Optimistic Linear Programming gives Logarithmic Regret for Irreducible MDPs

Authored by:

Peter L. Bartlett Ambuj Tewari

Abstract

We present an algorithm called Optimistic Linear Programming (OLP) for learning to optimize average reward in an irreducible but otherwise unknown Markov decision process (MDP). OLP uses its experience so far to estimate the MDP. It chooses actions by optimistically maximizing estimated future rewards over a set of next-state transition probabilities that are close to the estimates: a computation that corresponds to solving linear programs. We show that the total expected reward obtained by OLP up to time \$T\$ is within \$C(P)log T\$ of the reward obtained by the optimal policy, where C(P) is an explicit, MDP-dependent constant. OLP is closely related to an algorithm proposed by Burnetas and Katehakis with four key differences: OLP is simpler, it does not require knowledge of the supports of transition probabilities and the proof of the regret bound is simpler, but our regret bound is a constant factor larger than the regret of their algorithm. OLP is also similar in flavor to an algorithm recently proposed by Auer and Ortner. But OLP is simpler and its regret bound has a better dependence on the size of the MDP.

1 Paper Body

Decision making under uncertainty is one of the principal concerns of Artificial Intelligence and Machine Learning. Assuming that the decision maker or agent is able to perfectly observe its own state, uncertain systems are often modeled as Markov decision processes (MDPs). Given complete knowledge of the parameters of an MDP, there are standard algorithms to compute optimal policies, i.e., rules of behavior such that some performance criterion is maximized. A frequent criticism of these algorithms is that they assume an explicit description of the MDP which is seldom available. The parameters constituting the description are themselves estimated by simulation or experiment and are thus not known with complete reliability. Taking this into account brings us to the well known exploration vs. exploitation trade-off. On one hand, we would like to explore the system as well as we can to obtain reliable knowledge about

the system parameters. On the other hand, if we keep exploring and never exploit the knowledge accumulated, we will not behave optimally. Given a policy ?, how do we measure its ability to handle this trade-off? Suppose the agent gets a numerical reward at each time step and we measure performance by the accumulated reward over time. Then, a meaningful quantity to evaluate the policy ? is its regret over time. To understand what regret means, consider an omniscient agent who knows all parameters of the MDP accurately and behaves optimally. Let VT be the expected reward obtained by this agent up to time T . Let VT? denote the corresponding quantity for ?. Then the regret RT? = VT ? VT? measures how much ? is hurt due to its incomplete knowledge of the MDP up to time T . If we can show that the regret RT? grows slowly with time T , for all MDPs in a sufficiently big class, then we can safely conclude that ? is making a judicious trade-off between exploration and exploitation. It is rather remarkable that 1

for this notion of regret, logarithmic bounds have been proved in the literature [1,2]. This means that there are policies? with RT? = $O(\log T)$. Thus the per-step regret RT? /T goes to zero very quickly. Burnetas and Katehakis [1] proved that for any policy? (satisfying certain reasonable assumptions) RT?? CB (P) log T where they identified the constant CB (P). This constant depends on the transition function P of the MDP1. They also gave an algorithm (we call it BKA) that achieves this rate and is therefore optimal in a very strong sense. However, besides assuming that the MDP is irreducible (see Assumption 1 below) they assumed that the support sets of the transition distributions pi (a) are known for all state-action pairs. In this paper, we not only get rid of this assumption but our optimistic linear programming (OLP) algorithm is also computationally simpler. At each step, OLP considers certain parameters in the vicinity of the estimates. Like BKA, OLP makes optimistic choices among these. But now, making these choices only involves solving linear programs (LPs) to maximize linear functions over L1 balls. BKA instead required solving non-linear (though convex) programs due to the use of KL-divergence. Another benefit of using the L1 distance is that it greatly simplifies a significant part of the proof. The price we pay for these advantages is that the regret of OLP is C(P) log T asymptotically, for a constant C(P)? CB(P). We should note here that a number of algorithms in the literature have been inspired by the ?optimism in the face of uncertainty? principle [3]?[7]. The algorithm of Auer and Ortner (we refer to it as AOA) is another logarithmic regret algorithm for irreducible 2 MDPs. AOA does not solve an optimization problem at every time step but only when a confidence interval is halved. But then the optimization problem they solve is more complicated because they find a policy to use in the next few time steps by optimizing over a set of MDPs. The regret of AOA is $CA(P) \log T$ where CA(P) = c

