $m_k = \min(m, k)$;
  Set $\Delta_k = [\delta_{k-m_k}, \ldots, \delta_k] \in \mathbb{R}^{\mathcal{S} \times (m+1)}$ with $\delta_i = Tv_i - v_i \in \mathbb{R}^{\mathcal{S}}$;
  Solve $\min_{\alpha \in \mathbb{R}^{m+1}} \|\Delta_k \alpha\|$ s.t. $\sum_{i=0}^{m_k} \alpha_k = 1$;
  Set $v_{k+1} = \sum_{i=0}^{m_k} \alpha_i T v_{k-m_k+i}$;

The rationale of this acceleration scheme is better understood with an affine operator. We consider here the Bellman evaluation operator T_π. Given the current and the m previous estimates, define

$$\tilde{v}_{k+1}^{\alpha} = \sum_{i=0}^{m} \alpha_i v_{k-m+i} \text{ with } \sum_{i=0}^{m} \alpha_i = 1.$$

Thanks to this constraint, for an affine operator (here T_π), we have that

$$T_\pi \tilde{v}_{k+1}^{\alpha} = \sum_{i=0}^{m} \alpha_i T_\pi v_{k-m+i}.$$

Then, one searches for a vector α (satisfying the constraint) that minimizes the residual

$$\|T_\pi \tilde{v}_{k+1}^{\alpha} - \tilde{v}_{k+1}^{\alpha}\| = \|\sum_{i=0}^{m} \alpha_i (T_\pi v_{k-m+i} - v_{k-m+i})\|.$$

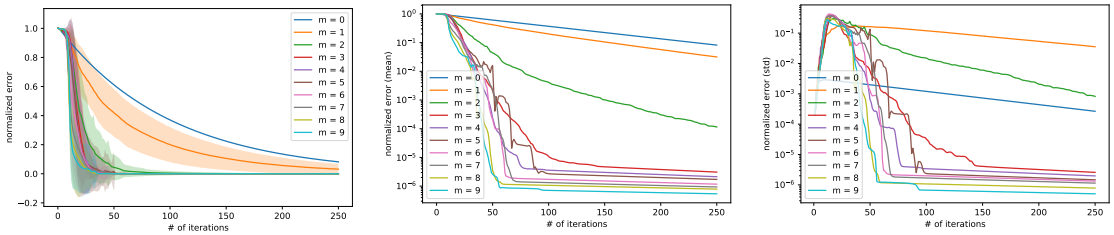
Eventually, the new estimate is obtained by applying the operator to the vector $\tilde{v}_{k+1}^{\alpha}$ of minimal residual.

The same approach can be applied (heuristically) to non-affine operators. The convergence of this scheme has been studied (*e.g.*, Toth and Kelley (2015)) and it can be linked to quasi-Newton methods (Fang and Saad, 2009).

2.3 Preliminary Experimental Results

We consider Garnet problems (Archibald et al., 1995; Bhatnagar et al., 2009). 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 $G(|\mathcal{S}|, |\mathcal{A}|, b)$ 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 generate 100 random MDPs $G(100, 4, 3)$ and set γ to 0.99. For each MDP, we apply value iteration (denoted as $m = 0$ in the graphics) and Anderson accelerated value iteration for m ranging from 1 to 9. The inital value function v_0 is always the null vector. We run all algorithms for 250 iterations, and measure the normalised error for algorithm `alg` at iteration k, $\frac{\|v_* - v_k^{\texttt{alg}}\|_1}{\|v_*\|_1}$, where v_* stands for the optimal value function of the considered MDP.



a. Normalized error .

b. Normalized error (mean, log-scale).

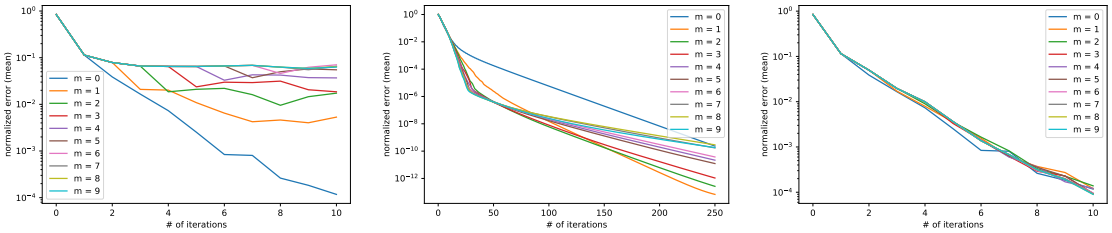
c. Normalized error (std, log-scale).

Figure 1: Results on the Garnet problems.