for some universal constant c. Here —S—, —A— denote the state and action space size, Tw (P) is the worst case hitting time over deterministic policies (see Eqn. (12)) and ?? (P) is the difference between the long term average return

of the best policy and that of the next best policy. The constant ?(P) is also defined in terms of hitting times. Under Auer and Ortner?s assumption of bounded rewards, we can show that the constant for OLP satisfies C(P)?

```
2—S——A—T (P )2 . ?? (P )
(2)
```

Here T (P) is the hitting time of an optimal policy is therefore necessarily smaller than Tw (P). We get rid of the dependence on ?(P) while replacing Tw (P) with T (P)2. Most importantly, we significantly improve the dependence on the state space size. The constant ??(P) can roughly be thought of as the minimum (over states) difference between the quality of the best and the second best action (see Eqn. (9)). The constants ??(P) and ??(P) are similar though not directly comparable. Nevertheless, note that C(P) depends inversely on ??(P) not ??(P)2.

2

Preliminaries

Notation for MDP parameters is defined in Section 2 below. Auer & Ortner prove claims for unichain MDPs but their usage seems non-standard. The MDPs they call unichain are called irreducible in standard textbooks (for example, see [9, p. 348]) 2

2

policies that take actions in D. Probability and expectation under a policy ?, transition function P and starting state i0 will be denoted by P?,P and E?,P respectively. Given history ?t , let Nt (i), i0 i0 Nt (i, a) and Nt (i, a, j) denote the number of occurrences of the state i, the pair (i, a) and the triplet (i, a, j) respectively in ?t . We make the following irreducibility assumption regarding the MDP. Assumption 1. For all ? ? ?D , the transition matrix P ? = (pi,j (?(i)))i,j?S is irreducible (i.e. it is possible to reach any state from any other state). Consider the rewards accumulated by the policy ? before time T , T ?1 X

```
\begin{array}{l} {\rm VT?}\;({\rm i0}\;,\,{\rm P}\;) := {\rm E?,P}\;{\rm i0}\;[\\ {\rm r(st}\;,\,{\rm at}\;)]\;,\\ {\rm t=}0 \end{array}
```

where at is the random variable representing the action taken by ? at time t. Let VT (i0 , P) be the maximum possible sum of expected rewards before

time T, VT (i0, P) := $\sup VT$? (i0, P). ???

The regret of a policy? at time T is a measure of how well the expected rewards of? compare with the above quantity, RT? (i0 , P) := VT (i0 , P)? VT? (i0 , P) . Define the long term average reward of a policy? as V? (i0 , P) . ?? (i0 , P) := lim inf T T?? T Under assumption 1, the above limit exists and is independent of the starting state i0 . Given a restricted set D? A of actions, the gain or the best long term average performance is ?(P, D) := sup ?? (i0 , P) . ???(D)

As a shorthand, define ?? (P) := ?(P, A). 2.1

Optimality Equations

A restricted problem $(P,\,D)$ is obtained from the original MDP by choosing subsets D(i)? A(i) and setting D=?i?S D(i). The transition and reward functions of the restricted problems are simply the restrictions of <math display="inline">P and r to D. Assumption 1 implies that there is a bias vector $h(P,\,D)=\{h(i;\,P,\,D)\}i?S$ such that the gain $?(P,\,D)$ and bias $h(P,\,D)$ are the unique solutions to the average reward optimality equations: ?i ? S, $?(P,\,D)+h(i;\,P,\,D)=\max\,[r(i,\,a)+hpi\,(a),\,h(P,\,D)i]$. (3) a?D(i)

We will use h (P) to denote h(P, A). Also, denote the infinity norm kh? (P)k? by H? (P). Note that if h? (P) is a solution to the optimality equations and e is the vector of ones, then h? (P) + ce is also a solution for any scalar c. We can therefore assume ?i? ? S, h? (i? ; P) = 0 without any loss of generality. It will be convenient to have a way to denote the quantity inside the ?max? that appears in the optimality equations. Accordingly, define L(i, a, p, h) := r(i, a) + hp, hi, ? L (i; P, D) := max L(i, a, pi (a), h(P, D)) . a?D(i)

To measure the degree of suboptimality of actions available at a state, define ?? (i, a; P) = L? (i; P, A) ? L(i, a, pi (a), h? (P)) . Note that the optimal actions are precisely those for which the above quantity is zero. O(i; P, D) := {a ? D(i) : ?? (i, a; P) = 0} , O(P, D) := ?i?S O(i; P, D) . Any policy in O(P, D) is an optimal policy, i.e., ?? ? O(P, D), ?? (P) = ?(P, D) . 3 2.2

Critical pairs

From now on, ?+ will denote the probability simplex of dimension determined by context. For a suboptimal action a ? / O(i; P, A), the following set contains probability distributions q such that if pi (a) is changed to q, the quality of action a comes within of an optimal action. Thus, q makes a look almost optimal: MakeOpt(i, a; P,) := {q??+: L(i, a, q, h? (P))? L? (i; P, A)?}. (4) Those suboptimal state-action pairs for which MakeOpt is never empty, no matter how small is, play a crucial role in determining the regret. We call these critical state-action pairs, $Crit(P) := \{(i, a) : a? / O(i; P, A)? (?; 0, MakeOpt(i, a; P,) 6=?)\}$. (5) Define the function, Crit(P) := (i, a) : a? / O(i; P, A)? (?; 0, A) MakeOpt(i, a; P,)}. (6) To make sense of this definition, consider Crit(P) := (i, a) : A MakeOpt(i, a; P,)}. (6) To make sense of this definition, consider Crit(P) := (i, a) : A MakeOpt(i, a; P,)}. (6) To make the suboptimal action a look -optimal. Taking the limit of this as decreases gives us a quantity that also plays a crucial role in determining the regret, Crit(P) := (i, a) : A and Crit(P) := (i, a) : A are definition, consider Crit(P) := (i, a) : A and Crit(P) := (i, a) : A are definition, consider Crit(P) := (i, a) : A are definition, consider Crit(P) := (i, a) : A are definition, consider Crit(P) := (i, a) : A are definition, consider Crit(P) := (i, a) : A are definition, consider Crit(P) := (i, a) : A are definition and Crit(P) := (i, a) : A are definition and Crit(P) : A are defini

Intuitively, if K(i, a; P) is small, it is easy to confuse a suboptimal action with an optimal one and so it should be difficult to achieve small regret. The constant that multiplies log T in the regret bound of our algorithm OLP (see Algorithm 1 and Theorem 4 below) is the following: X 2?? (i, a; P) . (8) C(P) := K(i, a; P) (i,a)?Crit(P)

This definition might look a bit hard to interpret, so we give an upper bound on C(P) just in terms of the infinity norm H? (P) of the bias and ?? (P). This latter quantity is defined below to be the minimum degree of suboptimality of a critical action. Proposition 2. Suppose A(i) = A for all i? S. Define ?? $(P) := \min ??$ (i, a; P). (9) (i,a)? Crit(P)

```
Then, for any P , C(P) ?
```

See the appendix for a proof. 2.3

Hitting times

The following constant is the minimum hitting time among optimal policies: T (P) := min T? (P) . ??O(P,D) (11)

The following constant is defined just for comparison with results in [2]. It is the worst case hitting time over all policies: Tw $(P) := \max T? (P)$. (12) ???D

We can now bound C(P) just in terms of the hitting time T(P) and ??(P). Proposition 3. Suppose A(i) = A for all i? S and that r(i, a)? [0, 1] for all i? S, a? A. Then for any P, 2—S—A—T(P)2. C(P)? ?? (P) See the appendix for a proof. 4

The optimistic LP algorithm and its regret bound

Algorithm 1 Optimistic Linear Programming 1: for $t=0,\,1,\,2,\,\ldots$ do 2: st? current state 3: 4: 5: 6: 7: 8: 9: 10:

. Compute solution for ?empirical MDP? excluding ?undersampled? actions 1+Nt (i,a,j) ?i, j ? S, a ? A(i), p?ti,j (a) ? —A(i)—+N t (i,a) ?i ? S, Dt (i) ? {a ? A(i) : Nt (i, a) ? log2 Nt (i)} ? t, ? ? t ? solution of the optimality equations (3) with P = P? t , D = Dt h . Compute indices of all actions for the current state q log t ? t i : k? } ?a ? A(st), Ut (st , a) ? supq??+ {r(st , a) + hq, h ptst (a) ? qk1 ? N2t (s t ,a)

11: 12: . Optimal actions (for the current problem) that are about to become ?undersampled? 13: ?1t ? {a ? O(st ; P? t , Dt) : Nt (st , a) ; log2 (Nt (st) + 1)} 14: 15: . The index maximizing actions 16: ?2t ? arg maxa?A(st) Ut (st , a) 17: 18: if ?1t = O(st ; P? t , Dt) then 19: at ? any action in ?1t 20: else 21: at ? any action in ?2t 22: end if 23: end for

Algorithm 1 is the Optimistic Linear Programming algorithm. It is inspired by the algorithm of Burnetas and Katehakis [1] but uses L1 distance instead of KL-divergence. At each time step t, the algorithm computes the empirical

estimates for transition probabilities. It then forms a restricted problem ignoring relatively undersampled actions. An action a? A(i) is considered ?undersam? t,?? t might be misleading due to estimation errors. pled? if Nt (i, a); log2 Nt (i). The solutions h To avoid being misled by empirical samples we compute optimistic ?indices? Ut (st, a) for all legal actions a? A(st) where st is the current state. The index for action a is computed by looking at an L1 -ball around the empirical estimate p?tst (a) and choosing a probability distribution q that max? t). Note that if the estimates were perfect, we would take an action maximizing imizes L(i, a, q, h t? L(i, a, p?st (a), ht). Instead, we take an action that maximizes the index. There is one case where we are forced not to take an index-maximizing action. It is when all the optimal actions of the current problem are about to become undersampled at the next time step. In that case, we take one of these actions (steps 18?22). Note that both steps 7 and 10 can be done by solving LPs. The LP for solving optimality equations can be found in several textbooks (see, for example, [9, p. 391]). The LP in step 10 is even simpler: the L1 ball has only 2—S— vertices and so we can maximize over them efficiently. Like the original Burnetas-Katehakis algorithm, the modified one also satisfies a logarithmic regret bound as stated in the following theorem. Unlike the original algorithm, OLP does not need to know the support sets of the transition distributions. Theorem 4. Let? denote the policy implemented by Algorithm 1. Then we have, for all i0? S and for all P satisfying Assumption 1, RT? (i0, P)? C(P), log TT?? where C(P) is the MDP-dependent constant defined in (8). lim sup

```
Proof. From Proposition 1 in [1], it follows that X X ? RT? (i0, P) = E?,P
i0 [NT (i, a)]? (i, a; P) + O(1) . i?S a?O(i;P,A) /
   5
   (13)
   Define the event
   ? t ? h? (P)k? ? ? O(P? t , Dt) ? O(P)} . At := \{kh
   Define, NT1 (i, a;) :=
   T ?1 X
   1 [(st, at) = (i, a) ? At ? Ut (i, a) ? L? (i; P, A) ? 2],
   NT2 (i, a;) :=
   T ?1 X
   1 [(st, at) = (i, a) ? At ? Ut (i, a) ; L? (i; P, A) ? 2],
   NT3() :=
   T ?1 X
   1 A?t,
   where A?t denotes the complement of At . For all i, 0, NT (i, a)? NT1 (i,
a; + NT2 (i, a; + NT3 () .
   (15)
```