Fig. 1 shows the results. Fig. 1.a shows how the normalized error evolves with the number of iterations (recall that $m = 0$ stands for classic value iteration). Shaded areas correspond to standard deviations and lines to means (due to randomness of the MDPs, the algorithms being deterministic given the fixed initial value function). Fig. 1.b and 1.c show respectively the mean and the standard deviation of these errors, in a logarithmic scale. One can observe that Anderson acceleration consistently offers a significant speed-up compared to value iteration, and that rather small values of m ($m \approx 5$) seem to be enough.

2.4 Nuancing the Acceleration

We must highlight that the optimal policy is the object of interest, the value function being only a proxy to it. Regarding the value function, its level is not that important, but its relative differences are. This is addressed by the relative value iteration algorithm (Puterman, 1994, Ch. 6.6). For a given state s_0, it iterates as $v_{k+\frac{1}{2}} = Tv_k$, $v_{k+1} = v_{k+\frac{1}{2}} - v_{k+\frac{1}{2}}(s_0)\mathbf{1}$. It usually converges much faster than value iteration (towards $v_* - v_*(s_0)\mathbf{1}$), but the greedy policies resp. to each iterate's estimated values are the same for both algorithms. This scheme can also be easily accelerated with Anderson's approach.



a. Error on greedy policies (accelerated VI).

b. Normalized error (accelerated relative VI).

c. Error on greedy policies (accelerated relative VI).

Figure 2: Additional results.

We provide additional results on Fig. 2 (for the same MDPs as previously). Fig. 2.a shows the error of the greedy policy, that is $\|v_* - v_{\pi_k^{\texttt{alg}}}\|_1 / \|v_*\|_1$, with $\pi_k^{\texttt{alg}}$ being greedy respectively to $v_k^{\texttt{alg}}$, for the first 10 iterations (same data as for Fig. 1). This is what we're really interested in. One can observe that value iteration provides more quickly better solutions than Anderson acceleration. This is due to the fact that if the level of the value function converges slowly, its relative differences converge more quickly.

So, we compare relative value iteration and its accelerated counterpart in Fig. 2.b (normalized error of the estimate, not of the greedy policy), to be compared to Fig. 1.b. There is still an acceleration with Anderson, at least at the beginning, but the speed-up is much less than in Fig. 1. We compare the error on greedy policies for the same setting in Fig. 2.c, and all approaches perform equally well.

3. Anderson Acceleration for Reinforcement Learning

So, the advantage of Anderson acceleration applied to exact value iteration on simple Garnet problems is not that clear. Yet, it could still be interesting for policy evaluation or in the approximate setting. We discuss briefly its possible applications to (deep) RL.

3.1 Approximate Dynamic Programming

Anderson acceleration could be applied to approximate dynamic programming and related methods. For example, the well-known DQN algorithm (Mnih et al., 2015) is nothing else than a (very smart) approximate value iteration approach. A state-action value function Q is estimated (rather than a value function), and this function is represented as a neural network. A target network Q_k is maintained, and the Q-function is estimated by solving the least-squares problem (for the memory buffer $\{(s_i, a_i, r_i, s'_i)_{1 \leq i \leq n}\}$)

$$\frac{1}{n} \sum_{i=1}^{n} (y_i - Q_\theta(s_i, a_i))^2 \text{ with } y_i = r_i + \gamma \max_{a \in \mathcal{A}} Q_k(s'_i, a).$$

Anderson acceleration can be applied directly as follows. Assume that the $m+1$ previous target networks $Q_k, \ldots, Q_{k-m}$ are maintained. Define for $k - m \leq j \leq k$

$$\delta_j = [r_1 + \gamma \max_a Q_j(s'_1, a) - Q_j(s_1, a_1), \ldots, r_n + \gamma \max_a Q_j(s'_n, a) - Q_j(s_n, a_n)]^\top \in \mathbb{R}^n$$

and $\Delta_k = [\delta_{k-m}, \dots, \delta_k] \in \mathbb{R}^{n\times(m+1)}$. Solve α_{k+1} as in Eq. (1) and define for all $1 \leq i \leq n$

$$y_i = \sum_{j=0}^{n} \alpha_j (r_i + \gamma \max_{a\in\mathcal{A}} Q_{k-m+j}(s'_i, a)).$$

So, Anderson acceleration would modify the targets in the regression problem, the necessary coefficients being obtained with a cheap least-squares (given m is small enough, as suggested by our preliminary experiments). Notice that the estimate α^{k+1} is biased, as being the solution to a residual problem with sampled transitions. However, if a problem, this could probably be handled with instrumental variables, giving an LSTD-like algorithm (Bradtke and Barto, 1996). Variations of this general scheme could also be envisionned, for example by computing the α vector on a subset of the memory replay or even on the current mini-batch, or by considering variations of Anderson acceleration such as the one of Henderson and Varadhan (2018).