The result then follows by combining (13) and (15) with the following three propositions and then letting? 0 sufficiently slowly. Proposition 5. For all P and i0? S, we have lim lim sup

```
?0 T ??
   Χ
   X
   i?S a?O(i;P,A) /
   1 E?,P i0 [NT (i, a; )] ? ? (i, a; P ) ? C(P ) . log T
   Proposition 6. For all P, i0, i? S, a? / O(i; P, A) and sufficiently small,
we have 2 \text{ E?,P i0 [NT (i, a; )]} = o(\log T).
   Proposition 7. For all P satisfying Assumption 1, i0? S and j. 0, we have 3
E?,P i0 [NT ()] = o(log T).
   4
   Proofs of auxiliary propositions
   We prove Propositions 5 and 6. The proof of Proposition 7 is almost the same
as that of Proposition 5 in [1] and therefore omitted (for details, see Chapter
6 in the first author?s thesis [8]). The proof of Proposition 6 is considerably
simpler (because of the use of L1 distance rather than KL-divergence) than the
analogous Proposition 4 in [1]. Proof of Proposition 5. There are two cases
depending on whether (i, a)? Crit(P) or not. If (i, a)? / Crit(P), there is an
0 ; 0 such that MakeOpt(i, a; P, 0) = ?. On the event At (recall the ? t i ?
hq, h? (P)i—? for any q?? + . Therefore, definition given in (14)), we have
-hq, h ? t i Ut (i, a) ? sup {r(i, a) + hq, h q??+
   ? \sup \{r(i, a) + hq, h? (P)i\} + q?? + ?
   ; L (i; P, A) ? 0 + ; L? (i; P, A) ? 2 provided that 3 ; 0
   [? MakeOpt(i, a; P, 0) = ?]
   Therefore for i = 0 /3, NT1 (i, a; ) = 0. Now suppose (i, a) ? Crit(P). The
event Ut (i, a)? L? (i; P, A)? 2 is equivalent to
   2\,\log\,t + t\,2\ ?\ t\ i\ ?\ L?\ (i;\,P,\,A)\ ?\ 2\ .\ ?\ r(i,\,a) + hq,\,h\ ?q\ ?\ ?\ s.t.\ k?\ pi\ (a)
? qk1 ? Nt (i, a) ? t i ? hq, h? (P )i— ? and thus the above implies On the
event At , we have —hq, h
   P, A) ? 3) . Nt (i, a) 6
   Recalling the definition (6) of Ji,a (p; P, ), we see that this implies Ji,a (?
pti (a); P, 3)?
   2 log t . Nt (i, a)
   We therefore have, NT1 (i, a; )
   2 \log t + ? 1 (st, at) = (i, a) ? Ji, a (? pi (a); P, 3) ? Nt (i, a) t=0
   T ? 1 X 2 log t + ? ? 1 (st, at) = (i, a) ? Ji,a (pi (a); P, 3) ? Nt (i, a) t=0
```

1 (st, at) = (i, a) ? Ji,a (pi (a); P, 3) ; Ji,a (? pti (a); P, 3) + ?

T ?1 X (16)

```
where? ; 0 is arbitrary. Each time the pair (i, a) occurs Nt (i, a) increases
by 1, so the first count is no more than 2 log T . (17) Ji,a (pi (a); P, 3)?
To control the expectation of the second sum, note that continuity of Ji,a in its
first argument implies that there is a function f such that f (?) ; 0 for ? ; 0,
f (?) ? 0 as ? ? 0 and Ji,a (pi (a); P, 3) ; Ji,a (? pti (a); P, 3) + ? implies
that kpi (a) ? p?ti (a)k1 ¿ f (?). By a Chernoff-type bound, we have, for some
constant C1, ?ti (a)k1; f (?) — Nt (i, a) = m] ? C1 exp(?mf (?)2). P?,P i0
[kpi (a)? p and so the expectation of the second sum is no more than T?1 X
   E?,P i0 [
   ? X
   C1 \exp(\text{?Nt (i, a)f (?)2})]?
   C1 \exp(?mf(?)2) =
   m=1
   C1 . 1 ? \exp(?f(?)2)
   Combining the bounds (17) and (18) and plugging them into (16), we get 1
E?,P i0 [NT (i, a; )]?
   2 log T C1 + . Ji,a (pi (a); P, 3) ? ? 1 ? exp(?f (?)2)
   Letting?? 0 sufficiently slowly, we get that for all i, 0, 1 E?,P i0 [NT (i, a;
)] ?
   2 \log T + o(\log T). Ji,a (pi (a); P, 3)
   Therefore, lim lim sup
   ?0 T ??
   1 E?,
P 2 2 i<br/>0 [NT (i, a; )] ? lim = , ?0 Ji,a (pi (a); P, 3) log T K<br/>(i, a; P )
   where the last equality follows from the definition (7) of K(i, a; P). The
result now follows by summing over (i, a) pairs in Crit(P). Proof of Proposition
6. Define the event A0t (i, a; ) := {(st, at) = (i, a)? At? Ut (i, a); L? (i; P,
A) ? 2}, so that we can write NT2 (i, a; )
   T ?1 X
   1 [A0t (i, a; )].
   (19)
   t=0
   Note that on A0t (i, a; ), we have ?1t ? O(i; P? t, Dt)? O(i; P, A). So,
a? / O(i; P, A). But a was taken at time t, so it must have been in ?2t which
means it maximized the index. Therefore, for all optimal actions a? ? O(i; P,
A), we have, on the event A0t (i, a; ), Ut (i, a? )? Ut (i, a); L? (i; P, A)? 2.
   Since L? (i; P, A) = r(i, a?) + hpi (a?), h? (P)i, this implies s 2 log t? t
i; hpi (a?), h? (P) i? 2. ? hq, h?q? ?+, kq? p?ti (a?) k1? Nt (i, a?)?
t i? hq, h? (P)i—?. We therefore have, for any a?? O(i; P, A), Moreover,
on the event At,—hq, h s () 2 log t 0 + t?? At (i, a;)??q??, kq?p?i
(a)k1? ? hq, h (P )i ; hpi (a), h (P )i ? Nt (i, a) s ) ( 2 log t
   + t ? kq ? pi (a)k1 ; ? ? ?q ? ? , kq ? p?i (a)k1 ? Nt (i, a) kh (P )k? s ( )
```

2 log t? k? pti (a) ? pi (a) k1 ; ? + h (P) Nt (i, a) s) (t [2 log t

```
 \begin{array}{l} t ? + Nt \ (i, \, a) = m ? \ k? \ pi \ (a) ? \ pi \ (a)k1 \ \emph{\i} ? \ kh \ (P)k? \ Nt \ (i, \, a) \ m=1 \\ Using a Chernoff-type bound, we have, for some constant C1 , P?,P pti (a) ? pi (a)k1 \ \emph{\i} ? \ - Nt \ (i, \, a) = m] ? \ C1 \ exp(?m? \ 2 \ /2) \ . \ i0 \ [k? \ Using a union bound, we therefore have, t X ? m 2 \\ r \ !2 ? 2 \log t ? m \\ + ? \ (P)k \ kh ? \ m=1 ? \\ ? \ 2m \log t \ 1 \ C1 \ X \ m2 = o ? \ ? \ exp ? \ . \ t \ m=1 \ 2kh? \ (P)k2? \ kh? \ (P)k? \ t \\ 0 \ P?,P \ i0 \ [At \ (i, \, a; \,)] ? \end{array}
```

Combining this with (19) proves the result.

2 References

C1 exp ??

[1] Burnetas, A.N. & Katehakis, M.N. (1997) Optimal adaptive policies for Markov decision processes. Mathematics of Operations Research 22(1):222?255 [2] Auer, P. & Ortner, R. (2007) Logarithmic online regret bounds for undiscounted reinforcement learning. Advances in Neural Information Processing Systems 19. Cambridge, MA: MIT Press. [3] Lai, T.L. & Robbins, H. (1985) Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics 6(1):4?22. [4] Brafman, R.I. & Tennenholtz, M. (2002) R-MAX a general polynomial time algorithm for near-optimal reinforcement learning. Journal of Machine Learning Research 3:213?231. [5] Auer, P. (2002) Using confidence bounds for exploitation-exploration trade-offs. Journal of Machine Learning Research 3:397?422. [6] Auer, P., Cesa-Bianchi, N. & and Fischer, P. (2002) Finite-time analysis of the multiarmed bandit problem. Machine Learning 47(2-3):235-256. [7] Strehl, A.L. & Littman, M. (2005) A theoretical analysis of model-based interval estimation. In Proceedings of the Twenty-Second International Conference on Machine Learning, pp. 857-864. ACM Press. [8] Tewari, A. (2007) Reinforcement Learning in Large or Unknown MDPs. PhD thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley. [9] Puterman, M.L. (1994) Markov Decision Processes: Discrete Stochastic Dynamic Programming. New York: John Wiley and Sons.