This acceleration scheme could be more generally applied to approximate modified policy iteration, or AMPI (Scherrer et al., 2015), that generalizes both approximate policy and value iterations. Modified policy iteration is similar to policy iteration, except that instead of computing the fixed point in the evaluation step, the Bellman evaluation operator is applied p times ($p = 1$ gives value iteration, $p = \infty$ policy iteration), the improvement step (computing the greedy policy) being the same (up to possible approximation). In the approximate setting, the evaluation step is usually performed by performing the regression of p-step returns, but it could be done by applying repeatedly the evaluation operator, this being combined with Anderson acceleration (much like DQN, but with T_π instead of T).

3.2 Policy Optimization

Another popular approach in reinforcement learning is policy optimization, or direct policy search (Deisenroth et al., 2013), that maximizes $J(w) = \mathbb{E}_{S\sim\mu}[v_{\pi_w}(S)]$ (or a proxy), for a user-defined state distribution μ, over a class of parameterized policies. This is classically done by performing a gradient ascent:

$$w_{k+1} = w_k + \eta \nabla_w J(w)|_{w=w_k}. \tag{2}$$

This gradient is given by $\nabla_w J(w) = \mathbb{E}_{S\sim d_{\mu,\pi_w}, A\sim\pi}[Q_{\pi_w}(S, A)\nabla_w \ln \pi_w(A|S)]$. Thus, it depends on the state-action value function of the current policy. This gradient can be estimated with rollouts, but it is quite common to estimate the Q-function itself. Related approaches are known as actor-critic methods (the actor being the policy, and the critic the Q-function). It is quite common to estimate the critic using a SARSA-like approach, especially in deep RL. In other words, the critic is estimated by applying repeatedly the Bellman evaluation operator. Therefore, Anderson acceleration could be applied, in the same spirit as what we described for DQN.

Yet, Anderson acceleration could also be used to speed up the convergence of the policy. Consider the gradient ascent in Eq. (2). This can be seen as a fixed-point iteration to solve $w = w+\eta\nabla_w J(w)$. Anderson acceleration could thus be used to speed it up. Seeing gradient descent as a fixed point is not new (Jung, 2017), nor is applying Anderson acceleration to speed it up (Scieur et al., 2016; Xie et al., 2018). Yet, it has never been applied to policy optimization, as far as we know.

References

Donald G Anderson. Iterative procedures for nonlinear integral equations. *Journal of the ACM (JACM)*, 12(4):547–560, 1965.

TW Archibald, KIM McKinnon, and LC Thomas. On the generation of Markov decision processes. *Journal of the Operational Research Society*, pages 354–361, 1995.

Shalabh Bhatnagar, Richard S Sutton, Mohammad Ghavamzadeh, and Mark Lee. Natural actor-critic algorithms. *Automatica*, 45(11):2471–2482, 2009.

Steven J. Bradtke and Andrew G. Barto. Linear Least-Squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

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.

Haw-ren Fang and Yousef Saad. Two classes of multisecant methods for nonlinear acceleration. *Numerical Linear Algebra with Applications*, 16(3):197–221, 2009.

Nicholas C Henderson and Ravi Varadhan. Damped anderson acceleration with restarts and monotonicity control for accelerating em and em-like algorithms. *arXiv preprint arXiv:1803.06673*, 2018.

Alexander Jung. A fixed-point of view on gradient methods for big data. *Frontiers in Applied Mathematics and Statistics*, 3:18, 2017.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley-Interscience, 1994.

Bruno Scherrer, Mohammad Ghavamzadeh, Victor Gabillon, Boris Lesner, and Matthieu Geist. Approximate modified policy iteration and its application to the game of tetris. *Journal of Machine Learning Research*, 16:1629–1676, 2015.

Damien Scieur, Alexandre d'Aspremont, and Francis Bach. Regularized nonlinear acceleration. In *Advances In Neural Information Processing Systems*, pages 712–720, 2016.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press Cambridge, 1998.

Alex Toth and CT Kelley. Convergence analysis for anderson acceleration. *SIAM Journal on Numerical Analysis*, 53(2):805–819, 2015.

Homer F Walker and Peng Ni. Anderson acceleration for fixed-point iterations. *SIAM Journal on Numerical Analysis*, 49(4):1715–1735, 2011.

Guangzeng Xie, Yitan Wang, Shuchang Zhou, and Zhihua Zhang. Interpolatron: Interpolation or extrapolation schemes to accelerate optimization for deep neural networks. *arXiv preprint arXiv:1805.06753*, 2018.